www.euro6ix.net

Title:	Document Version:
Technical Report TR4.1A.3 **Deployment of Static VPNs and Security**	1.2

Project Number:	Project Acronym:	Project Title:
IST-2001-32161	Euro6IX	European IPv6 Internet Exchanges Backbone

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type* - Security**:
31/12/2002	25/02/2003	R – PU

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other
** Security Class: PU- Public, PP– Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Responsible and Editor/Author:	Organization:	Contributing WP:
Antonio F. Gómez-Skarmeta	UMU	WP4

Authors (organizations) in alphabetical order:

Cesar Olvera (Consulintel), Cristina Peña (TID), Félix J. García (UMU), Gregorio Martínez (UMU), David Fernández (UPM), Javier Sedano (UPM).

Abstract:

This document is divided in two different parts. The first part is related to Static VPNs with IPv6. We have analyzed the main IPsec/IKE implementations currently existing, differentiating between open-source and commercial solutions. After this analysis, we have designed several IPsec test-bed scenarios for IPv6 and evaluated the IPsec/IKE interoperability and the conformance of the analyzed solutions using these scenarios. The evaluation results have provided us with several considerations regarding the interoperability of IPv6 IPsec implementations. We have defined the VPN Broker service for IPv6 IXs and proposed a first approach to dynamic VPNs. The second part describes the existing IPv6 network security elements. Differences between filter and firewall features are discussed. Finally a generalised network model and its security, applicable in the Euro6IX project context, are described.

Keywords:

Dynamic VPN, Filter, Firewall, IKE, IPsec, Static VPN

Revision History

Revision	Date	Description	Author (Organization)
v1.0	01/03/2002	Document creation	Antonio Skarmeta (UMU) David Fernández (UPM)
v1.1	17/05/2002	Document updated	Antonio Skarmeta (UMU)
v1.2	30/05/2002	Changes	Antonio Skarmeta (UMU) Cesar Olvera (Consulintel)
v1.3	7/06/2002	Minor updates and clarifications.	Antonio Skarmeta (UMU)
v1.4	19/06/ 2002	Document updated to match Project Deliverable Template	Antonio Skarmeta (UMU)
v1.5	25/11/2002	Document updated	Félix Jesús García (UMU)
v1.6	05/12/2002	Document updated	Félix Jesús García (UMU) Cristina Peña (TID)
v1.7	11/12/2002	Document updated	Antonio Skarmeta (UMU) Félix Jesús García (UMU)
v1.8	13/12/2002	Document updated	Félix Jesús García (UMU) Cristina Peña (TID)
v1.9	16/12/2002	Document updated	Félix Jesús García (UMU) Javier Sedano Jarillo (UPM)
v1.10	17/12/2002	Document updated	Félix Jesús García (UMU)
v1.11	18/12/2002	Final changes	Antonio Skarmeta (UMU) Gregorio Martínez (UMU)
v1.12	8/01/2003	References list update and minor corrections	David Fernández (UPM)
v1.2	25/02/2003	Logos added an PDF generated	Jordi Palet (Consulintel)

Table of Contents

Table of Figures

1 STATIC VPNS WITH IPV6

1.1 Introduction

IPsec [1] provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services.

These objectives are met through the use of two traffic security protocols, the Authentication Header (AH) [2] and the Encapsulating Security Payload (ESP) [3], and through the use of cryptographic key management procedures and protocols. Moreover the support for both manual and automatic distribution of keys (IKE - [4], [5], [6]) is required.

IPsec was designed for interoperability. It should not affect networks and hosts that do not support it. IPsec is independent of the current cryptographic algorithms it can accommodate new ones as they become available. It works both with IPv4 and IPv6, but actually is a mandatory component of IPv6.

The IPsec protocols are designed so that different implementations should be able to work together. The evaluation of the interoperability between IPsec/IKE implementations is necessary to establish the scenarios that can be really developed, mainly for IPv6 networks.

We have analyzed the main IPsec/IKE implementations for IPv6 currently existing, and designed the basic scenarios where a host and/or a secure-gateway apply IPsec. We have evaluated the IPsec/IKE interoperability and the conformance of the analyzed solutions. We have considered the next steps to evaluate the solutions:

- Step-1. Interoperability with uni-implementation environment.
- Step-2. Interoperability with multi-implementation environment.

Using a test suite we have checked the features and limitations of the implementations using different scenarios.

1.2 IPsec/IKE roadmap in the IETF

1.2.1 IPsec/IKE protocol descriptions

1.2.1.1 IPsec

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services

These objectives are met through the use of two traffic security protocols, the **Authentication Header** (AH) [2] and the **Encapsulating Security Payload** (ESP) [3], and through the use of cryptographic key management procedures and protocols.

IPsec was designed for interoperability. It should not affect networks and hosts that do not support it. IPsec is independent of the current cryptographic algorithms it can accommodate new ones as they become available. It works both with IPv4 and IPv6, but actually is a mandatory component of IPv6.

Both AH and ESP are described in more detail in their respective RFCs.

- The IP Authentication Header (AH) provides connectionless integrity, data origin authentication, and an optional anti-replay service.
- The Encapsulating Security Payload (ESP) protocol may provide confidentiality (encryption), and limited traffic flow confidentiality. It also may provide connectionless integrity, data origin authentication, and an anti-replay service. (One or the other set of these security services must be applied whenever ESP is invoked.)
- Both AH and ESP are vehicles for access control, based on the distribution of cryptographic keys and the management of traffic flows relative to these security protocols.

These protocols may be applied alone or in combination with each other to provide a desired set of security services in IPv4 and IPv6. Each protocol supports two modes of use: transport mode and tunnel mode. In transport mode the protocols provide protection primarily for upper layer protocols; in tunnel mode, the protocols are applied to tunneled IP packets.

IPsec allows the user (or system administrator) to control the granularity at which a security service is offered. For example, one can create a single encrypted tunnel to carry all the traffic between two security gateways or a separate encrypted tunnel can be created for each TCP connection between each pair of hosts communicating across these gateways. IPsec management must incorporate facilities for specifying:

- which security services to use and in what combinations
- the granularity at which a given security protection should be applied
- the algorithms used to effect cryptographic-based security

Because these security services use shared secret values (cryptographic keys), **IPsec relies on a separate set of mechanisms for putting these keys in place**. (The keys are used for authentication/integrity and encryption services.) IPsec requires support for both manual and automatic distribution of keys. It specifies a specific public-key based approach (**IKE** -- [4], [5], [6]) for automatic key management, but other automated key distribution techniques may be used.

ESP provides encryption service to the packets that causes the data to be random in nature, rendering compression at lower protocol layers ineffective. IP payload compression (IPComp) [7] provides a way to compress packet before encryption by ESP. This protocol reduces the size of IP datagrams and will increase the overall communication performance.

1.2.1.2 IKE

Internet Key Exchange (IKE), is the protocol used to establish security associations that are needed by various services, for example IPsec uses IKE to establish the security associations (SA) needed to generate and refresh its keys. To establish security associations, keys need to be formed in a secure and protected manner and IKE provides the mechanism to achieve this.

IKE is a protocol which includes part of Oakley [5] and part of SKEME [8] as key exchange protocol, inside the ISAKMP [4] framework.

A **Security Association** (SA) is a simplex "connection" that affords security services to the traffic carried by it. Security services are afforded to an SA by the use of AH, or ESP, but not both. To secure typical, bi-directional communication between two hosts, or between two security gateways, two Security Associations (one in each direction) are required.

A security association is uniquely identified by a triple consisting of a Security Parameter Index (SPI) that identifies the SA, an IP Destination Address, and a security protocol (AH or ESP) identifier.

IKE is made up of two phases as defined in the ISAKMP framework, and within these phases Oakley defines a number of modes that can be used.

Phase 1 is the process where the ISAKMP security association must be established. It assumes that no secure channel currently exists and therefore it must initially establish one to protect any ISAKMP messages. This SA is different from other SAs that are negotiated for other services in that it is owned by ISAKMP.

Phase 2 is where subsequent security associations required by various services are negotiated on their behalf. The ISKMP SA generated in Phase 1 protects all subsequent ISAKMP messages.

Two modes are available for use in Phase 1, **main mode** and **aggressive mode**. Support for main mode is a mandatory requirement for IKE, while aggressive mode has the advantage of being able to use three rather than six messages flows to establish the ISAKMP SA.

Within phase 2, **quick mode** is used to negotiate the SAs for the services.

Informational mode is used to give the other party some information, normally abnormal conditions due to failures.

The other mode is **new group mode**, which is used to negotiate private groups for Diffie-Hellman [9] exchanges. Although protected by a phase 1 exchange, this is not part of a Phase 2 exchange.

The IKE mechanism is quite efficient in that it is able of negotiate many security associations with relatively few messages. With a single Phase 1 negotiation, multiple Phase 2 negotiations can occur. And within a single Phase 2 negotiation, multiple SAs can be negotiated.

Phase 1

During phase 1, the partners exchange proposals for the ISAKMP SA and agree on one. This contains specifications of authentication methods, hash functions and encryption algorithms to be used to protect the key exchanges. The partners then exchange information for generating a shared master secret:

- Cookies that also serve as SPIs for the ISAKMP SA
- Diffie-Hellman values
- Random Numbers
- Optionally exchange IDs when public key authentication is used.

Both parties then generate keying material and shared secrets before exchanging additional authentication information.

When all goes well, both parties derive the same keying material and actual encryption and authentication keys without ever sending any keys over the network. In addition to this, phase 1 also authenticates the two parties involved in the exchange.

There are four methods of authentication available:

- Digital signatures
- Public key encryption
- Revised public key encryption
- Pre-shared keys

During phase 1 only a single SA is negotiated, that is the ISAKMP SA. Only one proposal is offered always proposing Oakley as the key exchange method. Within that proposal multiple transforms can be offered which negotiate the following parameters:

- Authentication method
- Lifetime/lifesize of the SA
- Diffie-Hellman group
- Hash algorithm
- Encryption algorithm

In Figure 1-1 there is an example of phase 1 main mode. The content of the messages would depend on the authentication method used.

From now on, notation used in figures will be:

E() .- Is an encryption function.

H_i .- Hash i value.

SAmt/SAst .- SA multiple transforms/single transform.

RNi .- Random Number i.

IDi/IDr .- Identification of initiator/responder.

DHi/DHr .- Diffie-Hellman key exchange from initiator/responder.

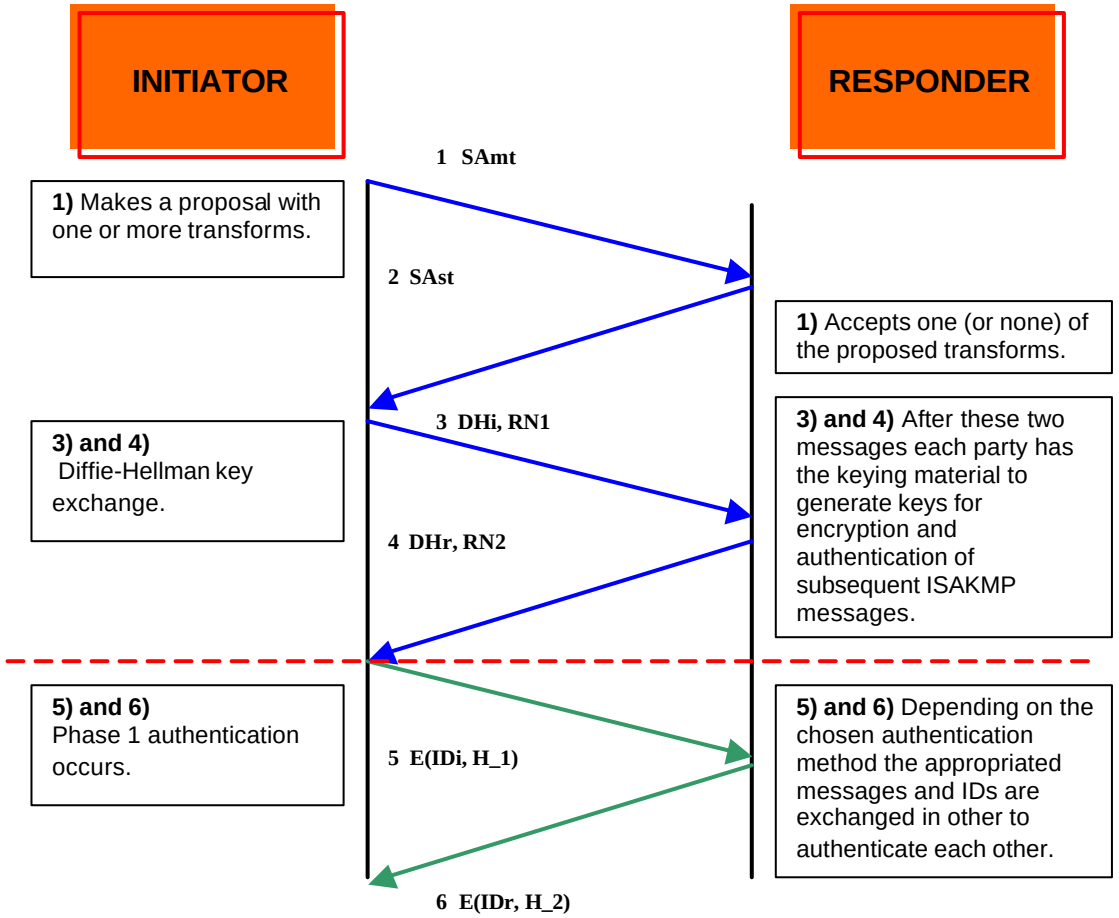


Figure 1-1: Phase 1: Main mode

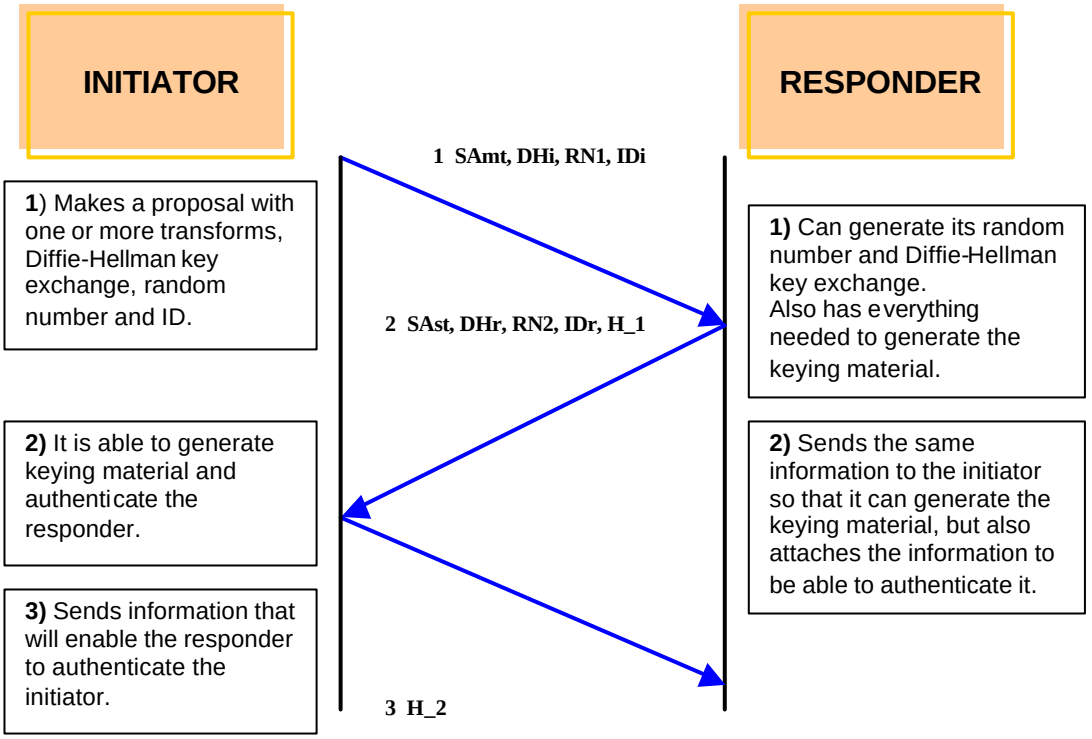


Figure 1-2: Phase 1: Aggressive mode

In **aggressive mode** only a total of three messages are needed to establish the SA. In Figure 1-2 there is an example of phase 1 aggressive mode.

As in the main mode, information that is exchanged to facilitate authentication is dependant on the negotiated authentication method.

Phase 2

During phase 2, the partners exchange proposals for protocol SAs and agree on one. This contains specifications of authentication methods, hash functions and encryption algorithms to be used to protect packets using AH and/or ESP. To generate keys, both parties use the keying material from a previous Phase 1 exchange and they can optionally perform an additional Diffie-Hellman exchange for PFS (Perfect Forward Security).

In **quick mode** there are basically three message flows:

For encryption, keying material obtained in phase 1 is used.

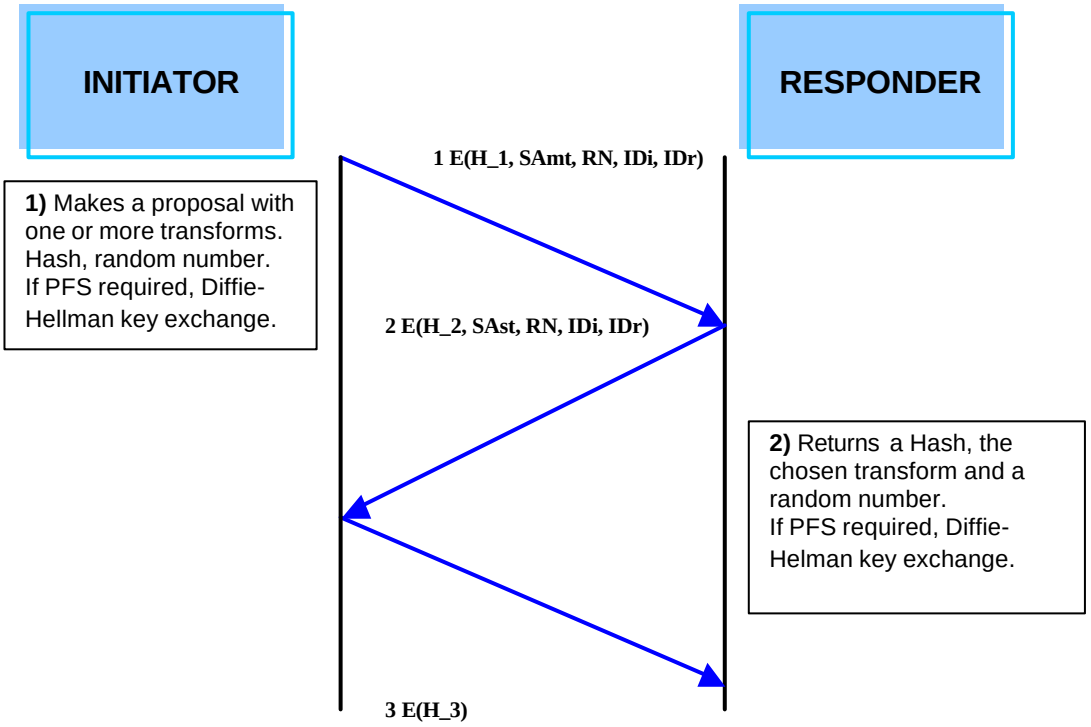


Figure 1-3: Phase 2: Quick mode

1.2.1.3 Interrelationship of IPsec/IKE documents

Some documents provide detailed definitions of some of the components of IPsec [10] and of their inter-relationship. They include RFCs on the following topics:

- "IP Security Document Roadmap" [11] -- a document providing guidelines for specifications describing encryption and authentication algorithms used in this system.
- Security protocols -- RFCs describing the Authentication Header (AH) [2] and Encapsulating Security Payload (ESP) [3] protocols.
- Algorithms for authentication and encryption -- a separate RFC for each algorithm.

?? Automatic key management -- RFCs on "The Internet Key Exchange (IKE)" [6], "Internet Security Association and Key Management Protocol (ISAKMP)" [4],"The OAKLEY Key Determination Protocol" [5], and "The Internet IP Security Domain of Interpretation for ISAKMP" [12].

More information about the IETF's IPsec Working Group: http://www.ietf.org/html.charters/ipsec-charter.html.

Below is a table of related RFCs.

Topic	URL	Title	Summary
General IPsec	RFC2401 November 1998 Obsoletes RFC1825, Updated by RFC3168	Security Architecture for the Internet Protocol	This memo specifies the base architecture for IPsec compliant systems
	RFC 2411 November 1998	IP Security Document Roadmap	The IPsec protocol suite is used to provide privacy and authentication services at the IP layer. Several documents are used to describe this protocol suite. The interrelationship and organization of the various documents covering the IPsec protocol are discussed here. An explanation of what to find in which document, and what to include in new Encryption Algorithm and Authentication Algorithm documents are described.
	RFC 2709 October 1999	Security Model with Tunnel-mode IPsec for NAT Domains	There are a variety of NAT flavours, as described in [13]. Of the domains supported by NATs, only Realm-Specific IP clients are able to pursue end-to-end IPsec secure sessions. However, all flavors of NAT are capable of offering tunnel-mode IPsec security to private domain hosts peering with nodes in external realm. This document describes a security model by which tunnel-mode IPsec security can be architected on NAT devices. A section is devoted to describing how security policies may be transparently communicated to IKE (for automated KEY exchange) during Quick Mode. Also outlined are applications that can benefit from the Security Model described.
AH and ESP Headers	RFC2402 November 1998 Obsoletes RFC1826	IP Authentication Header	The IP Authentication Header (AH) is used to provide connectionless integrity and data origin authentication for IP datagrams (hereafter referred to as just "authentication"), and to provide protection against replays.
	draft-ietf-ipsec-rfc2402bis-00.txt March 2002	IP Authentication Header	This document describes an updated version of the IP Authentication Header (AH), which is designed to provide authentication services in IPv4 and IPv6. This document is based upon RFC 2402 (November 1998). Section 7 provides a brief review of the differences between this document and RFC 2402.
	RFC2406 November 1998 Obsoletes RFC1827	IP Encapsulating Security Payload (ESP)	The Encapsulating Security Payload (ESP) header is designed to provide a mix of security services in IPv4 and IPv6
	draft-ietf-ipsec-esp-v3-02.txt March 2002	IP Encapsulating Security Payload (ESP)	This document describes an updated version of the Encapsulating Security Payload (ESP) protocol, which is designed to provide a mix of security services in IPv4 and Ipv6. ESP is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality. This document is based upon RFC 2406 (November 1998). Section 7 provides a brief review of the differences between this document and RFC 2406.
IKE	RFC 2407 November 1998	The Internet IP Security Domain of Interpretation for ISAKMP	The Internet Security Association and Key Management Protocol (ISAKMP) defines a framework for security association management and cryptographic key establishment for the Internet. This framework consists of defined exchanges, payloads, and processing guidelines that occur within a given Domain of Interpretation (DOI). This document defines the Internet IP Security DOI (IPSEC DOI), which instantiates ISAKMP for use with IP when IP uses ISAKMP to negotiate security associations.
	RFC 2408 November 1998	Internet Security Association and Key	This memo describes a protocol utilizing security concepts necessary for establishing Security Associations (SA) and cryptographic keys in an Internet

		Management Protocol (ISAKMP)	environment. A Security Association protocol that negotiates, establishes, modifies and deletes Security Associations and their attributes is required for an evolving Internet, where there will be numerous security mechanisms and several options for each security mechanism. The key management protocol must be robust in order to handle public key generation for the Internet community at large and private key requirements for those private networks with that requirement. The Internet Security Association and Key Management Protocol (ISAKMP) defines the procedures for authenticating a communicating peer, creation and management of Security Associations, key generation techniques, and threat mitigation (e.g. denial of service and replay attacks). All of these are necessary to establish and maintain secure communications (via IP Security Service or any other security protocol) in an Internet environment.
	RFC2409 November 1998	The Internet Key Exchange (IKE)	ISAKMP ([4]) provides a framework for authentication and key exchange but does not define them. ISAKMP is designed to be key exchange independent; that is, it is designed to support many different key exchanges. Oakley ([5]) describes a series of key exchanges—called "modes"-- and details the services provided by each (e.g. perfect forward secrecy for keys, identity protection, and authentication). SKEME ([8]) describes a versatile key exchange technique which provides anonymity, repudiability, and quick key refreshment. This document describes a protocol using part of Oakley and part of SKEME in conjunction with ISAKMP to obtain authenticated keying material for use with ISAKMP, and for other security associations such as AH and ESP for the IETF IPsec DOI.
	RFC 2412 November 1998	The OAKLEY Key Determination Protocol	This document describes a protocol, named OAKLEY, by which two authenticated parties can agree on secure and secret keying material. The basic mechanism is the Diffie-Hellman key exchange algorithm. The OAKLEY protocol supports Perfect Forward Secrecy, compatibility with the ISAKMP protocol for managing security associations, user-defined abstract group structures for use with the Diffie-Hellman algorithm, key updates, and incorporation of keys distributed via out-of-band mechanisms.
	RFC 2367 July 1998	PF_KEY Key Management API, Version 2	A generic key management API that can be used not only for IP Security [10] [2] [3] but also for other network security services is presented in this document. Version 1 of this API was implemented inside 4.4-Lite BSD as part of the U. S. Naval Research Laboratory's freely distributable and usable IPv6 and IPsec Implementation [14]. It is documented here for the benefit of others who might also adopt and use the API, thus providing increased portability of key management applications (e.g. a manual keying application, an ISAKMP daemon, a GKMP daemon [15][16], a Photuris daemon, or a SKIP certificate discovery protocol daemon).
Crypto Algorithms	RFC 2405 November 1998	The ESP DES-CBC Cipher Algorithm With Explicit IV	This document describes the use of the DES Cipher algorithm in Cipher Block Chaining Mode, with an explicit IV, as a confidentiality mechanism within the context of the IPsec Encapsulating Security Payload (ESP).
	RFC 2451 November 1998	The ESP CBC-Mode Cipher Algorithms	This document describes how to use CBC-mode cipher algorithms with the IPsec ESP (Encapsulating Security Payload) Protocol. It not only clearly states how to use certain cipher algorithms, but also how to use all CBC-mode cipher algorithms.
	RFC 2104 February 1997	HMAC: Keyed-Hashing for Message Authentication	This document describes HMAC, a mechanism for message authentication using cryptographic hash functions. HMAC can be used with any iterative cryptographic hash function, e.g., MD5, SHA-1, in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.
	RFC 2202 September 1997	Test Cases for HMAC-MD5 and HMAC-SHA-1	This document provides two sets of test cases for HMAC-MD5 and HMAC-SHA-1, respectively. HMAC-MD5 and HMAC-SHA-1 are two constructs of the HMAC [17] message authentication function using the MD5 [18] hash function and the SHA-1 [19] hash function. Both constructs are used by IPSEC [20] and other protocols to authenticate messages. The test cases and results provided in this document are meant to be used as a conformance test for HMAC-MD5 and HMAC-SHA-1 implementations.
	RFC 2403 November 1998	The Use of HMAC-MD5-96 within ESP and AH	This memo describes the use of the HMAC algorithm [17] in conjunction with the MD5 algorithm [18] as an authentication mechanism within the revised IPSEC Encapsulating Security Payload [3] and the revised IPSEC Authentication Header [2]. HMAC with MD5 provides data origin authentication and integrity protection.

	RFC 2404 November 1998	The Use of HMAC-SHA-1-96 within ESP and AH	This memo describes the use of the HMAC algorithm [17] in conjunction with the SHA-1 algorithm [21] as an authentication mechanism within the revised IPSEC Encapsulating Security Payload [3] and the revised IPSEC Authentication Header [2]. HMAC with SHA-1 provides data origin-authentication and integrity protection.
	RFC 2857 June 2000	The Use of HMAC-RIPEMD-160-96 within ESP and AH	This memo describes the use of the HMAC algorithm [17] in conjunction with the RIPEMD-160 algorithm [22] as an authentication mechanism within the revised IPSEC Encapsulating Security Payload [3] and the revised IPSEC Authentication Header [2]. HMAC with RIPEMD-160 provides data origin authentication and integrity protection
	RFC 2410 November 1998	The NULL Encryption Algorithm and Its Use With IPsec	This memo defines the NULL encryption algorithm and its use with the IPsec Encapsulating Security Payload (ESP). NULL does nothing to alter plaintext data. In fact, NULL, by itself, does nothing. NULL provides the means for ESP to provide authentication and integrity without confidentiality. Further information on the other components necessary for ESP implementations is provided by [3] and [11].
	RFC 1828 August 1995	IP Authentication using Keyed MD5	This document describes the use of keyed MD5 with the IP Authentication Header.
	RFC 1829 August 1995	The ESP DES-CBC Transform	This document describes the DES-CBC security transform for the IP Encapsulating Security Payload (ESP).
	RFC 2085 February 1997	HMAC-MD5 IP Authentication with Replay Prevention	This document describes a keyed-MD5 transform to be used in conjunction with the IP Authentication Header [2]. The particular transform is based on [23]. An option is also specified to guard against replay attacks.
	RFC 3173 September 2001	IP Payload Compression Protocol (IPComp)	This document describes a protocol intended to provide lossless compression for Internet Protocol datagrams in an Internet environment.
	RFC 2394 December 1998	IP Payload Compression Using DEFLATE	This document describes a compression method based on the DEFLATE compression algorithm. This document defines the application of the DEFLATE algorithm to the IP Payload Compression Protocol.
	RFC 2395 December 1998	IP Payload Compression Using LZS	This document describes a compression method based on the LZS compression algorithm. This document defines the application of the LZS algorithm to the IP Payload Compression Protocol [7]. [7] defines a method for applying lossless compression to the payloads of Internet Protocol datagrams.
	RFC 3051 January 2001	IP Payload Compression Using ITU-T V.44 Packet Method	This document describes a compression method based on the data compression algorithm described in International Telecommunication Union (ITU-T) Recommendation V.44. Recommendation V.44 is a modem standard but Annex B, Clause B.1, of the recommendation describes the implementation of V.44 in packet networks (e.g., V.44 Packet Method). This document defines the application of V.44 Packet Method to the Internet Protocol (IP) Payload Compression Protocol (RFC 2393). RFC 2393 defines a method for applying lossless compression to the payload portion of IP datagrams. V.44 Packet Method is based upon the LZJH data compression algorithm. Throughout the remainder of this document the terms V.44 Packet Method and LZJH are synonymous.

Figure 1-4: IPsec RFCs

1.3 IPv6 IPsec/IKE solutions to analyze

In order to do this analysis, we have made two different groups, the first one with the open-source solutions and the second one with commercial solutions. The selected open-source solutions to analyze are:

- FreeS/WAN with support IPv6
- USAGI Project
- KAME Project

And the commercial ones are:

- Windows XP

- Solaris 9
- 6WIND IP Edge Device

6WIND IP Edge Device is the only solution provided by a router manufacturer, 6WIND. CISCO routers will be used to implement the test scenarios, but they will be only IPv6 routers and they will not be secure gateways because currently CISCO does not support IPsec/IKE with IPv6.

Next sections describe the features and limitations of the IPsec selected implementations according to their web pages and public documentation. As soon as we test the implementations, we will check these features and limitations in a real environment and we will establish the interoperability level among different IPsec solutions.

1.3.1 Open-Source Solutions

Security for many users is a very sensitive issue. Sometimes it is so sensitive, that they will not or even are not allowed to trust any commercial products simply because it is not possible to evaluate their functionality.

A number of projects are working on IPv6 implementation. A prominent Open Source effort is KAME (www.kame.net), is a collaboration of several companies in Japan to provide a free IPv6 and IPsec stack for BSD variants to the world. Other major players are also working on IPv6, they are IABG (www.ipv6.iabg.de) with software FreeS/WAN (www.freeswan.org) and USAGI Project (www.linux-ipv6.org) based on IABG and FreeS/WAN.

1.3.1.1 FreeS/WAN Project

FreeS/WAN offers an open source implementation of the IPsec protocol suite running on the Linux operating system. IABG is currently working on IPsec services based on the software FreeS/WAN for Linux. IABG is closely working with the FreeS/WAN team to integrate the necessary extensions for IPv6 as part of the IST project 6WINIT (www.6winit.org).

Architecture of FreeS/WAN

The architecture of FreeS/WAN consists of two major parts, the IKE daemon (called Pluto) with a set of utilities and the Kernel IPsec Support (called KLIPS, with virtual IPsec devices), as shown in the figure below (current design and functionality of FreeS/WAN).

Figure 1-5: FreeSWAN architecture

User Space

FreeS/WANs user space programs and utilities are:

- Pluto, the IKE daemon.
- A script machinery with the shell script IPsec as user front-end.
- Some utilities for set-up and configuration.

Pluto is responsible for negotiating ISAKMP and IPsec SAs with other IKE daemons (according to its SPD) and to install these IPsec SAs in KLIPS.

Kernel Space

The kernel part of FreeS/WAN contains the virtual IPsec devices, the SAD and the crypto machinery. An IPsec device is a virtual interface that can be established between the IP layer and a physical network device (e.g. an Ethernet device). With this design FreeS/WAN is able to force traffic that should make use of IPsec to go through the KLIPS machinery without changing too much in the kernels networking code.

Outgoing packet processing

A route lookup inside the IP layer returns an IPsec device as outgoing interface, thus the IP networking code calls the output routine of KLIPS. Now the Destination IP address and the virtual IPsec device are used to select the appropriate IPsec SA to be applied. Then the output routine of the attached physical network device is called.

Incoming packet processing

An incoming packet is received by a physical device, and is passed to KLIPS machinery only if it has an ESP or AH header, where it is assigned to the attaching IPsec device and processed according to the appropriate SAs. Then it is passed up the stack to the next protocol handler.

Available functionality

IABG's IPsec implementation is based on the work done by FreeSWAN. Currently not all the functionality, which is supported for IPv4, is supported for IPv6. This paragraph describes the status of the IPv6 aspects of this IPsec implementation.

Currently it is the release 1.91 of Linux FreeS/WAN with IPv6 support. It's a prototype implementation of IPsec for IPv6. Currently not everything is working perfectly, the following features have been tested successfully by IABG according to IABG's web page:

- IPSEC transforms: AH and ESP in both transport and tunnel mode.
- Encryption algorithms: Triple-DES.
- Authentication algorithms: HMAC-MD5 and HMAC-SHA-1.
- Key negotiation: IKE Main Mode with preshared keys.

The RFCs define two modes for IKE negotiations, main mode and aggressive mode. FreeS/WAN does not implement aggressive mode, so any negotiation another implementation tries that way will fail. This should not be a problem since main mode support is required in all implementations and aggressive mode is optional.

FreeS/WAN does not implement single DES because DES is insecure. FreeS/WAN does not implement Diffie-Hellman group 1 because it is not entirely clear that this is secure. And not support IP compression.

1.3.1.2 USAGI Project

USAGI (UniverSAl playGround for Ipv6) Project works to deliver the production quality IPv6 protocol stack for the Linux system. The IPsec/IKE solution of the USAGI Project is based on FreeS/WAN-1.9 and IABG. There are differences between them in the architecture and the management and configuration, but these differences are not notables.

The last release is the USAGI STABLE RELEASE 4. Currently not everything is working perfectly, the following features are supported according to USAGI's web page (http://www.linux-ipv6.org):

- Key negotiation: USAGI kit does not include IKE daemon, only manual key exchange.
- IPSEC transforms: AH and ESP in transport mode.
- Encryption algorithms: DES, Triple-DES and Rijndael/AES.
- Authentication algorithms: HMAC-MD5 and HMAC-SHA-1.

1.3.1.3 KAME Project

KAME Project (www.kame.net) is the most prominent Open Source effort for IPsec/IKE. Researchers from several Japanese companies joined the project.

KAME Project aims to provide free reference implementations of

- IPv6
- IPsec (for both IPv4 and IPv6)
- advanced internetworking such as advanced packet queuing, ATM, mobility, and whatever interesting

on BSD variants.

Architecture of KAME for IPsec/IKE

KAME kernel maintains two databases to use IPsec. One is the Security Policy Database (SPD). Kernel refers to SPD in order to decide whether to apply IPsec to a packet or not. Also SPD entries specify which/how IPsec SA is applied. Another one is the Security Association Database (SAD). SAD entries contain key of each IPsec SA.

The following figure specifies a flow until kernel applies IPsec SA to a packet.

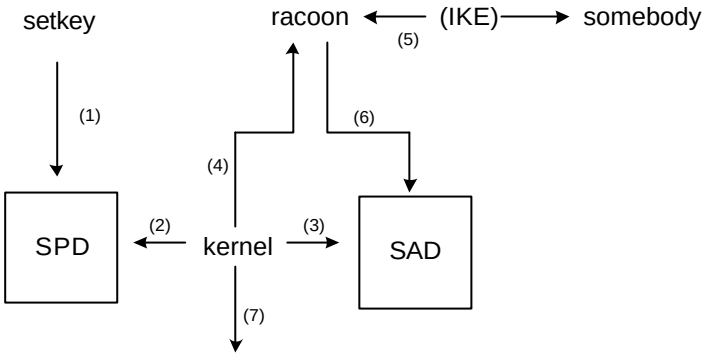


Figure 1-6: Flow of kernel

(1) The administrator sets a policy to SPD by using *setkey* command.
(2) Kernel refers to SPD in order to make a decision of applying IPsec to a packet.
(3) If IPsec is required, then kernel gets the key for IPsec SA from SAD.
(4) If it is failed, then kernel sends a request to get the key to racoon (IKE daemon).
(5) racoon exchanges the key by using IKE with the other extreme to be established IPsec SA.
(6) racoon puts the key into SAD.
(7) Kernel can send a packet applied IPsec.

So that the administrator must configure SPD entries by using *setkey* command, and must configure racoon. Also it must be required to run racoon or other IKE daemon on the other extreme.

Available functionality

KAME Project has developed an IPsec/IKE solution with a wide functionality compliant with the corresponding RFCs.

Different configurations are possible:

- IPsec with static keys (manual key exchange)
- IPsec with IKE pre-shared keys
- IPsec with IKE certificates

The involved IPsec algorithms are the following ones:

- Encryption algorithms: DES, Triple-DES, Blowfish, Cast128, Rijndael/AES, Twofish.
- Authentication algorithms: HMAC-MD5 and HMAC-SHA-1.

The KAME solution supports the tunnel and transport operation modes: AH and ESP can be used separately; and both AH and ESP can be applied together to an IP datagram.

1.3.2 Commercial Solutions

The documentation of the commercial IPsec/IKE solutions is very limited in comparison with the open-source solutions. We show below a summary of the features of three commercial solutions, Microsoft XP, Solaris 9 and 6WIND IP Edge Device.

1.3.2.1 Microsoft XP

To the Microsoft XP System Operative, IPsec for IPv6 is supported with the following limitations:

- The Authentication Header (AH) and Encapsulating Security Payload (ESP) are supported for both transport and tunnel modes. However, ESP for the IPv6 Protocol for Windows XP does not support data encryption.
- IPsec in the IPv6 Protocol for Windows XP does not support the use of Internet Key Exchange (IKE) to negotiate security associations (SAs). IPsec policies, SAs, and the keys to calculate the Message Digest 5 (MD5) keyed hash for AH or ESP must be manually configured.
- IPsec for IPv6 traffic is completely independent from IPsec for IPv4 traffic. IPv6 IPsec security policies are not managed with the Windows XP IPsec Policies snap–in. IPsec policies and SAs for the IPv6 Protocol for Windows XP are manually configured with the Ipsec6.exe command–line tool.

1.3.2.2 Solaris 9

To the Solaris 9 System Operative, according to Sun's web page (http://www.sun.com), IPsec for IPv6 is supported with the following limitations:

- IPsec in the IPv6 Protocol for Solaris 9 does not support the use of Internet Key Exchange (IKE) to negotiate security associations (SAs). Currently, SAs on IPv4 packets can take advantage of automatic key management, while SAs on IPv6 packets require manual management.
- Authentication algorithms include HMAC-MD5 and HMAC-SHA-1.
- Encryption algorithms include DES, Triple-DES (3DES), Blowfish and Rijndael/AES.

1.3.2.3 6WIND IP Edge Device

The IP Edge Device provides for setting up secured Virtual Private Networks on a non-secured architecture by establishing tunnels between remote sites thus guaranteeing authentication, data integrity and data confidentiality between these sites. Thus, 6WIND Edge Device is acting as an IPsec Gateway.

Architecture of 6WIND IP Edge Device

The security mechanisms take place at two different levels. The first one concerns the algorithms used for data encryption in order to ensure that the data have not been altered (data integrity) and that unauthorised users are not able to read them (data confidentiality). The second one defines how keys used by the algorithms are available at the user side. The key exchange has to be itself secure enough to avoid key disclosure.

The below figure depicts the architecture selected for the IP Edge Device.

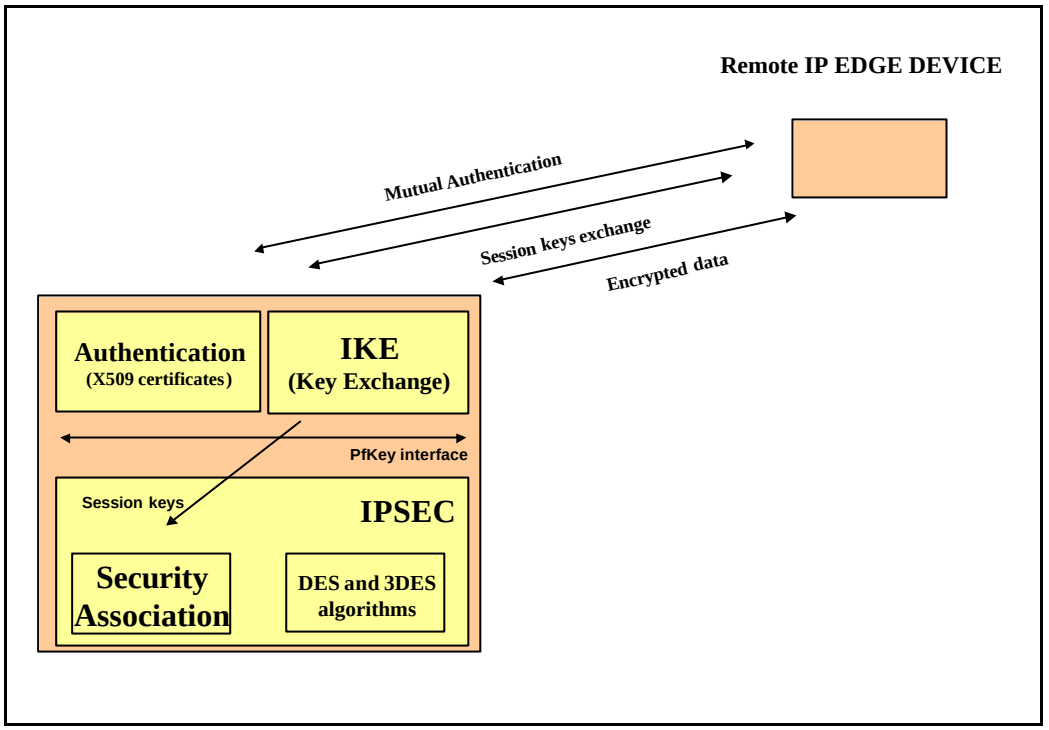


Figure 1-7: 6WIND IP Edge Device architecture

The 6WIND IP Edge Device implements IPsec mechanisms compliant with the latest RFC. The key management can be manual or based on the IKE key exchange management protocol.

The DES and 3DES algorithms are symmetric and reversible and they need session keys to operate. The keys have to remain secret, so a mechanism has to be implemented to guarantee the non-disclosure of these keys. The keys are exchanged during the authentication phase covered by an algorithm using asymmetrical keys. The algorithm implemented for that purpose in the IP Edge Device is the RSA with 1024 bit keys. The process ensures a strong and mutual authentication without any directory distribution. The keys are generated thanks to a private key manager able to generate X.509 certificates according to the Public Key Infrastructure (PKI) defined by the user. A Certification Authority delivers trusted certificates.

The IP Edge Device can also make use of pre-shared keys but this method is less practical as the certificate-based one. As a matter of fact, inserting a new device in a network implies to program each device with a key shared by the peers.

Available functionality

The 6WIND IP Edge Device implements IPsec and IKE functions compliant with the corresponding RFCs.

Different configurations are possible:

- IPsec with static keys
- IPsec with IKE pre-shared keys
- IPsec with IKE certificates

The involved IPsec algorithms are the following ones:

- Encryption algorithms: DES and Triple-DES.
- Authentication algorithms: HMAC-MD5 and HMAC-SHA-1.

The IP Edge Device supports the tunnel operation mode: AH and ESP can be used separately; and both AH and ESP can be applied together to an IP datagram.

6WIND has defined several ways to manage IP Edge Devices. The first way to configure an IP Edge Device is to use the Command Line Interface developed by 6WIND. All the device functions can be programmed through this interface. The second way is a management tool called Network Management System (NMS). It is based on the SNMP protocol and on a standard management platform with a customised GUI. The entire security configuration can be done via this graphical tool.

1.3.3 Features and limitations

This paragraph resumes the features of the IPsec selected implementations according to their web pages and public documentation. Later we will test the implementations and we will check these features and limitations and then we will establish the level of interoperability between the IPsec solutions.

Regarding the operation modes, all the implementations support both modes except three solutions; currently USAGI and Solaris do not implement the tunnel mode and obviously 6WIND do not implement the transport mode.

Operation Mode	*FreeS/WAN*	*USAGI*	*KAME*	*Windows*	*Solaris*	*6WIND*
Transport	*support*	*support*	*support*	*support*	*support*	
Tunnel	*support*	*future*	*support*	*support*		*support*

Table 1. Operation mode supported

Except KAME solution, all the implementations do not support IP Payload Compression Protocol (IPComp). This affirmation is according to the public documentation of the implementations.

Compression Algorithms	*FreeS/WAN*	*USAGI*	*KAME*	*Windows*	*Solaris*	*6WIND*
Deflate	*future*		*support*			
LZS			*support*			

Table 2. Compression algorithms supported

In regards to the authentication algorithms, all the solutions support HMAC-MD5 and HMAC-SHA1 algorithms but Windows.

Authentication Algorithms	*FreeS/WAN*	*USAGI*	*KAME*	*Windows*	*Solaris*	*6WIND*
HMAC-MD5	*support*	*support*	*support*	*support*	*support*	*support*
HMAC-SHA-1	*support*	*support*	*support*		*support*	*support*

Table 3. Authentication algorithms supported

The number of encryption algorithms support to the implementations, except the Japanese solutions (KAME and USAGI), is very limited. The Tripe-DES algorithm is the most used encryption algorithm.

Encryption Algorithms	FreeS/WAN	USAGI	KAME	Windows	Solaris	6WIND
DES-CBC		*support*	*support*		*support*	*support*
Triple DES	*support*	*support*	*support*		*support*	*support*
Rijndael/AES		*support*	*support*		*support*	
Cast128			*support*			
Twofish			*support*			
Blowfish			*support*		*support*	

Table 4. Encryption algorithms supported

In relation to the negotiation methods, we establish four possible configurations, one does not use IKE (static keys manually keyed) and the rest uses IKE (preshared keys, certificates and secure DNS). Only both KAME and 6WIND support certificates, FreeS/WAN needs a specific patch. In the next future, FreeS/Wan will support Secure DNS.

Configurations	FreeS/WAN	USAGI	KAME	Windows	Solaris	6WIND
Static keys		*support*	*support*	*support*	*support*	*support*
Preshared keys	*support*	*future*	*support*	*future*	*future*	*support*
Certificates	*just released*		*support*			*support*
Secure DNS	*future*					

Table 5. Configurations supported

1.4 Designed evaluation plan

Our objective is to evaluate the IPsec/IKE interoperability and the conformance of all proposal solutions. We consider the below steps to evaluate the solutions:

- Step-1. Interoperability with uni-implementation environment. This step will show if nodes with the same implementation are compatible between them. We will check the features and limitations of the implementation using different scenarios.
- Step-2. Interoperability with multi-implementation environment. This is the same than verifying interoperability with "the real world". This step will show if nodes with a different implementation are compatible between them. We will know the interoperability problems between implementations using the same scenarios than the Step-1.

These evaluations will generate reports of two types:

- Configuration and installations guides.
- Test reports.

This document will present the most important considerations of all reports generated for the evaluation.

1.4.1 Background

The IPsec protocols are designed so that different implementations should be able to work together. IPsec has a lot of details, but considerable success has been achieved. The evaluation of the interoperability between several IPsec/IKE implementations is not new. The main effort is being done by TAHI Project (http://www.tahi.org), this project has published several test reports about the implementation of KAME and the solution of USAGI for Linux. Currently, the TAHI project has not published other test reports with other implementations such as FreeS/WAN, Windows XP, Solaris 9 or 6WIND. Moreover the TAHI objective is not the same than our one. The TAHI objective is to develop and provide the verification technology for IPv6, included IPsec/IKE. On the other hand our objective is to deploy a static VPN service, so our interoperability tests will look for the response for the question: in one particular scenario, which combinations of IPv6 IPsec implementations are possible and which one is the most suitable?

Other player is VPNC (vpnc.org) who provides IPsec conformance tests. Conformance means that the product was tested against two different servers, and it passed the test on each server. The two systems used to test for conformance are OpenBSD and KAME. Thus, if an IPsec solution passes the test, it indicates that the product interoperates with the servers against which they were tested, but not necessarily with other products that passed the same test. Our objective is more ambitious in the sense that we want to design a multi-implementation environment.

1.4.2 Test Scenarios

We find three basic configurations to define all IPsec scenarios.

?? No secure gateway applies IPsec, only hosts do. When we need secure communications between two hosts across an insecure medium. This below figure shows a schema of this configuration.

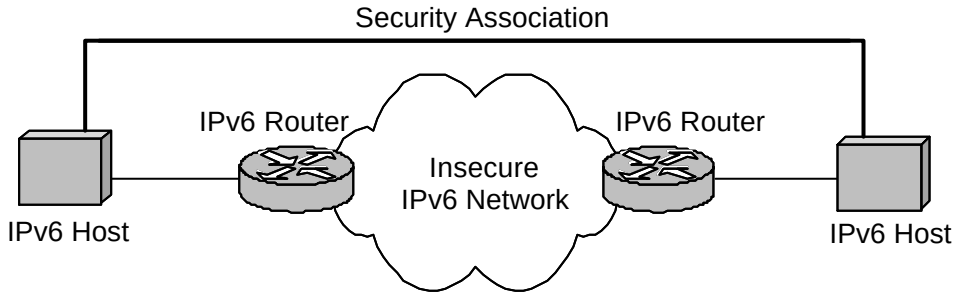


Figure 1-8: Host to Host

?? No host applies IPsec, only secure gateways do. When we need a virtual private network (VPN) between two networks. This below figure shows a schema of this configuration.

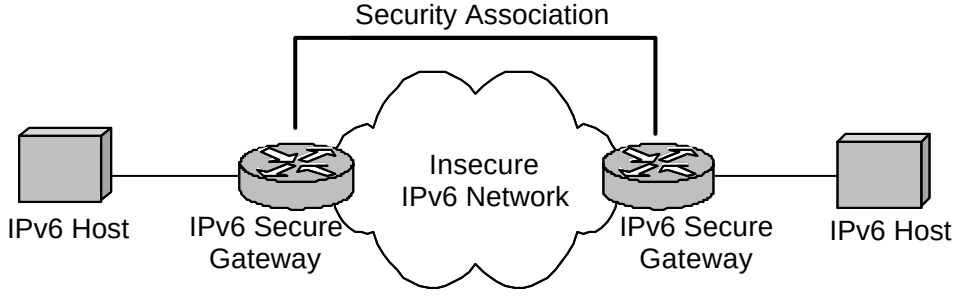


Figure 1-9: Secure Gateway to Secure Gateway

?? Both host and secure gateway apply IPsec. When we need a host can establish a secure communication with a network. This below figure shows a schema of this configuration.

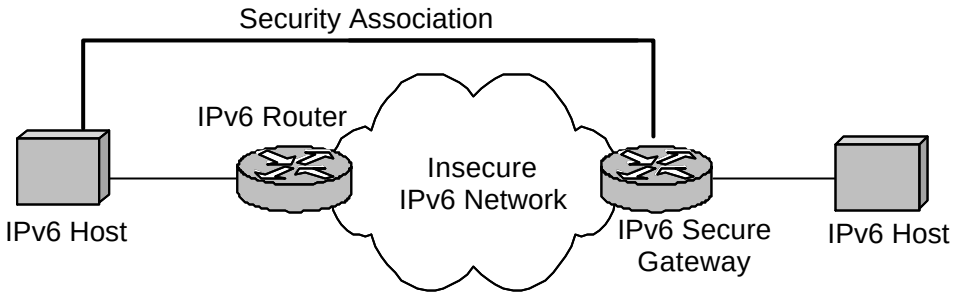


Figure 1-10: Host to Secure Gateway

Each configuration has several possible scenarios and these can combine to generate a new scenario. This paragraph shows the main scenarios.

1.4.2.1 No secure gateway applies IPsec, only hosts do

These scenarios are defined when two hosts across an insecure medium establish a secure communication using IPsec. There are several scenarios depending on the used mode, AH or ESP, and the number of established SAs. When we establish one SA between hosts, we can get four basic scenarios:

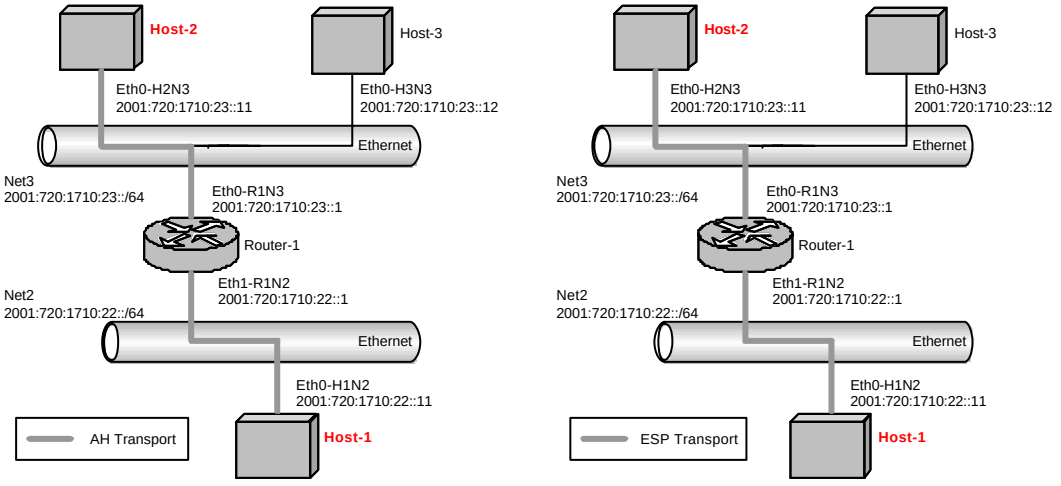
?? Transport mode AH: two hosts get a SA using transport mode AH.

?? Transport mode ESP: two hosts get a SA using transport mode ESP.

?? Tunnel mode AH: two hosts get a SA using tunnel mode ESP.

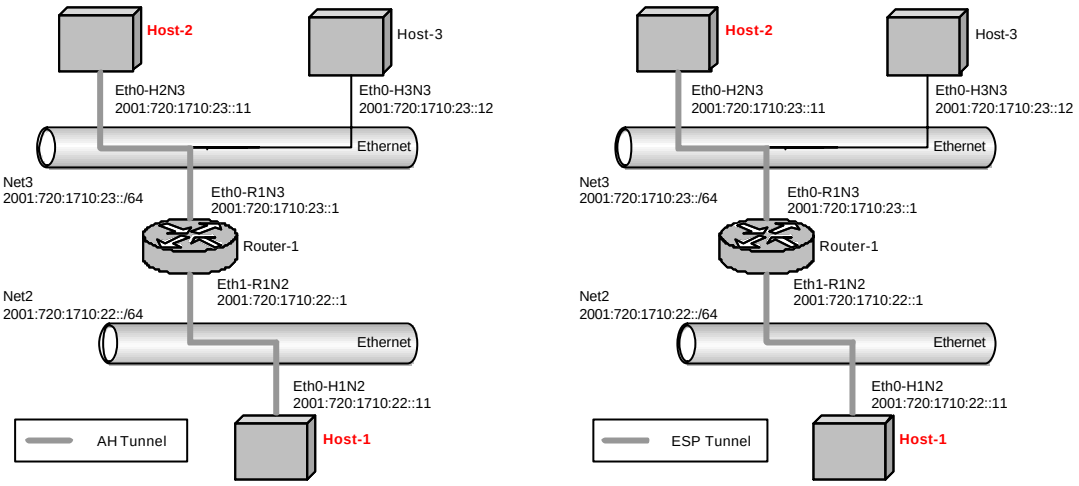
?? Tunnel mode ESP: two hosts get a SA using tunnel mode ESP.

Each scenario is showed on below figure. In the figure, Host-2 and Host-1 are the hosts establish the SA in each scenario.



a) Tranport mode AH b) Tranport mode ESP

Figure 1-11: Only hosts: a) Transport mode AH, b) Transport mode ESP



a) Tunnel mode AH b) Tunnel mode ESP

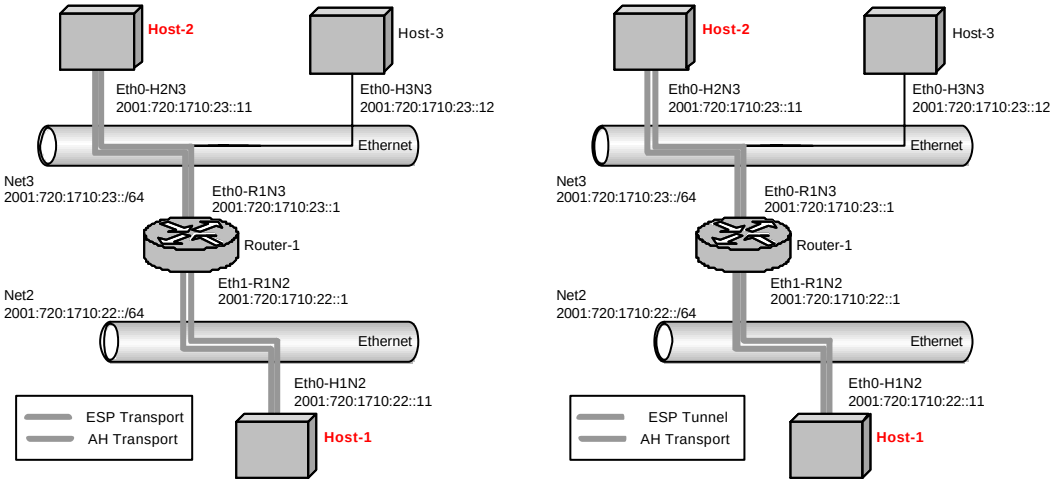
Figure 1-12: Only hosts: a) Tunnel mode AH, b) Tunnel mode ESP

If we establish two SAs, both endpoints for the SAs are the same. The inner and outer tunnels could each be either AH or ESP, though it is unlikely that host would specify both to be the same, i.e., AH inside of AH or ESP inside of ESP. So we define two basic scenarios:

?? Transport mode AH and ESP: two hosts get a SA using transport mode ESP and after get a new SA using transport mode AH.

?? Transport mode AH and tunnel mode ESP: two hosts get a SA using tunnel mode ESP and after get a new SA using transport mode AH.

Each scenario is showed on below figure. In the figure, Host-2 and Host-1 are the hosts establish the SA in each scenario.



a) Transport mode AH and ESP b) Transport mode AH and tunnel mode ESP

Figure 1-13: Only hosts: a) Transport mode AH and ESP,

b) Transport mode AH and tunnel mode ESP

The rest of scenarios between hosts you can establish with these basic scenarios.

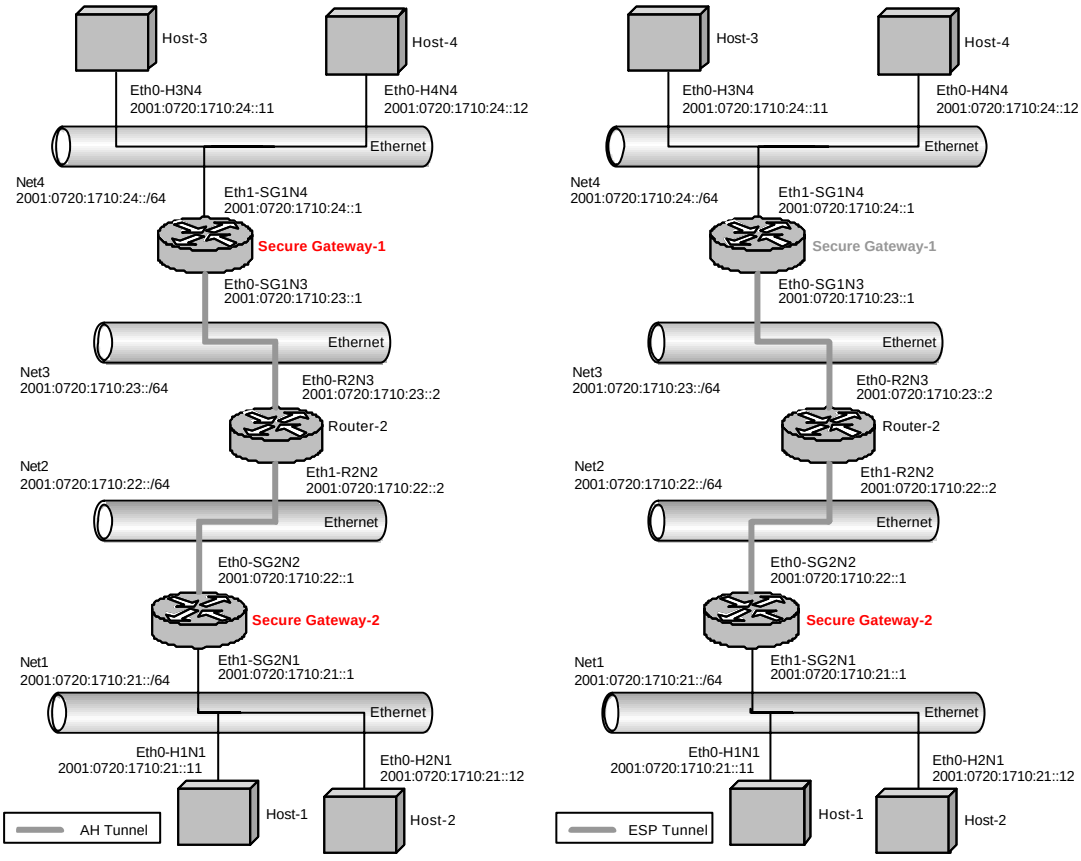
1.4.2.2 No host applies IPsec, only secure gateways do

The purpose of these scenarios is to test situations in which a virtual private network (VPN) is created between two local networks. When we establish one SA between two secure gateways, we can get two basic scenarios:

- Tunnel mode AH: two secure gateways get a SA using tunnel mode AH.
- Tunnel mode ESP: two secure gateways get a SA using tunnel mode ESP.

The hosts do not enter to establish the SA between the secure gateways, but one of them can be which cause the creation of the SA when it needs a secure communication with a host of the other network.

Each scenario is showed on below figure. In the figure, Secure Gateway-1 and Secure Gateway-2 are the secure gateways establish the SA in each scenario.



a) Tunnel mode AH

b) Tunnel mode ESP

Figure 1-14: Only SGs: a) Tunnel mode AH, b) Tunnel mode ESP

If we establish two SAs between two secure gateways, we can get one basic scenario, this is below:

- AH Transport to ESP Tunnel mode: two secure gateways get a SA using tunnel mode ESP and after get a new SA using transport mode AH.

This scenario is showed on below figure. In the figure, Secure Gateway-1 and Secure Gateway-2 are the secure gateways establish the SA.

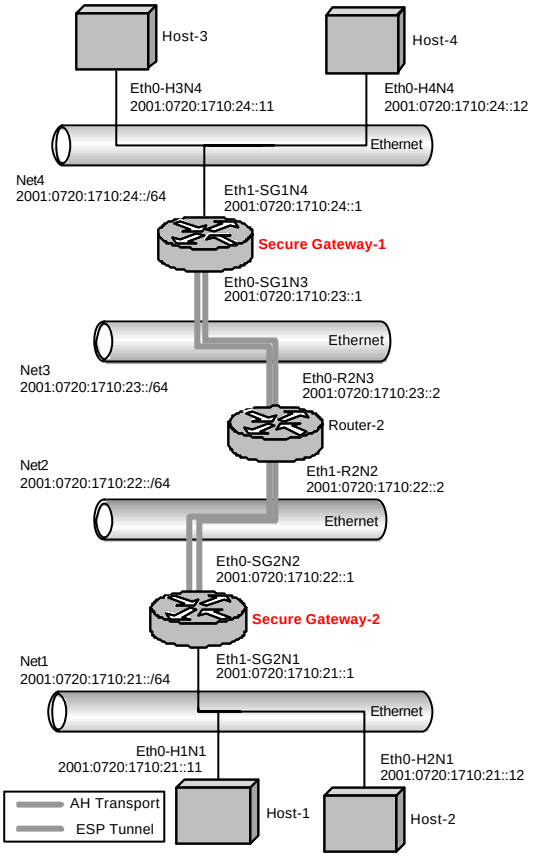


Figure 1-15: Only SGs: AH Transport to ESP Tunnel mode

Other possibility is when we have a sub-network into one of the networks and we need a secure communications from our network to this sub-network, we can build the below scenario:

- IPsec Tunnel to IPsec Tunnel mode: two secure gateways get a SA using tunnel mode ESP/AH and after one of them get a new SA using tunnel mode ESP/AH with an inner secure gateway of the other network.

This scenario is showed on below figure. In the figure, Secure Gateway-1 and Secure Gateway-2 are the secure gateways establish the first SA and Secure Gateway-3 is the secure gateway get the second SA with Secure Gateway-1.

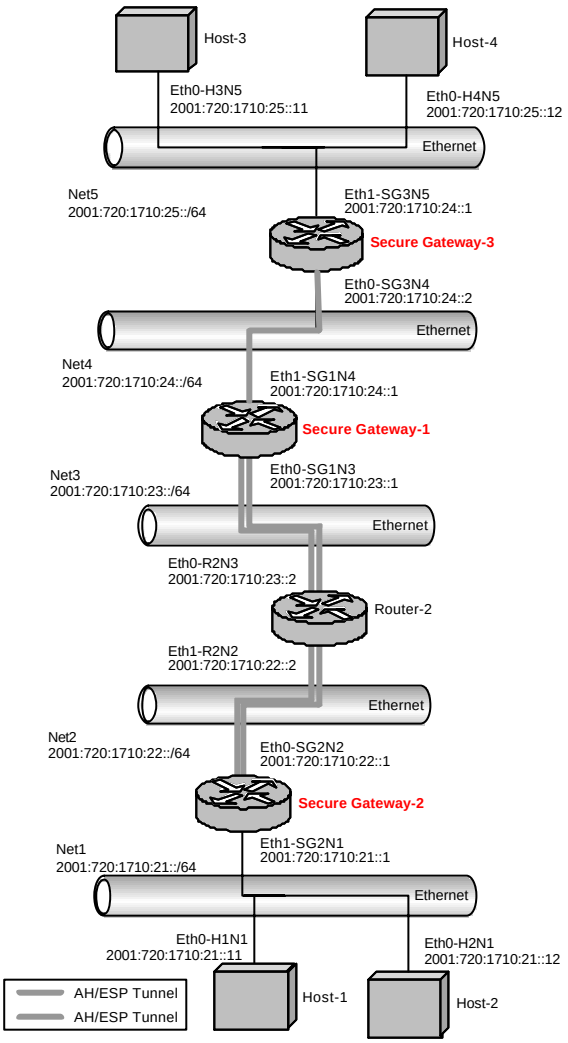


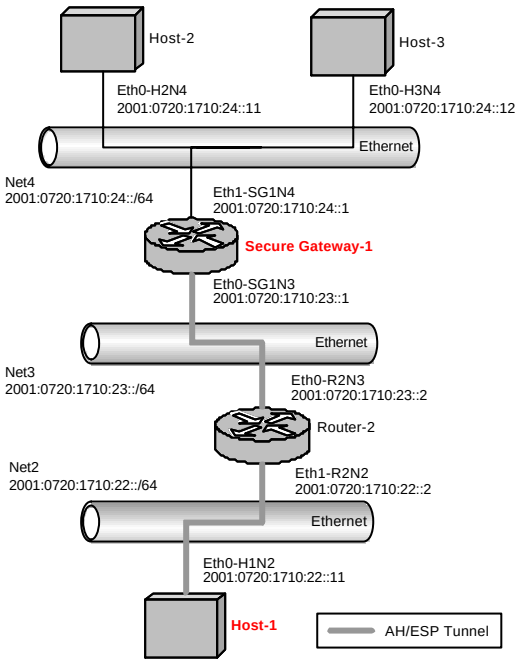
Figure 1-16: Only SGs: IPsec Tunnel to IPsec Tunnel mode

1.4.2.3 Both host and secure gateway apply IPsec

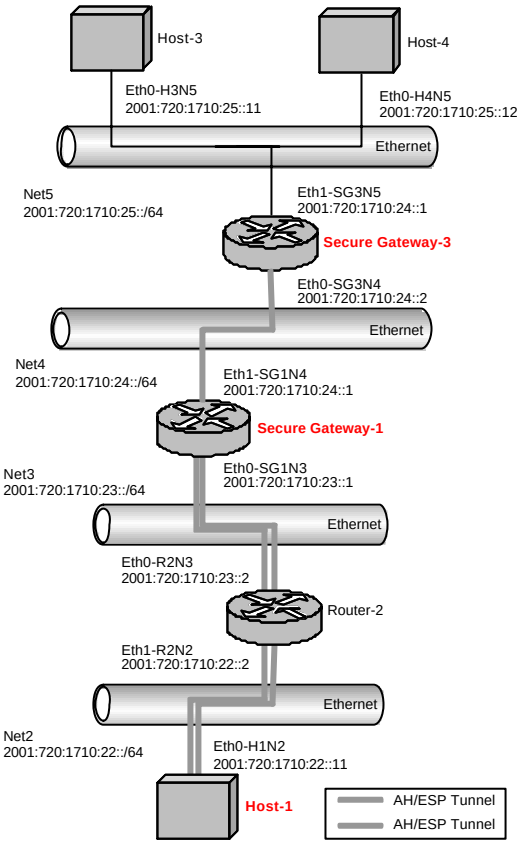
The purpose of these scenarios is to test situations in which a virtual private network (VPN) is created between a secure gateway and a host. These are the scenarios known as the “Road Warrior”. We can find three basic scenarios:

- IPsec Tunnel mode with host: one host and one secure gateway get a SA using tunnel mode AH/ESP.
- Transport mode and tunnel mode: one host and one secure gateway get a SA using tunnel mode AH/ESP and after the host get a new SA using transport mode AH/ESP with an inner host of the network.
- IPsec Tunnel to IPsec Tunnel mode with host: one host and one secure gateway get a SA using tunnel mode AH/ESP and after the host get a new SA using tunnel mode AH/ESP with an inner gateway of the network.

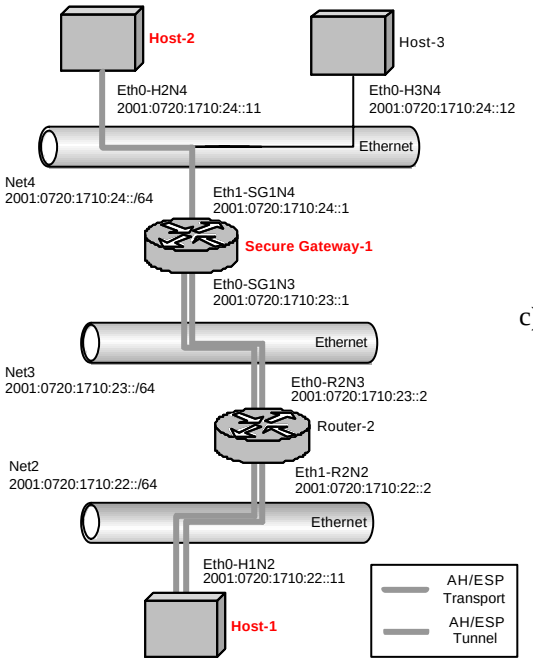
This scenario is showed on below figure. In the figure, Host-1 and Secure Gateway-1 establish the first SA and Secure Gateway-3 or Host-2 get the second SA with Host-1.



a) IPsec Tunnel mode with Host



c) IPsec Tunnel to IPsec Tunnel mode with Host



b) Transport mode and tunnel mode

Figure 1-17: Both host and SG

1.4.3 Test Suite

The test suite are tools to measure network throughput and advanced applications to evaluate the network performance when we use real applications. This test suite will be used with the several scenarios. We consider two groups of test tools:

- Basic tools. These tools are to measure network throughput and to test the IPsec conformance.
- Advanced applications. These tools are real applications to test the IPsec conformance.

The result of the test using these tools will be reported and will establish the features and limitations for each scenario.

1.4.3.1 Basic tools

These tools are to measure network throughput using TCP and UDP packets and ICMP message. These tools are below:

- ping6: send ICMPv6 ECHO_REQUEST packets to network hosts.
- telnet: user interface to the TELNET protocol.
- pchar: perform network measurements along an Internet path.

The most important parameter that we can measure to evaluate the network throughput is RTT (Round Trip Time). It is a measure of the time it takes for a packet to travel from a computer, across a network to another computer, and back.

1.4.3.2 Advanced applications

These applications are real applications require a minimal network performance. We will evaluate these applications using our several scenarios and will define the better working conditions for each scenario. Currently, the only advanced application considered is ISABEL. The ISABEL CSCW application is a group communication tool for the Internet, based on advanced videoconferencing features.

1.5 Evaluation

We have evaluated the IPsec/IKE interoperability and the conformance of all proposal solutions. First we have tested the interoperability with uni-implementation environment and later the interoperability with multi-implementation environment. We have used the test suite and PKIv6 [24] to test the IPsec/IKE solutions. These evaluations have generated several reports, theses are configuration and installations guides and test reports.

The reports are grouped by the type of scenario. The following figure shows an overview of all proposal scenarios.

No.	Category	Target
(1)	Transport mode AH	Host To Host
(2)	Transport mode ESP	
(3)	Tunnel mode AH	
(4)	Tunnel mode ESP	
(5)	Transport mode AH & ESP	
(6)	Transport mode AH & tunnel mode ESP	
(7)	Tunnel mode AH	Secure Gateway To Secure Gateway
(8)	Tunnel mode ESP	
(9)	AH Transport to ESP Tunnel mode	
(10)	IPsec Tunnel to IPsec Tunnel mode	
(11)	IPSec Tunnel mode with HOST	Host To Secure Gateway
(12)	IPsec Tunnel to IPsec Tunnel mode with HOST	
(13)	Transport mode & tunnel mode	

Figure 1-18: IPsec scenarios

The following table shows syntax of test suite that we use to evaluate the scenarios. Ping Type I and Type II return RTT minimum, average and maximum. Pchar returns several measures but the most important is RTT average of the path. Telnet does not return any measure but we use it to test TCP/IPv6 communication.

Test Suite	*Description*	*Syntax*
Ping Type I	Repeat 10000 times, with 64 bytes ICMP payload, interval 10 ms.	# ping6 -s 64 -i 0.01 -c 10000 ipv6_addr
Ping Type II	Repeat 10000 times, with 1024 bytes ICMP payload, interval 10 ms.	# ping6 -s 1024 -i 0.01 -c 10000 ipv6_addr
Pchar	Using UDP/IPv6, packet size increments from 32 to 1500 by 32.	# pchar -v –R 10 –s 3 –g 0.01 –b 20 ipv6_addr
Telnet	Using TCP/IPv6.	# telnet -6 ipv6_addr

Figure 1-19: Test Suite

The evaluation has generated a lot of measures and reports. This paragraph shows the results of several tests and the most important consideration about the implementations evaluated. The following results have been obtained with the basic tools of test suite. The tests using advanced applications have been realized in the project trials.

1.5.1 No secure gateway applies IPsec, only hosts do

For the configuration where only hosts apply IPsec, the interoperability is only complete between FreeS/WAN and KAME. The following table shows the possible combinations.

IPsec Implementations		*AH Transport*	*ESP Transport*	*AH Tunnel*	*ESP Tunnel*
Host 1	*Host 2*				
FreeS/WAN	FreeS/WAN	*support*	*support*	*support*	*support*
KAME	KAME	*support*	*support*	*support*	*support*

Windows	Windows	*support*	*support*	*support*	--
Solaris	Solaris	*support*	*support*	--	--
6WIND	6WIND	--	--	--	--
FreeS/WAN	KAME	*support*	*support*	*support*	*support*
FreeS/WAN	Windows	--	--	--	--
FreeS/WAN	Solaris	--	--	--	--
FreeS/WAN	6WIND	--	--	--	--
KAME	Windows	*support*	*support*	--	--
KAME	Solaris	*support*	*support*	--	--
KAME	6WIND	--	--	--	--
Windows	Solaris	*support*	*support*	--	--
Windows	6WIND	--	--	--	--
Solaris	6WIND	--	--	--	--

Figure 1-20: Host-To-Host IPsec Interoperability

The first four rows show the interoperability for the uni-implementation environments and the rest of rows do the interoperability for the multi-implementation environments.

The scenarios ‘AH Transport’ and ‘ESP Transport’ are the most supported. The following table shows the results for the scenario ‘ESP Transport’. The title ‘problems’ is normal for this scenario.

IPsec Implementations		*ESP Transport*
Host 1	*Host 2*	
FreeS/WAN	FreeS/WAN	*problems*
KAME	KAME	*support*
Windows	Windows	--
Solaris	Solaris	*support*
6WIND	6WIND	--
FreeS/WAN	KAME	*problems*
FreeS/WAN	Windows	--
FreeS/WAN	Solaris	--
FreeS/WAN	6WIND	--
KAME	Windows	*problems*
KAME	Solaris	*problems*
KAME	6WIND	--
Windows	Solaris	*problems*
Windows	6WIND	--
Solaris	6WIND	--

Figure 1-21: Scenario 'ESP Transport'

Considerations:

- Windows XP supports ESP but does not support data encryption.
- FreeS/WAN loses IPsec packets.
- The authentication process goes wrong between Solaris and KAME.

The interoperability between solutions exists only between KAME stacks; it is also possible with FreeS/WAN but with problems.

The following figure shows a graphic that relates IPsec/IKE solutions with RRT increase for ping Type I and Type II. For example, KAME with preshared keys for 'ping Type I' generates increase in RTT about 14%.

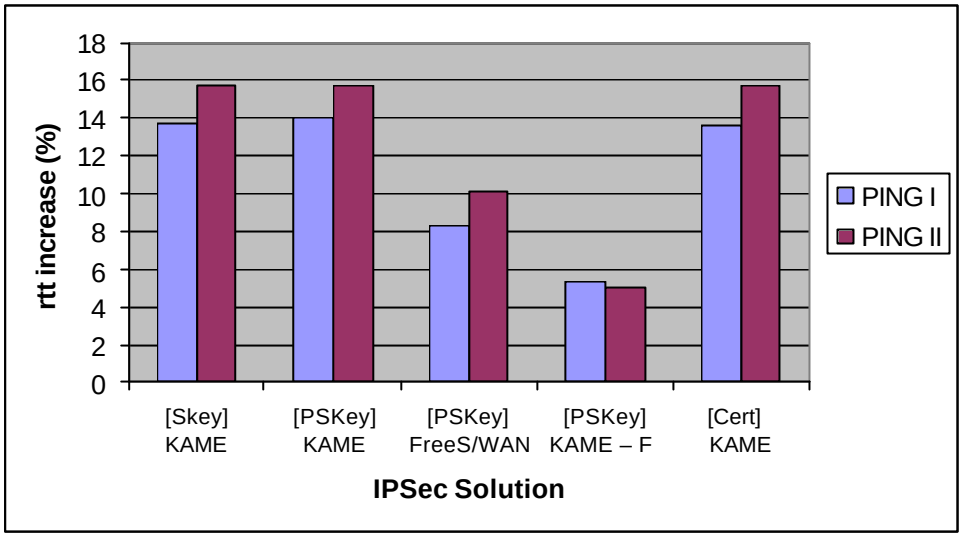


Figure 1-22: RTT for Transport ESP (ping)

Considerations:

- FreeS/WAN version evaluated does not have installed the patch that enable IKE with certificates.
- 3DES-CBC and HMAC_MD5 are the algorithms used.

KAME has a RTT increase that is similar for preshared keys, certificates and manual keyed. FreeS/WAN has results better than KAME. RTT increase is never over 16%.

1.5.2 No host applies IPsec, only secure gateways do

For the configuration where only secure gateways apply IPsec, the interoperability is only complete between 6WIND and KAME. The following table shows the possible combinations.

IPsec Implementations		*ESP Tunnel*	*AH Tunnel*	*AH Transport to ESP Tunnel mode*	*IPsec Tunnel to IPsec Tunnel mode*
SG 1	*SG 2*				
FreeS/WAN	FreeS/WAN	*support*	*support*	--	--
KAME	KAME	*support*	*support*	*support*	*support*
Windows	Windows	--	--	--	--
Solaris	Solaris	--	--	--	--
6WIND	6WIND	*support*	*support*	*support*	*support*
FreeS/WAN	KAME	*support*	*support*	--	--
FreeS/WAN	Windows	--	--	--	--
FreeS/WAN	Solaris	--	--	--	--
FreeS/WAN	6WIND	*support*	*support*	--	--
KAME	Windows	--	--	--	--

KAME	Solaris	--	--	--	--
KAME	6WIND	*support*	*support*	*support*	*support*
Windows	Solaris	--	--	--	--
Windows	6WIND	--	--	--	--
Solaris	6WIND	--	--	--	--

Figure 1-23: SecureGateway-To-SecureGateway IPsec Interoperability

The scenarios 'AH Tunnel' and 'ESP Tunnel' are the most supported. The following table shows the results for the scenario 'ESP Tunnel'.

IPsec Implementations		*ESP Tunnel*
SG 1	*SG 2*	
FreeS/WAN	FreeS/WAN	*support*
KAME	KAME	*support*
Windows	Windows	--
Solaris	Solaris	--
6WIND	6WIND	*support*
FreeS/WAN	KAME	*support*
FreeS/WAN	Windows	--
FreeS/WAN	Solaris	--
FreeS/WAN	6WIND	*support*
KAME	Windows	--
KAME	Solaris	--
KAME	6WIND	*support*
Windows	Solaris	--
Windows	6WIND	--
Solaris	6WIND	--

Figure 1-24: Scenario 'ESP Tunnel'

The interoperability between solutions exists but we have to keep in mind several considerations:

- Preshared keys have to be given in hexadecimal notation when we use FreeS/WAN and any other implementation as 6WIND or KAME, as they interpret the ASCII strings in a different manner.
- Because FreeS/WAN uses PFS in default configuration, in contrast to KAME, we either had to activate PFS in KAME or deactivate PFS in FreeS/WAN. We took the first option.
- The key lifetimes in both gateways have to be the same.
- FreeS/WAN allows starting the renegotiation of the keying material in the IKE daemon before it is expired.

The following figure shows a graphic that relates IPsec/IKE solutions with RRT increase for pchar. For example, KAME with preshared keys for pchar generates increase in RTT about 18%.

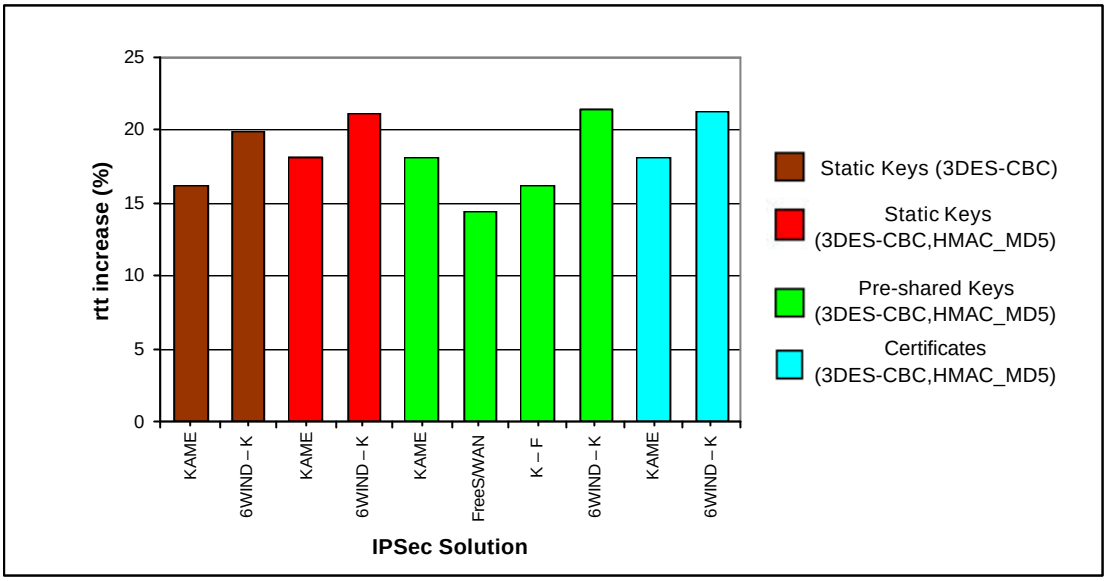


Figure 1-25: RTT for Tunnel ESP (pchar)

Considerations:

- FreeS/WAN version evaluated does not have installed the patch that enable IKE with certificates.
- 3DES-CBC and HMAC_MD5 are the algorithms used.

KAME has a RTT increase that is similar for preshared keys, certificates and manual keyed. FreeS/WAN has results better than KAME. The worst results are obtained with 6WIND. RTT increase is the band 15%-20%.

The following figure shows a graphic that relates IPsec/IKE solutions with RRT increase for ping Type I and Type II. This graphic confirms the comments above.

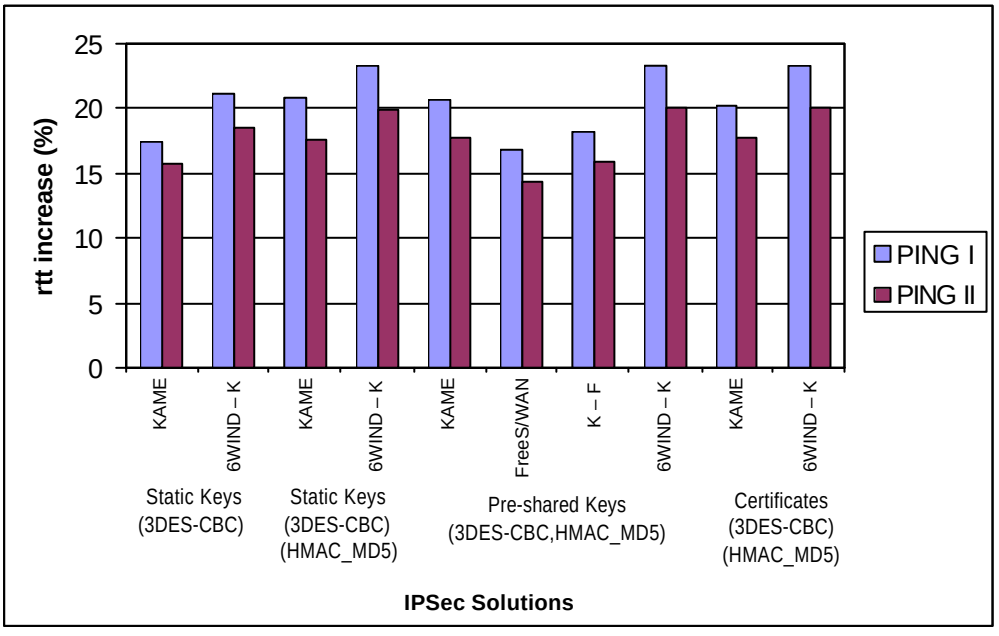


Figure 1-26: RTT for Tunnel ESP (ping)

We have modified the IKE daemons Raccon and Pluto to get the time of IKE negotiation. The following figure shows this time (ms).

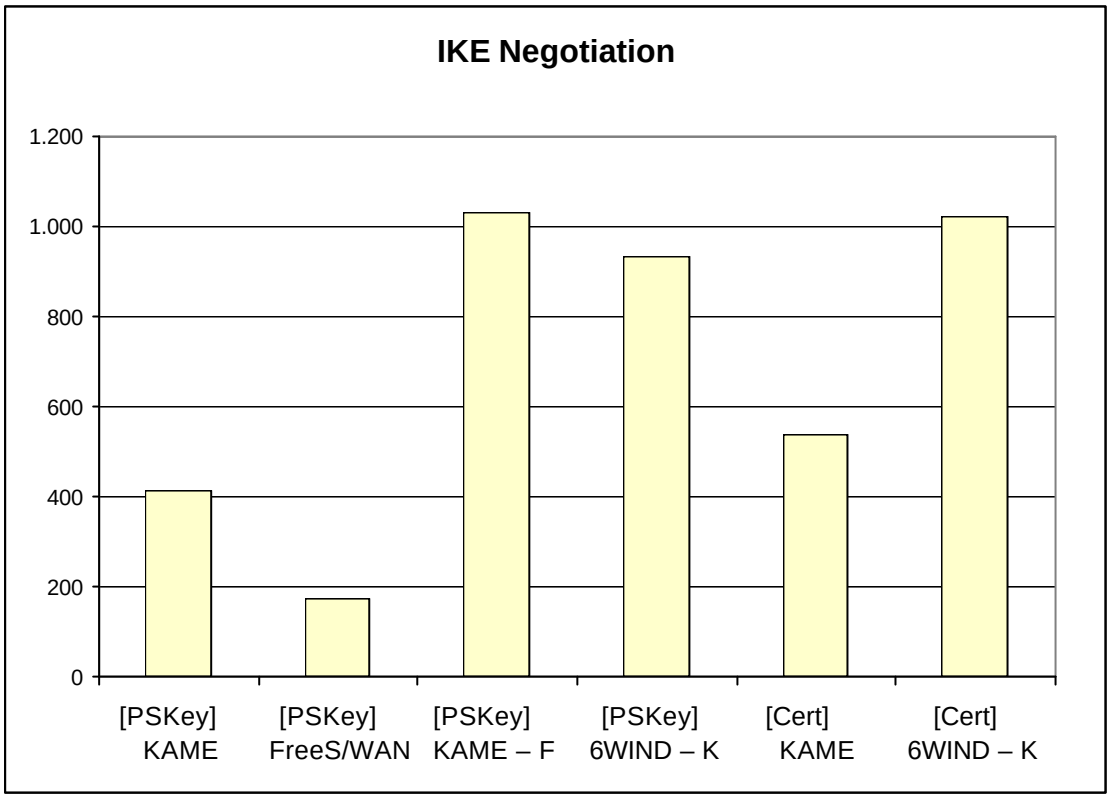


Figure 1-27: Time of IKE for Tunnel ESP

The worst results are when we use different implementations. The best result is obtained with FreeS/WAN.

1.5.3 Both host and secure gateway apply IPsec

For the configuration where both host and secure gateway apply IPsec, there are a lot of the combinations between implementations. The following table shows the possible combinations.

IPsec Implementations		*ESP Tunnel mode with host*	*AH Tunnel mode with host*
Secure Gateway	*Host*		
FreeS/WAN	FreeS/WAN	*future*	*future*
FreeS/WAN	KAME	*no Road Warrior*	*no Road Warrior*
KAME	KAME	*no Road Warrior*	*no Road Warrior*
KAME	FreeS/WAN	*no Road Warrior*	*no Road Warrior*
6WIND	KAME	*no Road Warrior*	*no Road Warrior*
6WIND	FreeS/WAN	*no Road Warrior*	*no Road Warrior*

Figure 1-28:Host-To-SecureGateway IPsec Interoperability

The objective of a Road Warrior scenario is that mobile users which have a dynamic changing IPv6 address depending on its connection to the network can access the private network using IPsec.

Considerations:

- The interoperability between different implements is not complete. Besides this, you have to enter the IPv6 address in the configuration of every implementation, so the concept of Road Warrior does not really exist.
- FreeS/WAN road warrior is not fully implemented. Furthermore, FreeS/WAN road warrior has problems with address and routing autoconfiguration.
- The best scenario is KAME-KAME using IKE with certificates but transport mode, not tunnel mode.

1.5.4 Conclusions

Although the multi-implementation scenario is feasible, it is clear from the current stage of the implementations that the uni-implementation environments are more stable and provides a better performance. There are a lot of scenarios where the interoperability between different solutions does not exist and when it exists, a worse network throughput is obtained.

KAME and 6WIND provide the most complete IPsec/IKE support, but FreeS/WAN in the scenarios supported has better network throughput. On the other hand, commercial operating systems are far from having complete IPv6 IPsec implementations as, in some cases, they have for IPv4.

The RTT charts show that the use of IPsec compared with the case of not using it, increase this measure in the band of 15%-20%. The same level of increment is obtained when we use IKE compared with the case of not using it. Both values are not considered to be high. Furthermore, using authentication compared to the case of not using it increases the RTT very lowly.

Normally, implementations are manually configured. This is clearly prone to errors and creates a real disadvantage of IPsec compared to other higher-level security solutions, as SSL/TSL or SSH.

In any case we want to remark that due to the fact that these are on-going implementations a major number of the possible incompatibilities or problems may be easily solved during next months.

1.6 Approach to Dynamic VPNs

The IPsec configuration and administration are difficult and complex. The IPsec management policies provides a dynamic and automated mechanism to create IPsec VPNs, and it is called Dynamic VPNs. The concept of IPsec management policy is related to one or more rules which in turn are decomposed in IPsec/IKE conditions and actions.

The following figure shows the architecture of a policy based VPN management.

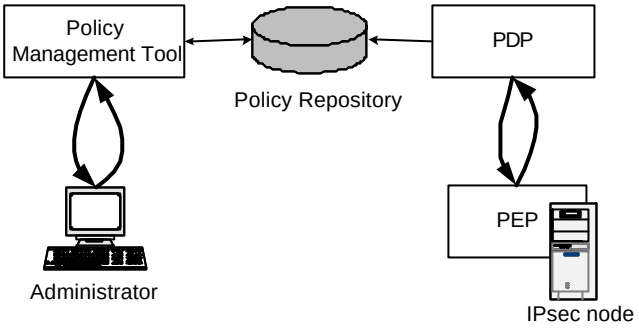


Figure 1-29: Policy based VPN management

Components:

- Policy Management Tool. The management of policies will be executed through this tool to allow constructing policies, deploying policies, and monitoring the status of the policy-managed environment. Policies are managed by an administrator.
- Policy decision point (PDP). It is the entity that decides if the conditions of a policy are fulfilled and as a consequence triggers the actions involved in that policy interacting with the PEP.
- Policy enforcement point (PEP). It is the entity that ensures that the actions ordered by the PDP are executed.
- Policy repository. It is a directory where policies and related information are stored.

We propose a preliminary architecture where there are not policies and the PDP and Policy Repository components are eliminated. This architecture is the first step to develop the whole policy system.

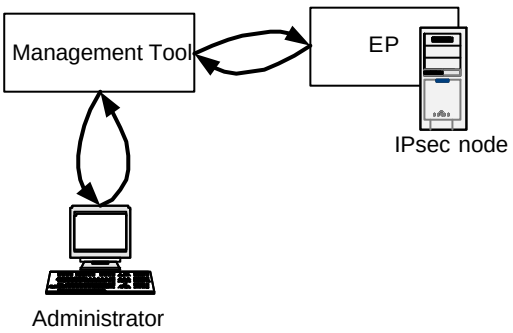


Figure 1-30: Basic VPN management

Components:

- Management Tool. The management of action will be executed through this tool by an administrator.
- Enforcement point (EP). It is the entity that ensures that the actions ordered by the Management Tool are executed.

We have developed this scenario using 6WIND routers. The Management Tool and EP are sited in the same machine called VPN EPoint that is the main element. The following figure shows the scenario.

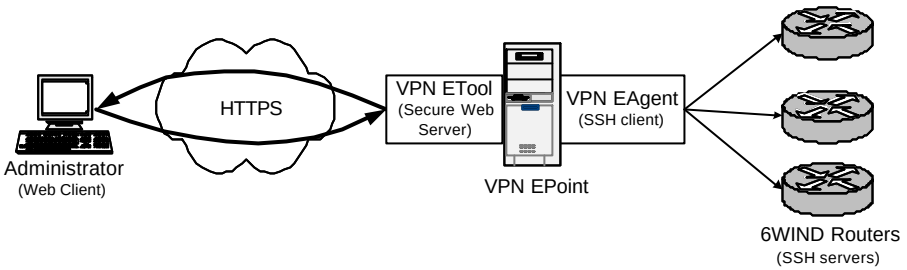


Figure 1-31: VPN Enforcement Point

VPN EPoint has two parts:

- VPN Enforcement Tool (VPN ETool). This is a web server and mainly the administrator interface. This is equivalent to Management Tool.
- VPN Enforcement Agent (VPN EAgent). This is a SSH client and applies the IPsec action defined by Administrator. This is equivalent to EP.

The following figure shows a screenshot of VPNETool. Once the administrator selects a router to connect, he access this main screen where an administrator can manage the router.

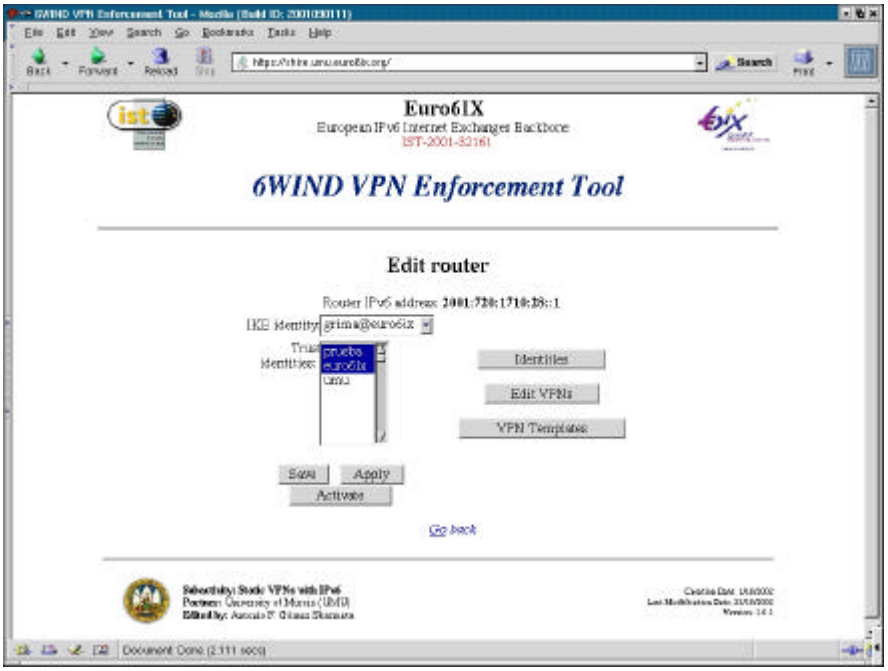


Figure 1-32: VPN Enforcement Tool

It is only an approach to Dynamic VPNs based policies. We expect to develop the whole system with multi-implementation support in the near semesters within the Euro6IX project.

1.7 VPNs in a IX: VPN Broker

IPsec seems to be the right way to protect the services and data communications of an IPv6 Internet Exchange, but the IPsec configuration and administration are clearly difficult and complex. The use of policy-based VPN management architectures provides a dynamic and automated mechanism to manage IPsec VPNs, so this service seems to be advisable for an IX.

The next picture shows an overview of the proposed scheme. VPN Broker is the set of components that the policy-based VPN management system consists of.

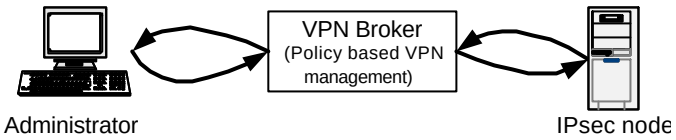


Figure 1-33: VPN Broker

The VPN Broker is sited in the IX (and maybe also in some ISPs), whilst the IPsec nodes are located in the client site. The administrator must be authenticated normally using certificates provided by the Euro6IX-PKIv6 service.

1.8 Examples and practical configuration

Along this section a few examples of practical configurations using IPv6sec will be shown. A general knowledge of the whole document will be highly advisable.

1.8.1 Simple host-to-host setup with FreeBSD+KAME

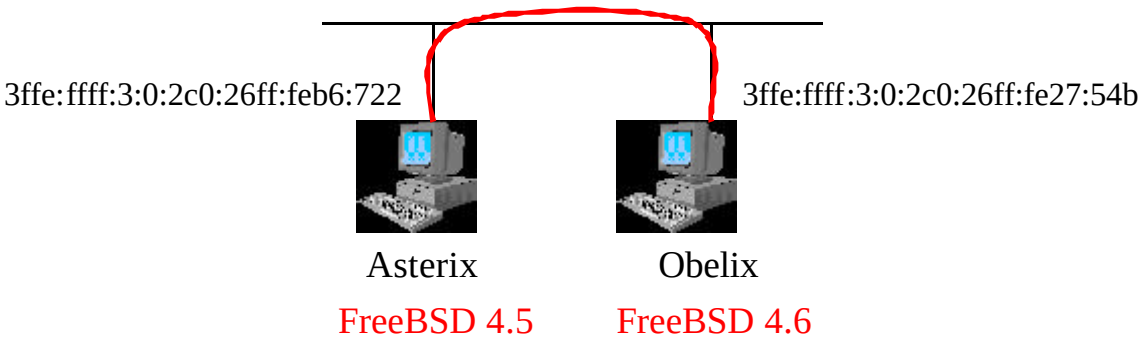


Figure 1-34: Simple host-to-host setup with FreeBSD+KAME

In this setup, two hosts running FreeBSD (version 4.5 and 4.6 respectively) will deploy a secure communication between them. In this setup, we will describe a host-to-host security, using transport mode with ESP headers, but other approaches are possible using this setup as a base.

In the example, the connection between the two hosts is a simple Ethernet LAN, but setting up the tunnel over a public network is not an extra problem, providing there is no firewall filtering or so.

We will show two different ways to setup the secure communication: using a static VPN and using an IKE negotiation.

We have seen `setkey` is in charge of setting up the SPD database to apply security to a given datagram, and `racoon` is used to perform the IKE negotiation to create the SAD database (in the static mechanism, `setkey` will statically define the used SAs), so we will focus in the configuration files for `setkey` and `racoon`:

- setkey –f <file> and racoon –F -f <file> to run them.
- setkey –D and setkey –DP can be used to show the SAD and SPD databases.
- setkey –F and setkey -FP are used to remove the SAD and SPD databases.

1.8.1.1 Static VPN

In this case, a static SA needs to be added to the SAD database, and `raacon` is not used (since there is not IKE negotiation).

Asterix: setkey.conf	**Obelix: setkey.conf**
`spdadd 3ffe:ffff:3:0:2c0:26ff:feb6:722 3ffe:ffff:3:0:2c0:26ff:fe27:54b any -P out ipsec esp/transport//require;`	`spdadd 3ffe:ffff:3:0:2c0:26ff:fe27:54b 3ffe:ffff:3:0:2c0:26ff:feb6:722 any -P out ipsec esp/transport//require;`
`spdadd 3ffe:ffff:3:0:2c0:26ff:fe27:54b 3ffe:ffff:3:0:2c0:26ff:feb6:722 any -P in ipsec esp/transport//require;`	`spdadd 3ffe:ffff:3:0:2c0:26ff:feb6:722 3ffe:ffff:3:0:2c0:26ff:fe27:54b any -P in ipsec esp/transport//require;`
`add 3ffe:ffff:3:0:2c0:26ff:feb6:722 3ffe:ffff:3:0:2c0:26ff:fe27:54b esp 1111 –E 3des-cbc "mykeyforthelink012345678"`	`add 3ffe:ffff:3:0:2c0:26ff:fe27:54b 3ffe:ffff:3:0:2c0:26ff:feb6:722 esp 2222 –E 3des-cbc "mykeyforthelink876543210"`
`add 3ffe:ffff:3:0:2c0:26ff:fe27:54b 3ffe:ffff:3:0:2c0:26ff:feb6:722 esp 2222 –E 3des-cbc "mykeyforthelink876543210"`	`add 3ffe:ffff:3:0:2c0:26ff:feb6:722 3ffe:ffff:3:0:2c0:26ff:fe27:54b esp 1111 –E 3des-cbc "mykeyforthelink012345678"`

Notice each association is unidirectional, so there is need to include one line for each way. The first two lines add an entry in the SPD database, and the two last lines, and a Security Association to the SAD database.

Once `setkey` is run with this configuration files, the traffic from one host to the other is secured.

1.8.1.2 Using IKE

In this case, there is no need to set a SA to be used for the communication, since it is negotiated using `raacon`. Pre-shared keys are used in the first phase of the IKE protocol (certificates can also be used, but it is not covered by this example).

Asterix: setkey.conf

```
spdadd 3ffe:ffff:3:0:2c0:26ff:feb6:722
3ffe:ffff:3:0:2c0:26ff:fe27:54b any -P out ipsec
esp/transport//require;

spdadd 3ffe:ffff:3:0:2c0:26ff:fe27:54b
3ffe:ffff:3:0:2c0:26ff:feb6:722 any -P in ipsec
esp/transport//require;
```

Obelix: setkey.conf

```
spdadd 3ffe:ffff:3:0:2c0:26ff:fe27:54b
3ffe:ffff:3:0:2c0:26ff:feb6:722 any -P out
ipsec esp/transport//require;

spdadd 3ffe:ffff:3:0:2c0:26ff:feb6:722
3ffe:ffff:3:0:2c0:26ff:fe27:54b any -P in ipsec
esp/transport//require;
```

Asterix: racoon.conf

```
path include "/etc/racoon" ;
# Use this file for the pre_shared_key
path pre_shared_key "/etc/racoon/psk.txt" ;
log debug;
# Don't touch padding unless you know what
# you are doing
padding
{
# maximum padding length.
maximum_length 20;
# enable randomize length.
randomize off;
# enable strict check.
strict_check off;
# extract last one octet.
exclusive_tail off;
}
# Since no listen directive  is given, raacon
# will listen to all
listen
{
}
timer
{
# maximum trying count to send.
counter 5;
# maximum interval to resend.
interval 10 sec;
# the number of packets per a send.
persend 1;
# timer for waiting to complete each phase.
phase1 15 sec;
phase2 10 sec;
}
# Configure phase 1 for the given IP address
remote 3ffe:ffff:3:0:2c0:26ff:fe27:54b
{
# List of the modes to try to use
exchange_mode aggressive,main;
# Avoid the usage of certificates
send_cert off;
send_cr off;
lifetime time 5 min;  # sec,min,hour
# Proposal for the 1st phase
proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key ;
        dh_group 2 ;
        }
}
# Info for the 2nd phase
sainfo anonymous
{
pfs_group 2;
lifetime time 1 hour;
# list of algorithms to try to negotiate
encryption_algorithm des, 3des, blowfish;
authentication_algorithm hmac_sha1, hmac_md5;
```

Obelix:racoon.conf

```
path include "/etc/racoon" ;
# Use this file for the pre_shared_key
path pre_shared_key "/etc/racoon/psk.txt" ;
log debug;
# Don't touch padding unless you know what
# you are doing
padding
{
# maximum padding length.
maximum_length 20;
# enable randomize length.
randomize off;
# enable strict check.
strict_check off;
# extract last one octet.
exclusive_tail off;
}
# Since no listen directive  is given, raacon
# will listen to all
listen
{
}
timer
{
# maximum trying count to send.
counter 5;
# maximum interval to resend.
interval 10 sec;
# the number of packets per a send.
persend 1;
# timer for waiting to complete each phase.
phase1 15 sec;
phase2 10 sec;
}
# Configure phase 1 for the given IP address
remote 3ffe:ffff:3:0:2c0:26ff:feb6:722
{
# List of the modes to try to use
exchange_mode aggressive,main;
# Avoid the usage of certificates
send_cert off;
send_cr off;
lifetime time 5 min;  # sec,min,hour
# Proposal for the 1st phase
proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key ;
        dh_group 2 ;
        }
}
# Info for the 2nd phase
sainfo anonymous
{
pfs_group 2;
lifetime time 1 hour;
# list of algorithms to try to negotiate
encryption_algorithm des, 3des, blowfish;
authentication_algorithm hmac_sha1, hmac_md5;
```

`compression_algorithm deflate ;` `}`	`compression_algorithm deflate ;` `}`

Asterix: psk.txt	**Obelix: psk.txt**
`3ffe:ffff:3:0:2c0:26ff:fe27:54b “apresharedkey”`	`3ffe:ffff:3:0:2c0:26ff:feb6:722 “apresharedkey”`

Once `setkey` and `racoon` are running, we can start to send traffic from one `asterix` to `obelix`. The first time some traffic goes through the secure channel, the IKE protocol is started. Since it takes from 4 to 8 seconds to complete the negotiation, the first datagrams in the channel are lost.

1.8.2 Simple gateway-to-gateway setup with 6WIND

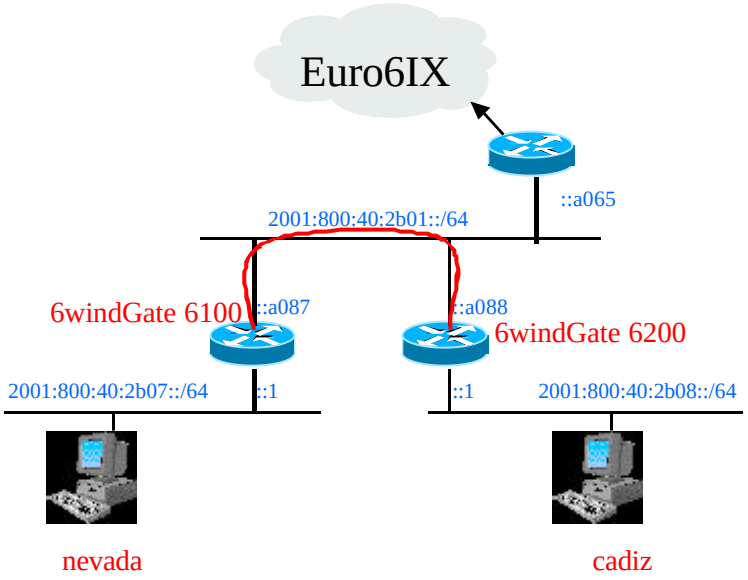


Figure 1-35: Simple gateway-to-gateway setup with 6WIND

In this setup, two 6windGate routers are used to set up a secure tunnel between two network gateways, but it can be used as a based for a different setup, as in the section above. There is only one host in each network, but including other computers, emulating a corporate network is pretty easy, since the whole /64 prefix is being secured: we want all the traffic from the first Ethernet LAN to the second one (and viceversa) to be secured through the tunnel in red.

Tunnel mode for ESP and transport mode for AH will be used in this setup (although maybe authentication is less useful in a gateway-to-gateway configuration that in host-to-host), using both a statically defined VPN and using IKE to negotiate the SA.

A general knowledge of the 6windGate routers configuration will be supposed along these examples. More in deep detail on the IPsec (both IPv4 and IPv6) configuration for the routers can be found in the 6windGate configuration guide itself. The only needed consideration in the 6windGate routers is the availability of *templates* to create these configurations. Once a template has been defined (the *static* template in the first example, the *psk_strong* one in the second one), there is only need to fill in a few parameters, since most of the IPsec parameters are already fixed in the template. The templates provided by the 6windGate default installation cover most of the usual situations, but new templates can be added if needed (this is out of the scope of this document, so the 6windGate manual should be used).

1.8.2.1 Static VPN

Since the VPN is being statically defined, there is no need to set the ID of the routers. The configuration is as follows:

	6100	**6200**
Enable security in the output interface	`r6w6100{myconfig-eth1_0}enable ipsec`	`r6w6200{myconfig-eth1_0}enable ipsec`
Define a VPN using the *static* template	`r6w6100{myconfig-sec}vpn static myvpn 2001:800:40:2b01::a087 2001:800:40:2b01::a088`	`r6w6100{myconfig-sec}vpn static myvpn 2001:800:40:2b01::a088 2001:800:40:2b01::a087`
Define the tunnel, using ESP tunnel mode and AH transport mode (esp_ah)	`r6w6100{myconfig-sec}tunnel mytunnel esp_ah 2001:800:40:2b07::/64 2001:800:40:2b08::/64 any myvpn`	`r6w6200{myconfig-sec}tunnel mytunnel esp_ah 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any myvpn`

Add the SA, using hmac-md5 algorithm for AH and 3des-cbc for ESP	`r6w6100{myconfig-sec}sa_ah 2001:800:40:2b01::a087 2001:800:40:2b01::a088 1111 hmac-md5 "mykeyforthelink1"`
	
	`r6w6100{myconfig-sec}sa_esp 2001:800:40:2b01::a087 2001:800:40:2b01::a088 2222 3des-cbc "myanotherkeyforthelink02"`
		`r6w6200{myconfig-sec}sa_ah 2001:800:40:2b01::a088 2001:800:40:2b01::a087 8888 hmac-md5 "mykeyforthelink1"`
		
		`r6w6200{myconfig-sec}sa_esp 2001:800:40:2b01::a088 2001:800:40:2b01::a087 9999 3des-cbc "myanotherkeyforthelink02"`

Once the configurations are saved and applied, a `ping6` from `nevada` to `cadiz` is secured in the path from one router to the other.

1.8.2.2 Using IKE

In this setup, we will use pre-shared keys to establish the first phase of the IPsec protocol. In this case, there is no need for a SA static definition, since it will be negotiated by the IKE protocol, but there is need to set the ID of the routers.

	6100	**6200**
Set the ID, with the FQDN and userFQDN parameters defined (others are not needed).	`r6w6100{}id myid r6w6100.upm.euro6ix.org abascal@dit.upm.es`	`r6w6200{}id myid r6w6200.upm.euro6ix.org abascal@dit.upm.es`

Enable security in the output interface	`r6w6100{myconfig-eth1_0}enable ipsec`	`r6w6200{myconfig-eth1_0}enable ipsec`
Define the ID to use for IKE	`r6w6100{myconfig-sec}ike_id myid`	`r6w6200{myconfig-sec}ike_id myid`
Set a pre-shared key to be used for the given IPv6 address	`r6w6100{myconfig-sec}psk 2001:800:40:2b01::a088`	`r6w6200{myconfig-sec}psk 2001:800:40:2b01::a087`
Define a VPN with the template *psk_strong*	`r6w6100{myconfig-sec}vpn myvpn psk_strong 2001:800:40:2b01::a087 2001:800:40:2b01::a088`	`r6w6200{myconfig-sec}vpn myvpn psk_strong 2001:800:40:2b01::a088 2001:800:40:2b01::a087`
Define the tunnel, using ESP tunnel mode and AH transport mode (esp_ah)	`r6w6100{myconfig-sec}tunnel mytunnel esp_ah 2001:800:40:2b07::/64 2001:800:40:2b08::/64 any myvpn`	`r6w6200{myconfig-sec}tunnel mytunnel esp_ah 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any myvpn`

Once the configurations are saved and applied, a `ping6` from `nevada` to `cadiz` is secured in the path from one router to the other.

1.8.3 Simple gateway-to-gateway setup with 6WIND and KAME

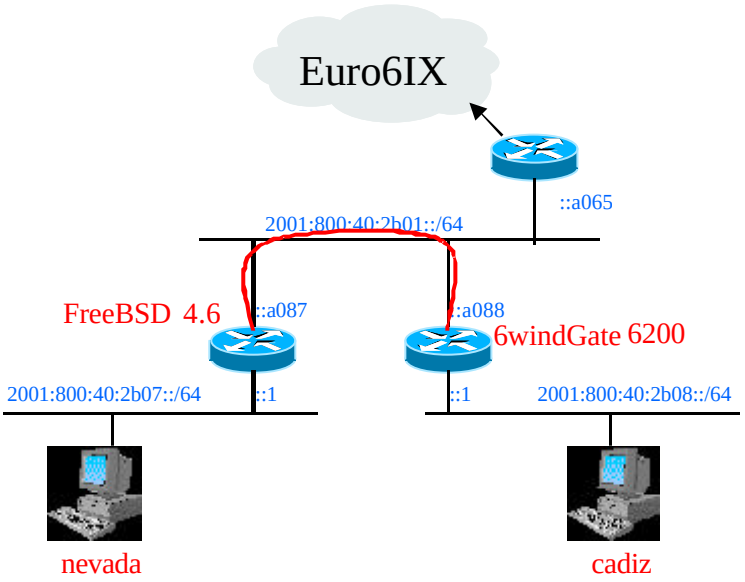


Figure 1-36: Simple gateway-to-gateway setup with 6WIND and KAME

In this setup, a 6windGate router and a router running KAME integrated FreeBSD version 4.6 are used to set up a secure tunnel between two network gateways. This setup shows the interoperability between IPsec implementations. The scheme used is the same shows above.

A general knowledge of the 6windGate routers configuration and FreeBSD/KAME configuration will be supposed along these examples.

1.8.3.1 Static VPN

The configuration for the 6windGate router is as follows:

	6200
Enable security in the output interface	`r6w6200{myconfig-eth1_0}enable ipsec`
Define a VPN using the *static* template	`r6w6200{myconfig-sec}vpn static myvpn 2001:800:40:2b01::a088 2001:800:40:2b01::a087`
Define the tunnel, using ESP tunnel mode	`r6w6200{myconfig-sec}tunnel mytunnel esp 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any myvpn`

Add the SA, using hmac-md5 and 3des-cbc algorithms for ESP	`r6w6200{myconfig-sec}sa_esp 2001:800:40:2b01::a088 2001:800:40:2b01::a087 1234 3des-cbc "myanotherkeyforthelink02" hmac-md5 "mykeyforthelink1"`
	
	`r6w6200{myconfig-sec}sa_esp 2001:800:40:2b01::a088 2001:800:40:2b01::a087 4321 3des-cbc "myanotherkeyforthelink02" hmac-md5 "mykeyforthelink1"`

The configuration for the router running KAME integrated FreeBSD version 4.6 is as follows:

FreeBSD 4.6

`spdadd 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any -P in ipsec esp/tunnel/2001:800:40:2b01::a088-2001:800:40:2b01::a087/require;`
 	
`spdadd 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any -P out ipsec esp/tunnel/2001:800:40:2b01::a088-2001:800:40:2b01::a087/require;`
 	
`add 2001:800:40:2b01::a088 2001:800:40:2b01::a087 esp 1234 -m tunnel -E 3des-cbc " myanotherkeyforthelink02" -A hmac-md5 "mykeyforthelink1" ;`
 	
`add 2001:800:40:2b01::a087 2001:800:40:2b01::a088 esp 4321 -m tunnel -E 3des-cbc " myanotherkeyforthelink02" -A hmac-md5 "mykeyforthelink1" ;`

Once the configurations are saved and applied, a `ping6` from `nevada` to `cadiz` is secured in the path from one router to the other.

1.8.3.2 Using IKE

Using pre-shared keys

In this setup, we will use pre-shared keys. The configuration for the 6windGate router is as follows:

	6200
Set the ID, with the FQDN and userFQDN parameters defined (others are not needed).	`r6w6200{}id myid r6w6200.umu.euro6ix.org fgarcia@umu.euro6ix.org`
Enable security in the output interface	`r6w6200{myconfig-eth1_0}enable ipsec`
Define the ID to use for IKE	`r6w6200{myconfig-sec}ike_id myid`
Set a pre-shared key to be used for the given IPv6 address	`r6w6200{myconfig-sec}psk 2001:800:40:2b01::a087 0x12345678`
Define a VPN with the template *psk_strong*	`r6w6100{myconfig-sec}vpn myvpn psk_strong 2001:800:40:2b01::a088 2001:800:40:2b01::a087`

Define the tunnel, using ESP tunnel mode	`r6w6200{myconfig-sec}tunnel mytunnel esp 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any myvpn`

The configuration for the router running KAME integrated FreeBSD version 4.6 is as follows:

FreeBSD 4.6

setkey.conf

```
spdadd 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any –P in ipsec
esp/tunnel/2001:800:40:2b01::a088-2001:800:40:2b01::a087/require;

spdadd 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any -P out ipsec
esp/tunnel/2001:800:40:2b01::a088-2001:800:40:2b01::a087/require;
```

racoon.conf

```
path include "/etc/racoon" ;
# Use this file for the pre_shared_key
path pre_shared_key "/etc/racoon/psk.txt" ;
log debug;
# Don't touch padding unless you know what
# you are doing
padding
{
# maximum padding length.
maximum_length 20;
# enable randomize length.
randomize off;
# enable strict check.
strict_check off;
# extract last one octet.
exclusive_tail off;
}
# Since no listen directive  is given, raacon
# will listen to all
listen
{
}
timer
{
# maximum trying count to send.
counter 5;
# maximum interval to resend.
interval 10 sec;
# the number of packets per a send.
persend 1;
# timer for waiting to complete each phase.
phase1 15 sec;
phase2 10 sec;
}
# Configure phase 1 for the given IP address
remote 2001:800:40:2b01::a088
{
# List of the modes to try to use
exchange_mode aggressive,main;
# Avoid the usage of certificates
send_cert off;
send_cr off;
lifetime time 1 hour; # sec,min,hour
lifetime byte 6000 KB ;
# Proposal for the 1st phase
proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key ;
        dh_group 2 ;
        }
}
```

```
# Info for the 2nd phase
sainfo anonymous
{
pfs_group 2;
lifetime time 1 hour;
lifetime byte 1000 MB ;
# list of algorithms to try to negotiate
encryption_algorithm 3des;
authentication_algorithm hmac_sha1;
compression_algorithm deflate ;
}
```

psk.txt

```
2001:800:40:2b01::a088 0x12345678
```

Once the configurations are saved and applied, a `ping6` from `nevada` to `cadiz` is secured in the path from one router to the other.

Using certificates

In this setup, we will use certificates generated by PKIv6. The configuration for the 6windGate router is as follows:

	6200
Set the ID, with the FQDN and userFQDN parameters defined (others are not needed).	`r6w6200{}id myid r6w6200.umu.euro6ix.org fgarcia@umu.euro6ix.org ES none none Euro6IX Euro6IX r6w6200 fgarcia@dif.um.es`
Set up the Certification Authority.	`r6w6200{}ca pkiv6 scp://pki.umu.euro6ix.org/ca` `r6w6200{}import ca_cert pkiv6`
Set up the 6windgate certificate.	`r6w6200{}cert_req r6w6200 pkiv6` `r6w6200{}export cert_req r6w6200 pkiv6`
After PKIv6 generate the certificate, import the certificate.	`r6w6200{}import cert_req r6w6200 pkiv6`
Enable security in the output interface	`r6w6200{myconfig-eth1_0}enable ipsec`
Define the ID to use for IKE	`r6w6200{myconfig-sec}trust pkiv6` `r6w6200{myconfig-sec}ike_id myid`
Define a VPN with the template *psk_strong*	`r6w6100{myconfig-sec}vpn myvpn cer_strong 2001:800:40:2b01::a088 2001:800:40:2b01::a087 pkiv6`
Define the tunnel, using ESP tunnel mode	`r6w6200{myconfig-sec}tunnel mytunnel esp 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any myvpn`

The configuration for the router running KAME integrated FreeBSD version 4.6 is as follows:

FreeBSD 4.6

setkey.conf

```
spdadd 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any –P in ipsec
esp/tunnel/2001:800:40:2b01::a088-2001:800:40:2b01::a087/require;

spdadd 2001:800:40:2b08::/64 2001:800:40:2b07::/64 any -P out ipsec
```

```
esp/tunnel/2001:800:40:2b01::a088-2001:800:40:2b01::a087/require;
```

racoon.conf

```
path include "/etc/racoon" ;
# Use this directory for the certificates
path certificate "/etc/ipsec/cert" ;
log debug;
# Don't touch padding unless you know what
# you are doing
padding
{
# maximum padding length.
maximum_length 20;
# enable randomize length.
randomize off;
# enable strict check.
strict_check off;
# extract last one octet.
exclusive_tail off;
}
# Since no listen directive  is given, raacon
# will listen to all
listen
{
}
timer
{
# maximum trying count to send.
counter 5;
# maximum interval to resend.
interval 10 sec;
# the number of packets per a send.
persend 1;
# timer for waiting to complete each phase.
phase1 15 sec;
phase2 10 sec;
}
# Configure phase 1 for the given IP address
remote 2001:800:40:2b01::a088
{
# List of the modes to try to use
exchange_mode aggressive,main;
lifetime time 1 hour; # sec,min,hour
lifetime byte 6000 KB ;
my_identifier asn1dn;
certificate_type x509 "r6kameUMU.cert" "r6kameUMU.priv";
# Proposal for the 1st phase
proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method rsasign;
        dh_group 2 ;
        }
}
# Info for the 2nd phase
sainfo anonymous
{
pfs_group 2;
lifetime time 1 hour;
lifetime byte 1000 MB ;
# list of algorithms to try to negotiate
encryption_algorithm 3des;
authentication_algorithm hmac_sha1;
compression_algorithm deflate ;
}
```

The private key of router must be sited in the file `/etc/ipsec/cert/r6kameUMU.priv` and the corresponding certificate does in the file `/etc/ipsec/cert/r6kameUMU.cert`. You must put the issuer's certificates into directory `/etc/ipsec/cert/` and create a link as following:

```
# ln –s <CACert_file> `openssl x509 –noout –hash –in ca.pem`.0
```

Once the configurations are saved and applied, a `ping6` from `nevada` to `cadiz` is secured in the path from one router to the other.

1.9 Conclusions

The main result of this subactivity is that now we really know the scenarios supported by the most representative IPsec/IKE implementations. Also we have extended other projects approach in the analysis of the implementations taking into consideration a broader spectrum of systems.

The selected open-source solutions are the implementations of FreeS/WAN, USAGI and KAME. And the commercial ones are the implementations of Windows, Solaris and 6WIND. On the other hand we have designed several IPsec test-bed scenarios for IPv6. We have grouped these scenarios in three different sections:

- No secure gateway applies IPsec, only hosts do.
- No host applies IPsec, only secure gateways do.
- Both host and secure gateway apply IPsec.

The main results of the evaluation are:

- There are several scenarios where the interoperability between different solutions does not exist and when it exists, the network throughput is worse than in the case of using the same implementation in both peers. In that sense seems to be clear that it is better to use the same implementation.
- KAME and 6WIND provide the most complete IPsec/IKE support, but FreeS/WAN in the scenarios supported has better network throughput.
- The RTT (round trip time) charts show that the use of IPsec compared with the case of not using it, increase this measure in the band of 15%-20%. The same level of increment is obtained when we use IKE compared with the case of not using it. Both values are not considered to be high, but it clearly depends on the environment where we are using IPsec and IKE (e.g. wired vs wireless environments). Furthermore, using authentication compared to the case of not using it increases the RTT lowly.
- Normally, implementations are manually configured. This is clearly prone to errors and creates a real disadvantage of IPsec compared to other higher-level security solutions, as SSL/TSL or SSH.

The IPsec/IKE implementations are mature for IPv4 but currently they are not for IPv6, though there are a lot of efforts to get a full support of IPsec/IKE compliant RFCs in the IPv6 stacks.

An architecture of dynamic VPNs based on policies, seems to be the right way to build a VPN service in an IX. We have proposed a preliminary architecture that is the first step to develop a complete policy system, called VPN Broker, as an IX service.

1.10 Future work

Our intention is to continue developing the architecture of Dynamic VPNs and to create the VPN Broker service in the IXs; we will do it in a new subactivity for the next year called “Dynamic VPNs”.

Another important issue will be to link the work on VPN with the provision of services for IXs, and the integration of the VPN and, in general the security services, with another network services such as mobility and end-to-end applications. Also the relation and functionalities of services like Tunnel Brokers and its integration with the VPN services will be investigated.

2 NETWORK SECURITY

2.1 Introduction

This section has been divided in two different parts.

The first part describes the existing IPv6 network security elements. Differences between filter and firewall features are discussed. Next, a complete list of IPv6 filters and firewalls is presented, where the main features are detailed. Finally, an IPv6 filter and an IPv6 firewall are selected to be tested and the results obtained during the tests carried out in order to check these firewalls features are presented.

The second part presents a generalised network model applicable in the Euro6IX context. It defines interfaces between functional blocks and recommends network security measures both at these interfaces and at network elements to protect the network from attacks.

2.2 Objectives

The main objective of this paper is to set the network security requirements in the Euro6IX network. This includes a description and analysis of network security elements (filters and firewalls) and the presentation of a network security model applicable in the Euro6IX context.

2.3 Network Security Elements

Among the existing IPv6 network security elements, firewalls are the most important. Firewalls may be classified in two groups: stateless firewalls, also known as filters, and stateful firewalls. Filters watch network traffic, and restrict or block packets based on source and destination addresses or other static values. They are not aware of traffic patterns or data flows. A stateless firewall uses simple rule-sets that do not account for the possibility that a packet might be received by the firewall pretending to be something you asked for.

Stateful firewalls can watch traffic streams from end to end and are aware of communication paths and can implement various IPsec functions such as tunnels and encryption. Stateful firewalls maintain state information about a connection in memory tables, such as source and destination IP address and port number pairs, protocol types, connection state and timeouts. Stateful firewalling is inherently more secure than its stateless counterpart, simple packet filtering.

2.3.1 IPv6 Filters

2.3.1.1 Open Source Filters

Netfilter6/ip6tables (Linux)

Netfilter and iptables are the framework inside the Linux 2.4.x kernel that enables packet filtering, network address translation (NAT) and other packet mangling with IPv4. It is the re-designed and heavily improved successor of the previous 2.2.x ipchains and 2.0.x ipfwadm systems.

Netfilter is a set of hooks inside the Linux 2.4.x kernel's network stack which allows kernel modules to register call-back functions called every time a network packet traverses one of those hooks. Iptables is a generic table structure for the definition of rulesets. Each rule within an IP table consists out of a number of classifiers (matches) and one connected action (target). Netfilter, iptables and the connection tracking as well as the NAT subsystems together build the whole framework.

Native IPv6 filtering is only supported in kernel versions 2.4+. In older 2.2- you can only filter IPv6-in-IPv4 by protocol 41. Netfilter6 and ip6tables form together the IPv6 equivalent framework to the IPv4 netfilter/iptables framework. The implementation is a little bit behind the IPv4 version, but is already able to protect a host or a LAN against unwanted IPv6 traffic.

Due to the lack of connection tracking, only traditional, stateless packet filtering is possible. Stateful filtering, already implemented in IPv4, is still missing in IPv6, but is on the to-do list of the Netfilter project.

As of this writing, the most important features supported by ip6tables are:

- IPv6 policy based access control.
- ICMPv6 service.
- Logging.
- Stateless packet filtering for TCP, UDP and ICMPv6 packets.
- IPv6 source and destination address field.
- TCP or UDP source and destination port field.
- ICMPv6 packet type.
- Extra modules.
- TCP flags filtering.
- MAC address filtering.
- Packet length filtering.
- Packet rate limitation.
- Packet rejection with different types of error messages.

Ip6fw (FreeBSD)

IPv6 feature has been merged at FreeBSD from KAME Project since release 4.0. FreeBSD provides an IPv6 packet filter called "ip6fw". It is based on an old version of “ipfw”, the IPv4 packet filter, and does not contain as many features.

- IPv6 policy based access control.
- ICMPv6 service.
- Logging.
- Stateless packet filtering for TCP, UDP and ICMPv6 packets.

2.3.1.2 Commercial Filters

JUNOS IPv6 Firewall Filtering

The JUNOS Internet software provides a policy framework, which is a collection of JUNOS policies that include routing policies and firewall filter policies. These policies share some fundamental similarities. However, when referring to a firewall filter policy the term firewall filter is used.

Depending on the hardware configuration of the router, you can use firewall filters for the following purposes:

- On routers equipped with an Internet Processor II ASIC, you can control data packets, which are chunks of data transiting the router as they are forwarded from a source to a destination.

- On all routers, you can control the local packets, which are chunks of data that are destined for or sent by the Routing Engine.

With the Internet Processor II ASIC, you can use filters on data packets passing through the router to provide protocol-based firewalls, hinder denial of service (DoS) attacks, prevent falsifying of source addresses, create access control lists, and implement rate limiting (policing).

You can use the filters to restrict the local packets that pass from the router's physical interfaces to the Routing Engine. Such filters are useful in protecting the IP services that run on the Routing Engine, such as telnet, ssh, and Border Gateway Protocol (BGP), from denial-of-service attacks. You can define input filters and output filters. You can also use policing, or rate limiting, to provide a finer level of control over local packets destined for the Routing Engine.

In a firewall filter, you define one or more terms that specify the filtering criteria and the action to take if a match occurs. Each term consists of two components:

- Match conditions: Values or fields that the IPv6 packet must contain. You can define match conditions based on the following components:
  - IPv6 source address field.
  - IPv6 destination address field.
  - TCP or UDP source port field.
  - TCP or UDP destination port field.
  - IP protocol field.
  - Next header field.
  - Traffic class field.
  - ICMP packet type.
- Action: Specifies what to do if a packet matches the match conditions. Possible actions are to accept, discard, or reject a packet, or to take no action. In addition, statistical information can be recorded for a packet: it can be counted, logged, or sampled.

Cisco IOS Software

The Cisco IOS Software provides IPv6 standard Access Control Lists (ACLs) only in the 12.0 ST, 12.0 S, and 12.2 T Cisco IOS software release trains, starting at Cisco IOS Release 12.0(21)ST, 12.0(22)S, and 12.2(2)T, respectively. IPv6 standard Access Control Lists are used for basic traffic filtering functions. As in IPv4, IPv6 ACLs filter traffic based on source and destination addresses, inbound and outbound to a specific interface, and with an implicit deny at the end of an access list.

The Cisco IOS Firewall feature set is a security-specific option for Cisco IOS software. It integrates robust firewall functionality and intrusion detection for every perimeter of the network and enriches existing Cisco IOS security capabilities. However, it does not provide IPv6 features at this moment.

6WINDGate 6200 Series

Taking advantage of five years of Research and Development in IPv6, 6WIND has developed its own expertise in IPv6. Branded under the name SixOS, its software architecture supports the main standard technologies needed to manage a comprehensive portfolio of advanced IP services today. Due to a dual stack architecture these services are available both in IPv4 and in IPv6.
6WIND's IPv6 firewall completes the range of features built into its 6WINDGate 6200 Series. The 6WINDGate is an IP Access router which integrates, in a single equipment, all the features to provide a new bunch of IP services. These features include Differentiated Services (DiffServ) Quality of Service (QoS) management, Multicast, IP mobility, security mechanisms based on IP Security (IPSec), IKE and IP Firewall, for both present (IPv4) and emerging (IPv6) IP standards. The IP Filtering / Firewalling software specifications include packet filtering for both IPv4 and IPv6 protocols.

2.3.2 IPv6 Firewalls

2.3.2.1 Open Source Firewalls

Packet Filter (OpenBSD)

Packet Filter is the firewall package that is included with OpenBSD 3.0 and above. It was written to take the place of the IPF firewall that was found in previous versions of OpenBSD. Packet Filter supports many of the options supported by IPF. Packet Filter also has several features that IPF doesn't. It has better support for IPv6, including stateful inspection for TCP, UDP, and ICMPv6 packets. It also has a similar command structure.

As of this writing, the most important features supported by Packet Filter are:

- Dual IP Stack (IPv4 and IPv6 firewall).
- IPv6 and IPv4 policy based access control.
- Logging.
- ICMPv6 service.
- Stateful inspection for TCP, UDP and ICMPv6 packets.

2.3.2.2 Commercial firewalls

Check Point FireWall-1™ NG FP2

The Check Point FireWall-1 NG FP2 provides IPv6 support as an addition to the existing FireWall-1 NG FP2 release.

These are the supported platforms.

- Solaris 8 operating system is supported, in both 32-bit and 64-bit modes.
- Nokia's operating system IPSO 3.6 with kernel patch including IPv6 support for Check Point VPN-1 NG. IPSO supports IPv6 since release 3.4.1.
- Currently, these are the supported features by the CheckPoint FireWall-1 NG FP2.
- Dual IP Stack (IPv4 and IPv6 firewall).
- IPv6 and IPv4 policy based access control.
- Logging (with some limitations described below).
- ICMPv6 service.
- FTP service.
- Simple TCP and UDP services (like HTTP, SMTP, Telnet, etc.).
- fw6 command, for interfacing the IPv6 kernel.

All features that are not mentioned as supported, are not supported for IPv6 traffic, but are fully supported for IPv4, for example:

- VPN.
- NAT.
- Security Servers.
- Anti-spoofing.

The logging utility presents some limitations:

- IPv6 source and destination addresses are shown in logs in the Information field (and not in the Source and Destination fields). In addition, the Source and Destination columns should be ignored for IPv6 log records.
- IPv6 addresses are not resolved in the Log Viewer.
- FTP data connections from Microsoft Windows XP clients are reset due to usage of scoped IPv6 addresses.
- Passive FTP data connection are logged to a separate log record.
- Internal communication is IPv4 based. It is necessary to have functional IPv4 interfaces for internal and for Module-Management communication.
- IPv6 packets with extension headers are dropped by default and there is no way to allow them. Fragmented packets, which use such headers, are always dropped.

There is a limit to the number of unique IPv6 addresses (not the number of connections) that the Firewall-1 can handle simultaneously. A system that has been exposed since start-up time to more IPv6 addresses than this limit will drop all new connections made from IPv6 addresses not previously seen. The limit is set to 10,000 unique IPv6 addresses by default, and can be changed by modifying the fw_ip6_max_addrs global variable (via /etc/system on Solaris and by using modzap on Nokia).

2.3.3 Testing IPv6 Filters and Firewalls

2.3.3.1 Netfilter6 / ip6tables

Among the existing IPv6 filters, the open source filter iptables v1.2.6a has been selected. A PC with Redhat Linux release 7.3 and kernel 2.4.18-3 has been used. The PC has a Pentium 4 1,6 GHz processor and 256M of RAM. The installation of ip6tables is not obvious because it requires to modify the Linux kernel and to recompile it, so some Linux administration concepts are needed.

In order to test the filter functionality, only a FastEthernet network card is required. This interface is attached to a local network. Therefore, all the traffic will be sent to and received from the firewall itself. There are other IPv6 hosts attached to this LAN which will be used to carry out the tests.

First, netfilter6 main features and some of the latest patches have been tested. For this tests very simple rule-sets have been created in order to check that this features really work in different conditions. These conditions include:

- Testing different targets (accept, drop, log, reject).
- Testing default policies.
- Testing in both directions.

The tested features and the tests that have been applied to check their functionality are:

- Interface filtering: try to send and receive packets from different interfaces.
- Protocol filtering: try to send and receive TCP, UDP and ICMPv6 packets.
- IPv6 source and destination address field filtering: try to send and receive packets from different machines attached to the local IPv6 network.
- TCP source and destination port field filtering: try to establish different TCP connections like telnet, ftp or ssh.
- UDP source and destination port field filtering: try to send and receive different UDP packets, i.e. DNS or SNMP.
- ICMPv6 packet type filtering: try to send and receive different ICMPv6 packet types, i.e. echo-request or echo-reply.
- TCP flags filtering: try to send and receive TCP packets with different flags set.
- MAC address filtering: try to send and receive packets from different machines attached to the local IPv6 network.
- Packet length filtering: try to send and receive packets with different lengths.
- Packet rate limitation: try to send and receive packets at a specific rate.
- Packet rejection with different types of error messages: try to reject incoming packets and send back a specific error packet.
- Logging with different options: try to log incoming and outcoming packets specifying a log message, the level of logging or including some options of the packet header.

Different tools have been used to carry out the tests described above.

- Ftp, telnet and ssh are used to test TCP features.
- Dig (DNS) and SNMP utilities are used to test UDP features.
- Ping6 is used to test ICMP features and packet length limitations.
- Ethereal is used to control the packets sent and received by the hosts.

In second place, a more complex rule-set has been built in order to simulate a more realistic scenario. This is the configuration script that has been used.

```
# DEFINITIONS
IFAZ="eth0"
IP_ADDR="3ffe:3328:6:2::3"
DNS_SERVER="3ffe:3328:6:2::5"

# Flush all the rules, delete the user-defined chains and reset the counters
ip6tables -F
ip6tables -X
ip6tables -Z

# Set DROP as default policy for all the built-in chains
ip6tables -P INPUT DROP
ip6tables -P FORWARD DROP
ip6tables -P OUTPUT DROP

# Allow all kind of traffic in loopback interface
ip6tables -A INPUT  -i lo -j ACCEPT
ip6tables -A OUTPUT -o lo -j ACCEPT

# SYN-FLOODING protection
ip6tables -N syn-flood
ip6tables -A INPUT -i $IFAZ -p TCP --syn -j syn-flood
ip6tables -A syn-flood -m limit --limit 1/minute --limit-burst 3 -j RETURN
ip6tables -A syn-flood -j DROP

# Anti-spoofing
ip6tables -A INPUT -i $IFAZ -s $IP_ADDR -j DROP
ip6tables -A INPUT -i $IFAZ -s ::1 -j DROP

# Allow DNS
ip6tables -A INPUT  -i $IFAZ -p UDP -s $DNS_SERVER --sport 53 -j ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p UDP -d $DNS_SERVER --dport 53 -j ACCEPT

# Allow SSH
ip6tables -A INPUT  -i $IFAZ -p TCP --sport 22 -j ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p TCP --dport 22 -j ACCEPT

# Allow TELNET
ip6tables -A INPUT  -i $IFAZ -p TCP --sport 23 -j ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p TCP --dport 23 -j ACCEPT

# Allow FTP connections
ip6tables -A INPUT  -i $IFAZ -p TCP --sport 21 -j ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p TCP --dport 21 -j ACCEPT
# Allow active FTP
ip6tables -A INPUT  -i $IFAZ -p TCP --sport 20 -j ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p TCP --dport 20 -j ACCEPT
# Allow passive FTP
ip6tables -A INPUT  -i $IFAZ -p TCP --sport 1024:65535 --dport 1024:65535 -j
ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p TCP --sport 1024:65535 --dport 1024:65535 -j
ACCEPT

# Reject ident probes with a TCP RESET
ip6tables -A INPUT -i $IFAZ -p TCP --dport 113 -j REJECT --reject-with TCP-
reset
```

```
# Allow some types of ICMPv6 packets
ip6tables -A INPUT  -i $IFAZ -p icmpv6 --icmpv6-type 129 -j ACCEPT
ip6tables -A INPUT  -i $IFAZ -p icmpv6 --icmpv6-type 136 -j ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p icmpv6 --icmpv6-type 128 -j ACCEPT
ip6tables -A OUTPUT -o $IFAZ -p icmpv6 --icmpv6-type 135 -j ACCEPT

# Any UDP not already allowed is logged and then dropped
ip6tables -A INPUT  -i $IFAZ -p UDP -m limit --limit 1/minute --limit-burst 1
-j LOG --log-prefix "IP6TABLES UDP-IN: "
ip6tables -A INPUT  -i $IFAZ -p UDP -j DROP
ip6tables -A OUTPUT -o $IFAZ -p UDP -m limit --limit 1/minute --limit-burst 1
-j LOG --log-prefix "IP6TABLES UDP-OUT: "
ip6tables -A OUTPUT -o $IFAZ -p UDP -j DROP

# Any TCP not already allowed is logged and then dropped
ip6tables -A INPUT  -i $IFAZ -p TCP -m limit --limit 1/minute --limit-burst 1
-j LOG --log-prefix "IP6TABLES TCP-IN: "
ip6tables -A INPUT  -i $IFAZ -p TCP -j DROP
ip6tables -A OUTPUT -o $IFAZ -p TCP -m limit --limit 1/minute --limit-burst 1
-j LOG --log-prefix "IP6TABLES TCP-OUT: "
ip6tables -A OUTPUT -o $IFAZ -p TCP -j DROP

# Any ICMPv6 not already allowed is logged and then dropped
ip6tables -A INPUT  -i $IFAZ -p icmpv6 -m limit --limit 1/minute --limit-
burst 1 -j LOG --log-prefix "IP6TABLES ICMPv6-IN: "
ip6tables -A INPUT  -i $IFAZ -p icmpv6 -j DROP
ip6tables -A OUTPUT -o $IFAZ -p icmpv6 -m limit --limit 1/minute --limit-
burst 1 -j LOG --log-prefix "IP6TABLES ICMPv6-OUT: "
ip6tables -A OUTPUT -o $IFAZ -p icmpv6 -j DROP

# Anything else not already allowed is logged and then dropped
# It will be dropped by the default policy anyway
ip6tables -A INPUT  -i $IFAZ -m limit --limit 1/minute --limit-burst 1 -j LOG
--log-prefix "IP6TABLES UNKNOWN-IN: "
ip6tables -A INPUT  -i $IFAZ -j DROP
ip6tables -A OUTPUT -o $IFAZ -m limit --limit 1/minute --limit-burst 1 -j LOG
--log-prefix "IP6TABLES UNKNOWN-OUT: "
ip6tables -A OUTPUT -o $IFAZ -j DROP
```

The tools used to check that this script works properly are the same tools used in the first part of the test, where all the features were checked individually.

All the results have been positive and no bugs have been found in the new IPv6 features.

2.3.3.2 Check Point FireWall-1TM NG FP2

Among the available IPv6 firewalls we have decided to test the CheckPoint FireWall-1 NG FP2. The firewall has been installed in a Sun Ultra5 with a UltraSPARC IIi 360 MHz processor and 256M of RAM, running Solaris 8. In order to test the firewall functionality, only a FastEthernet network card is required. It is necessary to apply an IPv6 patch for FW-1 NG FP2 and get a specific license, both provided by CheckPoint.

FireWall-1 is formed by three basic components: the Management Server, the Firewall Modules, and the Management Clients. The Management Server and one Firewall Module have been installed in the SunUltra5, while one Management Client has been installed in a PC with Windows 2000. The internal communication between modules is IPv4-based, so functional IPv4 interfaces must exist. Once this is understood, the installation and configuration of FW-1 is quite simple, despite the fact that it is necessary to install a couple of Solaris 8 patches.

However, some problems have arisen during the ipv6 patch installation despite the instructions described in the release notes have been followed. Once the installation was completed and the firewall seemed enabled, some errors in the IPv6 filtering were detected. It seems these errors are related to the installation failures experienced previously.

At the moment of this writing these problems are still being solved. Maybe the IPv6 patch is not enough tested in all platforms and there is some difference that has not been covered yet. In this sense, a close collaboration with the IPv6 team of CheckPoint is being maintained.

2.4 Network Security Design

This section presents a generalised network model applicable in the Euro6IX context, defines interfaces between functional blocks, and recommends network security measures both at these interfaces and at network elements to protect the network from attacks.

2.4.1 The basic network model

Euro6IX presents an ambitious network design. Its highly structured and hierarchical nature eases the definition of functional blocks and standardised interfaces.

The following figure shows the proposed network model:

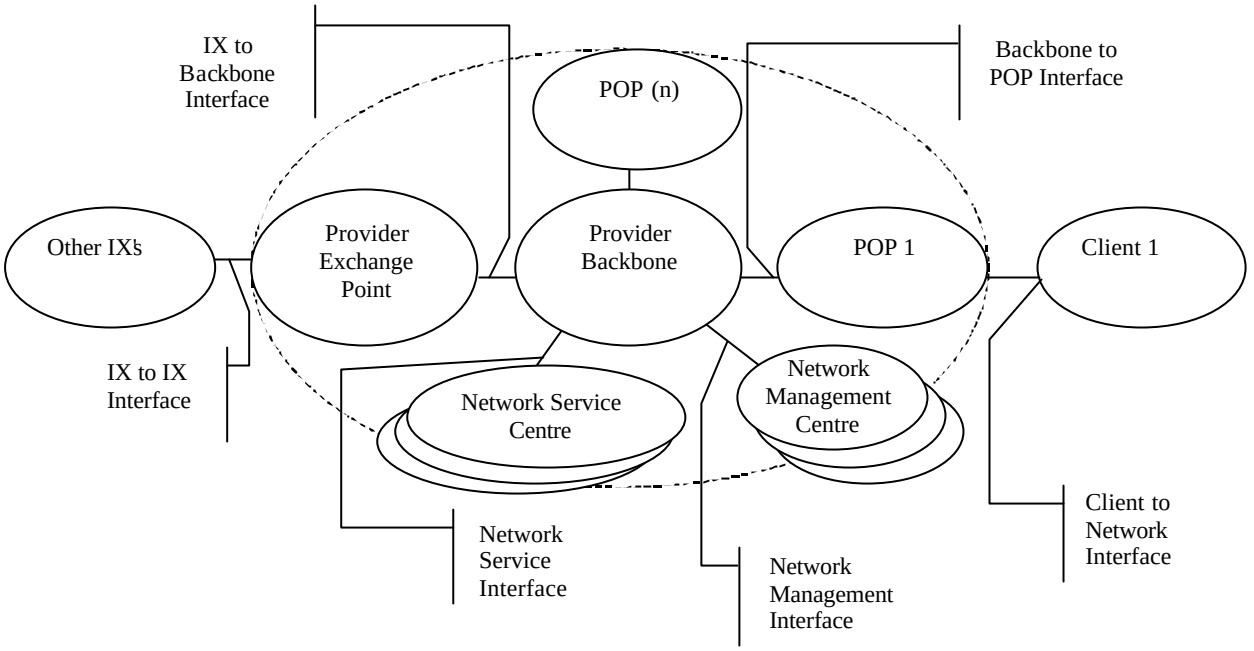


Figure 2-1: The Basic Network Model

The network is divided into the following functional blocks:

- Provider exchange point: The top provider exchange point (xIX) constitutes the of the hierarchy. It connects to the other exchange points in the Euro6IX network.
- Provider backbone: it constitutes the second level in the hierarchy and provides the connectivity between the xIX, the points of presence (POP) in the network and the network service centres.
- Point of presence (POP): the connection point of client's to the Euro6IX provider network.
- Network Service Centre: a logical location providing one or more common basic network services (i.e. DNS, NTP, etc.) to the provider's domain and its clients.
- Network Management Centre: a logical location implementing the basic network management functionalities (i.e. element configuration, statistics collection, etc).

2.4.2 Security measures at the different interfaces between functional blocks

2.4.2.1 xIX to xIX interface

This interface needs to combine security and performance requirements. Hence filtering has to be kept to a reasonable minimum, in order not to compromise the edge device's performance. Basic filtering which should be performed includes:

- Anti-spoofing: Rejecting incoming packets with forged origin addresses assigned to the network provider's address range.
- Protection of sensible subnetworks: rejecting incoming packets destined to ranges within the providers addressing space which have been reserved for special purposes.

2.4.2.2 xIX to backbone interface

This interface is interior to the network providers infrastructure. Since the current approach is based on peripheral security, no specific filtering is needed.

2.4.2.3 Backbone to PoP interface

This interface is interior to the network providers infrastructure. Since the current approach is based on peripheral security, no specific filtering is needed.

2.4.2.4 Backbone to service centre interface

Current best practices in security recommend that the network service centre should be isolated from the rest of the network via Firewalls, which should only allow the service specific traffic to and from the Network Service Centre.

2.4.2.5 Backbone to management centre interface

This interface is interior to the network providers infrastructure. Since the current approach is based on peripheral security, no specific filtering is needed. However, depending on the way the management is implemented (i.e. distributed management), it could be necessary to implement some security measures like firewalls.

2.4.2.6 Pop to client interface

This interface needs to combine security and performance requirements. Hence filtering has to be kept to a reasonable minimum, in order not to compromise the edge device's performance. Basic filtering which should be performed includes:

- Anti-spoofing: Rejecting incoming packets with forged origin addresses assigned to the network provider's address range.
- Protection of sensible subnetworks: rejecting incoming packets destined to ranges within the network provider's address range which have been reserved for special purposes.
- Origin checking: rejecting incoming packets which have not been originated in the client's address range.

Origin checking is an optional feature and should only be recommended when a the client is assigned static address range. It should also be considered in case of multi-homed clients to prevent them acting as a bypass between providers.

2.4.3 Security measures at network elements

The following rules apply to all network elements:

- Unused services shall not be active.
- CLI based management should use encrypted communications (i.e. SSH).
- Rate limiting mechanisms should be available at routers.
- All management accesses shall be limited by the appropriate mechanisms.
    - SNMP access shall be limited by communities and access-lists.

2.5 Conclusion

The analysed IPv6 filters and firewalls are still very far from their IPv4 equivalents and in some cases they are not enough tested. Porting IPv4 firewalling features to IPv6 is not trivial as the protocol stacks IPv4-IPv6 are separated. IPv6 security is still considered as experimental and there is not enough staff working on it. Efforts should concentrate to provide full connection tracking. Another drawback is the lack of documentation. There are very few references related to IPv6 filtering or firewalling.

2.6 Future Work

The problems encountered during the installation of CheckPoint FireWall-1 must be solved. The collaboration maintained with CheckPoint will be helpful.

The tested firewalls should be installed in a more realistic scenario in order to carry out more sophisticated tests.

Other firewalls described above which have not been tested (i.e. Packet Filter for OpenBSD) should be tested.

It is necessary to pay attention to the latest advances in the existing IPv6 firewalls and the presentation of new products from other companies.

3 REFERENCES

[1] IETF's IPsec Working Group, http://www.ietf.org/html.charters/ipsec-charter.html

[2] IP Authentication Header. S. Kent, R. Atkinson. RFC 2402. November 1998.

[3] IP Encapsulating Security Payload (ESP). S. Kent, R. Atkinson. RFC 2406. November 1998.

[4] Internet Security Association and Key Management Protocol (ISAKMP). D. Maughan, M. Schertler, M. Schneider, J. Turner. RFC 2408. November 1998.

[5] The OAKLEY Key Determination Protocol. H. Orman. RFC 2412. November 1998.

[6] The Internet Key Exchange (IKE). D. Harkins, D. Carrel. RFC 2409. November 1998.

[7] IP Payload Compression Protocol (IPComp). A. Shacham, B. Monsour, R. Pereira, M. Thomas. RFC 3173. September 2001.

[8] Krawczyk, H., "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", from IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security.

[9] Diffie, W., and Hellman M., "New Directions in Cryptography", IEEE Transactions on Information Theory, V. IT-22, n. 6, June 1977.

[10] Security Architecture for the Internet Protocol. S. Kent, R. Atkinson. RFC 2401 November 1998.

[11] IP Security Document Roadmap. R. Thayer, N. Doraswamy, R. Glenn. RFC 2411. November 1998.

[12] The Internet IP Security Domain of Interpretation for ISAKMP. D. Piper. RFC 2407. November 1998.

[13] IP Network Address Translator (NAT) Terminology and Considerations. P. Srisuresh, M. Holdrege. RFC 2663. August 1999.

[14] Randall J. Atkinson, Daniel L. McDonald, Bao G. Phan, Craig W. Metz, and Kenneth C. Chin, "Implementation of IPv6 in 4.4-Lite BSD", Proceedings of the 1996 USENIX Conference, San Diego, CA, January 1996, USENIX Association.

[15] Harney, H., and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification", RFC 2093, July 1997.

[16] Harney, H., and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture", RFC 2094, July 1997.

[17] HMAC: Keyed-Hashing for Message Authentication. H. Krawczyk, M. Bellare, R. Canetti. RFC 2104. February 1997.

[18] The MD5 Message-Digest Algorithm. R. Rivest. RFC 1321. April 1992.

[19] IP Authentication using Keyed SHA1 with Interleaved Padding (IP-MAC). P. Metzger, W. Simpson. November 2000.

[20] HMAC-MD5 IP Authentication with Replay Prevention, Oehler, M., and R. Glenn, RFC 2085, February 1997.

[21] NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995. http://csrc.nist.gov/fips/fip180-1.ps

[22] The Use of HMAC-RIPEMD-160-96 within ESP and AH. A. Keromytis, N. Provos. RFC 2857. June 2000.

[23] HMAC-MD5 IP Authentication with Replay Prevention. M. Oehler, R. Glenn. RFC 2085. February 1997.

[24] Euro6IX/UMU PKIv6, http://pki.umu.euro6ix.org/