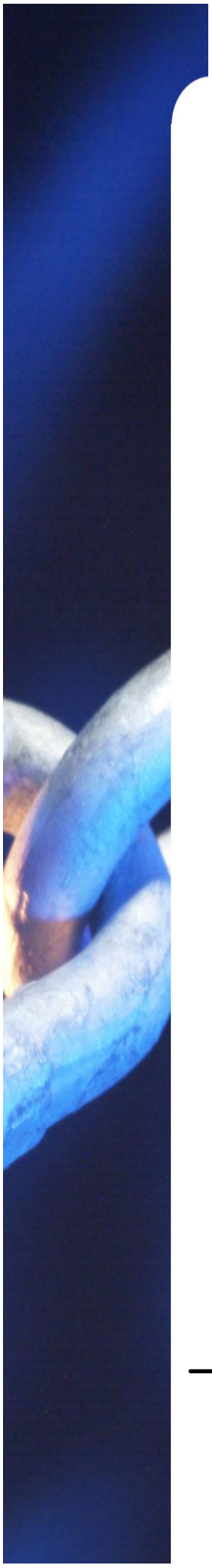

ipSEC EXPRESS

SSH IPSEC Express
Integration with Customer Applications
White Paper

Version 2.0, November 1999

ssh COMMUNICATIONS SECURITY

SSH IPSEC Express Integration with Customer Applications

White Paper

Version 2.0, November 1999

http://www.ipsec.com/

SSH Communications Security Ltd.
Tekniikantie 12
FIN-02150 Espoo
Finland

http://www.ipsec.com
E-mail: ipsec-sales@ssh.fi
Tel: +358-9-4354 3208
Fax: +358-9-4354 3222

Table of Contents

Introduction

The SSH IPSEC Express toolkit is a full, portable implementation of the IPSEC protocol that can be used by router, access router, VPN device, firewall vendors, and others for adding standards based security functionality into their existing or new products. It provides highly efficient and unrestricted cryptography for maximum security and efficiency.

As the security market is booming and the security methods are standardized and certified, an increasing number of vendors are adding IPSEC into their products. Two options for this exist: in-house development and outsourcing. The benefits for using a ready-made toolkit for implementing IPSEC are manifold, including radical reductions development time, guarantees of critical interoperability, reduced effort and cost of maintaining software, and possibilities to add a rich variety of security and management features. Not to mention the critical, military strength cryptographic algorithms that are not hindered by export regulations and which are coded by top engineers in the field.

However, while the advantages are many, the decision makers are still being puzzled by the key questions: If the toolkit does reduce our efforts, then how much? How much can I, and more importantly, how much do I have to modify the toolkit to get my solution out on the market? When can I expect to do my product launch and where do I set my release dates?

This white paper outlines the process of integrating the SSH IPSEC Express toolkit into the customer's product. Its purpose it to answer the critical questions of product development with SSH IPSEC Express and to help in the decision making process.

Depending on the type of application where SSH IPSEC Express is used, the requirements for integration differ. For VPN vendors that want to enter the market fast with simple black-box applications, the SSH IPSEC Express provides a fast, turn-key solution. These readers are advised to take a look at the chapter Integration of the SSH IPSEC Express into Black-Box VPNs, where such issues are covered. VPN and firewall vendors may also integrate the toolkit into their existing products and enhance them, or alternatively to provide high security desktop applications. For this purpose SSH IPSEC Express has support for the most important platforms and has APIs for making customized management applications. Chapter Integrating to Customer's Application on a Supported Platform gives more information about this task. Finally, the most comprehensive task is for those vendors that use a proprietary platform, for example, in routers. The SSH IPSEC Express is designed to be easily portable and to have clean interfaces to the surrounding operating environment to facilitate easy integration into such products. Chapter Integration of the SSH IPSEC Express into New Platforms further explores this task.

This white paper gives a generic overview of the customer integration process required. For more detailed information please see the integration manual that is included in the SSH IPSEC Express and SSH IKE (previously called as ISAKMP/Oakley) toolkits.

SSH IPSEC Express Integration

Integration of the SSH IPSEC Express into Black-Box VPNs

The idea of a black-box VPN is to have encrypting gateways in between trusted internal networks and untrusted public networks. The gateways encrypt the data flowing to the untrusted network and decrypt the data coming to the internal networks.

VPN boxes can basically range anywhere from ordinary desktop PCs to highly secure, tamper proof boxes. The VPN boxes themselves are typically stored in secure locations and managed remotely through a secured connection. Perhaps the simplest way for using SSH IPSEC Express in VPN products is to use it in such a solution.

In case the VPN device is stored in a secure location, no specific tamper proof device is needed. An ordinary desktop PC is enough. The SSH IPSEC Express can be run with NetBSD operating system out of the box, and configured to work as a security gateway. NetBSD is ideal for such a purpose as it is available in source code so that it can be stripped down from all unnecessary network services that may constitute a security risk. It is also so small in size that it fits on a single floppy, making it possible for using the VPNs without a hard disk or to operate it from ROM. The VPN devices can be configured by using a configuration file (typically there are only a handful of branch offices that need to communicate so static, manual configuration is sufficient), or by using the configuration and policy API of the toolkit.

Alternatively, the vendor can integrate the IPSEC functionality into Windows NT machines by using the Windows NT version of the SSH IPSEC Express. This is especially suitable in case a graphical GUI is desired, as sophisticated tools for making such can easily be found.

Therefore, the SSH IPSEC Express provides a fast and easy turn-key package for vendors who want to enter the VPN market. All necessary protocol code, application code, and management code can be found. The integration process in this case consists of the following steps:

1. Taking the ready-made NetBSD source code out of the box and compiling the VPN code
2. Creating a customized management and configuration program or interface with vendor logos and other graphical material
3. Copying the VPN code to product floppies or alternatively making an installation program for users to configure and write floppies themselves. Still, as an alternative, the vendor may buy the actual PCs or similar computers and pre-install the IPSEC in them to create ready-made VPN boxes.
4. Selling product floppies or CDs or VPN boxes with accompanying product documentation

Technically, the total time required for product launch is a couple of days at most, depending on how much customization the vendor wants to do.

Integrating to Customer's Application on a Supported Platform

A common option for product integration involves making IPSEC enabled security applications. Examples include IPSEC clients that run on the desktop PCs of the users themselves, or VPN applications that are installed on corporate servers. Such servers may perform also other tasks like acting as a firewall or taking care of e-mail service.

For this purpose, SSH IPSEC Express supports most common desktop platforms. At the time of writing, support for Windows NT 4 (service pack 3), Windows 95, Solaris 2.6, and NetBSD 1.3 is immediately available from SSH. Support for Windows 98, Solaris 2.51, Linux 2.0 and 2.2, and FreeBSD 3.0 and 3.1 is being implemented. As the operating system is already supported by the toolkit, no porting effort should be necessary to get the basic toolkit working.

In this scenario, the following tasks need to be performed:

1. Taking the ready made SSH IPSEC Express NT code and getting familiar with it.
2. In case a management application is desired, customizing the management and configuration application. For desktop PC users such an application is not necessarily needed or even desired as corporate security administrators may want to manage the security policy centrally. In this case the policy is simply stored on the PC. Configuration can also be done during program setup.
3. Compiling the code and making the required setup and installation programs. SSH provides ready-made examples of setups to make the task easy and painless. These can be customized to reflect the product features and vendor image.
4. Making product floppies or CDs and accompanying material such as user manuals.
5. Selling product floppies or CDs.

In case the SSH IPSEC Express is integrated to an existing security application such as a firewall, the vendor may need to fit the packet processing of the firewall with the IPSEC. The firewall typically already intercepts packets for processing and therefore the native packet intercepting capabilities of SSH IPSEC Express are not needed. In this case the vendor simply needs to forward the packets to IPSEC processing by using a simple API. A firewall also typically provides a policy management GUI. To make the approach consistent, the vendor probably wants to integrate that policy management with SSH IPSEC Express policy management by using the policy API.

If the operating system and hardware platform are already supported, integration of the IPSEC toolkit into a customer application will take anything from a few days to a few months, depending on how much customization is required, how well the people doing the integration are familiar with IPSEC, how management is going to be handled, and generally how experienced the developers are.

Integration of the SSH IPSEC Express into New Platforms

Perhaps the most demanding task in the SSH IPSEC Express integration is doing platform support for new platforms, such as a proprietary platform used in routers. These are typically optimized for the primary task, e.g. high speed routing, and have non-standard solutions in them. Requirements for the speed of encryption and the size of the code are high.

SSH Communications Security has put much effort in making the process as easy as possible, despite the high requirements. The whole architecture has been designed from

the beginning to be as portable as possible and to include a minimal amount of platform specific parts. Most of the code is very portable, and in many cases porting a library will involve nothing more than just compiling the code, or possibly modifying Makefiles or project files. The code should compile almost without modifications on most Unix platforms.

The main machine dependent module is what we call the packet interceptor. The packet interceptor is the part of the implementation that attaches into the actual TCP/IP stack. Because the TCP/IP stacks differ in different platforms, the module for "hooking" into the stack is the most common module that needs to be modified. The packet interceptor is the module responsible for intercepting packets from the native TCP/IP stack and forwarding them to IPSEC processing and back. Alternatively, in case TCP/IP source code is available, the SSH IPSEC can be placed in between TCP and IP layers, thus removing the need for the packet interceptor.

The functionality and interfaces that need to be provided by the packet interceptor are well documented in the product documentation.

Getting Prepared for Porting

Integration into new platforms typically consists of the following steps:

1. getting familiar with the source tree and documentation
2. choosing the generic architecture; kernel or user mode alternatives
3. porting the utility library
4. porting the math library
5. porting the cryptographic library
6. porting the certificate library
7. porting the IKE (ISAKMP) library
8. porting the configuration library
9. porting generic IPSEC code
10. writing a packet interceptor module
11. porting the IPSEC applications, and testing that they basically work
12. writing glue code for management and configuration

These steps are explained in more detail in the following sections. Porting the IKE and certificate libraries are described in separate sections as these are sold as independent products also.

Going through all the steps is a relatively demanding task, which requires a developer with experience in kernel-level programming, fairly good understanding of processor and operating system architecture, and good debuggers and other development tools.

The effort needed to port the SSH IPSEC Express to a new platform varies greatly depending on the platform, the goals, and the experience of the people doing the port. Early experience suggests that an experienced developer with no prior experience with the SSH IPSEC toolkit can do the port and get to an alpha-level product in two to three months. Integration into the management system is on top of this. Additional testing (estimated two more months for both a QA engineer and the main developer) is required

to make the product reliable. It should be noted, however, that these figures are preliminary, and the mileage depends a lot on the people involved and the complexity of the target platform and application.

Architecture

The SSH IPSEC Packet Processing Engine is intended to be implemented at the operating system kernel address space for best performance. It is linked together with the packet interceptor. Together they form a single module that is loaded into the kernel, as a device driver. This module includes functionality to intercept the packets from the host TCP/IP stack, and to communicate with security policy and key management. The later is typically done using a pseudo device driver.

The normal packet processing path is all in the OS kernel. The packet is intercepted, the filter code is executed against the packet possibly causing security transformation to be applied, and finally the packet is returned to the host stack.

The security policy and key management usually happens in the user address space at the SSH IPSEC Policy Manager. This program is responsible for making policy decisions for packets that the packet processing engine does not know how to process. It will take care of consulting with the application dependent security policy database, the IKE key exchanges, and about instructing the packet processing engine about the result of these.

Porting and Testing the Utility Library

To make the porting as clear and easy as possible, most of the machine-dependent code resides in a single library. This is the SSH Utility Library. The code in the library is used to implement a set of machine-independent abstractions. It implements the callback-based (event-based) programming model used in the rest of SSH code.

The requirements for the underlying operating system are as minimal as possible. Basically, what is required is that the system can send and receive UDP datagrams, there is a way to get routing information, and that there is way to communicate between user and kernel level processes (if such exist). Different ready-made implementations of these exist in the toolkit.

All system header files required by portable code are included in one single file that may need to be modified for each platform.

The event loop implements the event-driven programming model, and translates events into callbacks. The event loop also takes care of synchronization; it must guarantee that only one callback is being executed concurrently. It also implements timeouts, TCP/IP, and UDP communications. In addition, interprocess communication mechanisms used between the SSH IPSEC Engine, policy manager, and IKE (ISAKMP/Oakley) key manager need to be considered when implementing the event loop. In many cases, the event loop implements a generic mechanism (such as a select loop under Unix) which is then used to implement TCP, UDP, and interprocess communications.

The SSH IPSEC Express allocates all memory using the `malloc` and `free` functions. No porting is needed if these can be used; however, if a different memory allocation mechanism is desired, this can be accomplished by modifying one single file. The toolkit also includes replacement files for common Ansi C functions. In case the

corresponding function is missing on the target platform, these files should be compiled instead.

Porting and Testing the Math Library

SSH IPSEC Express uses its own math library for maximum speed. It also eliminates licensing issues related to such libraries as GMP. All code for the math library is in the same directory and should be compilable completely as such.

The toolkit includes assembler optimizations for Intel 386/486/Pentium processors. Implementing assembler optimizations for other platforms is up to the integrator (or can be contracted from SSH upon separate agreement). Assembler optimizations can have a very significant impact on the performance of public key cryptography and Diffie-Hellman key exchanges on some machines. The whole SSH IPSEC Express toolkit can be ported of course without the optimizations.

Porting and Testing the Cryptographic Library

Most of the SSH cryptographic library is portable C code. The C files should compile on most platforms without any modification.

The cryptographic library also supports assembler optimizations for Intel 386/486/Pentium processors. Implementing assembler optimizations for other platforms is up to the integrator (or can be contracted from SSH upon separate agreement).

Writing a Packet Interceptor for SSH IPSEC

The packet interceptor can be placed either between the network adapters and the IP protocol, inside the IP protocol, or between the IP protocol and the transport protocols such as UDP or TCP.

The location of the packet interceptor in between the network adaptor and the IP corresponds to the NDIS interface in Windows, and STREAMS interface for network adapters in SUN Solaris. Correspondingly the packet interceptor is implemented as NDIS driver for Windows platforms. When interceptor is located this way, the packet processing engine expects to receive packets that have media headers present, and it will require access to ARP packets (or alternatively access to system ARP tables).

If the interceptor is located inside the IP stack, or between the IP and the transport protocol, the interceptor should receive plain IP packets, and send such to the packet processing engine. In this case, the packet processing engine will expect the functionality behind the interceptor to take care of the media processing and packet fragmentation/reassembly.

There are other possible locations, such as hooks for firewall applications, or the IPSEC can be build into the host stack, if the source code for such is available.

The packet interceptor contains following parts:

- Interface to the network adaptors. This will typically look like a protocol to the network adapter. The packet interceptor may connect to multiple network adaptors (some of which may come up and down dynamically, as in case of PPP).

- Interface to the protocols. This will typically look like network adapter to the protocols. The interceptor typically connects as many times to each protocol as it has underlying network interfaces.
- Functionality to startup the interceptor, e.g. bind it to the adapters and protocols, create device entry points into the file system, and code to retrieve information about the existing network adapters and routing information. Typically this functionality will also start up the packet processing engine.
- Functionality that implements the defined interceptor API. The packet processing engine uses these API calls to access the underlying packet structure, and the interceptor uses it to pass the packet it has received from the adapter or protocols to the packet processing engine.
- Code that hides platform specific memory allocation routines behind ssh_xmalloc API.

Porting the SSH IPSEC Policy Manager

Normally there is only one user mode application at the SSH IPSEC, namely the policy manager. This application is intended to be run as a non interactive daemon process that is started at the system startup, and keeps on running in the background.

In embedded systems this does not have to run at the user-mode, as long as it can run independently from the packet processing engine. The policy manager makes public key cryptographic operations and key exchanges that may take a long time. One should make sure these operations do not delay the packet processing more than necessary. This may require splitting the public key operations into smaller parts.

The most important decision concerning the policy manager is where the configuration data is stored and how it is accessed. The default is to store configuration in a text file that is read in when changes happen. Other approaches would be to store the configuration into Windows registry, or at the LDAP server as signed policy blocks.

It is up the policy manager main program, and the implementation of the security policy API to arrange this storage and to reply the policy managers policy requests.

SSH IKE (ISAKMP/Oakley) Integration

The SSH IKE (ISAKMP/Oakley) is the key management part of the SSH IPSEC Express. It is also sold as a separate key management solution, e.g. for vendors who have already built their own IPSEC or have outsourced it from someone else than SSH.

The integration work typically consists of two separate tasks

1. Writing glue code between SSH IKE (ISAKMP/Oakley) and third party IPSEC, and
2. Porting the SSH IKE (ISAKMP/Oakley) code

Integrating SSH IKE with Third Party IPSEC Implementations

This task is not required when SSH IPSEC and Policy Manager are used.

The IKE library has three interfaces to the external world (in addition to using the SSH cryptographic and utility libraries):

- certificate management interface
- initiator policy functions
- responder policy functions.

There is no need to touch the certificate management functions when using the SSH IKE with the accompanying certificate library. However, when integrating to a third party policy manager and IPSEC implementation, the initiator and responder policy functions need to be rewritten.

The initiator and responder policy functions make policy decisions in initiator and responder mode, respectively. They also communicate information about new security associations with the underlying IPSEC and policy management implementation. Depending on the third party IPSEC, the task may take from a few days to a couple of weeks.

Porting the SSH IKE Code

The SSH IKE (ISAKMP/Oakley) library consists of portable C-code. The library should not include any platform-specific code, but new Makefiles or project files may need to be written in some environments.

SSH IKE uses functions from the Utility, Math, and Cryptographic libraries. For information about porting them, see chapters Porting and Testing the Utility Library, Porting and Testing the Math Library, and Porting and Testing the Cryptographic Library of this white paper.

SSH X.509 Certificate Tools Integration

The SSH X.509 Certificate Tools forms the interface between a management system and Public Key Infrastructures (PKIs). It can be used e.g. to retrieve certificates from a public Certification Authority (CA) as a part of SSH IPSEC Express and SSH IKE products or as a separate certificate management product.

The SSH X.509 Certificate Tools has a simple interface for doing the required tasks of certificate management. The toolkit includes example code and programs that can be used as such for creating and fetching certificates or as a basis for building a whole certificate enrollment system.

The SSH certificate library is entirely portable Ansi-C code. New Makefiles or project files may need to be written in some environments. The library also requires code from Utility, Math, and Cryptographic Libraries.

Writing Management and Configuration Applications

In many cases, the only task left for the developers is writing a customized management application. Typically vendors want to include such functionality into the general management mechanisms of the their applications. The SSH IPSEC Express provides all

the necessary functionality to support even the most demanding security policies. The SSH IPSEC Express toolkit provides function calls that can be used to supply new configuration information on the fly.

The configuration information should be generated by the management application. The management application will then communicate with the IPSEC engine using an application-specific mechanism, such as a named pipe or an authenticated TCP/IP connection. The SSH Policy Manager listens for configuration updates from the management application, and calls the appropriate functions whenever new configuration information is received.

Summary

There are many different options for the vendor in integrating the SSH IPSEC Express toolkit into a security solution. These range from a vary straightforward use of existing product possibilities, to a large scale integration into new product and product lines. Depending on the requirements, the total workload and time required may range from a few hours to a few months. Whatever the case, the SSH IPSEC Express provides both flexibility and readily set architectural solutions to fulfill the various vendor requirements.