The following paper was originally published in the

Proceedings of the 8th USENIX Security Symposium

Washington, D.C., USA, August 23–26, 1999

VAULTED VPN: COMPARTMENTED VIRTUAL PRIVATE NETWORKS ON TRUSTED OPERATING SYSTEMS

Tse-Huong Choo

Vaulted VPN:
Compartmented Virtual Private Networks
On Trusted Operating Systems

Tse-Huong Choo
Hewlett-Packard Laboratories, Bristol,
United Kingdom

Abstract

Virtual Private Networks for IPSec based on an intermediate packet-redirector in network-protocol stacks are becoming increasingly common for many standard operating systems and represent a well-understood method for retro-fitting such systems with IPSec support. This report describes how a different design structured around a Trusted Operating System can offer better security, performance and robustness. We describe in detail an implementation of an IPSec VPN consisting of a series of compartmented, concurrently executing IPSec stacks. The motivations and security-related benefits behind each design decision are discussed. In addition, we show how a configuration of independent IPSec stacks based on this design can be configured to execute in parallel for greater performance on single-threaded kernels, and how its design allows individual component-failures without affecting the system as a whole.

1 Introduction

There is an increasing prevalence of IPSec Virtual Private Networks (VPNs) based around an approach of inserting a plug-in into the network protocol stack of mainstream operating-systems. This plug-in typically redirects any incoming or outgoing network packet in order to perform IPSec-specific processing such as encryption or decryption. This approach is well understood and has been mentioned in many standard texts, most notably the IETF RFCs for IPSec [2][3].

However, this approach is not without its weaknesses. The important, but often conflicting design goals of security, performance and robustness often leads to cases where the development of one goal requires trade-offs in the other. For example, placing IPSec-processing in the kernel or operating system level is desirable performance-wise because of copy-avoidance of packets between kernel and user-mode spaces. However, the kernel or operating system can be compromised if an error occurs in the network protocol stack.

This report attempts to describe another approach that seeks to avoid some of these issues by compartmentalizing the components of a traditional VPN into isolated, independent processes. In particular, we aim to show how a more secure version of a conventional VPN can be built to make use of the security-features present in a trusted operating system and also satisfy the goals of high-performance and robustness.

The techniques used in designing a more secure VPN will typically introduce performance degradations that are otherwise not present on a standard VPN-design. For example, the general principle of reducing a monolithic system to a series of smaller, trusted components involves added overhead in synchronization between these components. Therefore, the issues of security and performance have to be considered as inextricably linked. These two issues, together with robustness, is fundamental to many of the topics presented here.

This report describes the design and operation of the Vaulted VPN, which is a software-based IPSec VPN built on a slightly modified version of the HP-UX 10.24 Virtual Vault Operating System. We begin by describing briefly the security-features of HP-UX 10.24 relevant to the Vaulted VPN and continue by describing the architecture and operation of the Vaulted VPN, followed by a security analysis of each aspect of VPN's operation.

We also show how compartmented VPNs such as the Vaulted VPN can take advantage of parallel processing and how they can continue to function even if some portions fail – not an unimaginable occurrence given the many well-documented attacks using malformed IP packets [19] [20]. This represents an unexpected level of robustness considering the monolithic nature of most network protocol stacks.

Although this report mentions the use of HP-UX 10.24 specifically as a target platform, the principles shown here could be applied to full-blown CMW platforms [9] such as HP-UX 10.26 or Sun Microsystems' Trusted Solaris 2.5.

2 Features of the HP-UX 10.24 Virtual Vault Operating System

The target platform of the Vaulted VPN is the HP-UX 10.24 Virtual Vault Operating System. The HP-UX 10.24 operating system is a UNIX-style Multi-Level Secure (MLS) operating system that implements additional security-features beyond those normally available on standard UNIX systems. It is a CMW-derived operating system [6] [9] and features the Mandatory Access Controls (MAC) and principles of least-privilege found on full CMW-type systems, but differs in that it lacks the Trusted Windowing and Trusted Networking features of a full CMW-type system. This section describes the security-features used by the Vaulted VPN.

2.1 Mandatory Access Control

In addition to the standard UNIX-style security features, HP-UX 10.24 also implements a Mandatory Access Control policy based on the Bell-LaPadula formal model [11]. Mandatory Access Control (MAC) is a set of rules that govern how data may be accessed on the trusted system. The policy is called mandatory because users cannot choose which data will be regulated under this policy – the trusted system enforces MAC consistently. In this model, all resources or objects in the system are given sensitivity labels, which consists of a classification and a number of compartments. Access to an object is controlled by its sensitivity label. To have read-access, a process must have a sensitivity label which 'dominates' that of the object being read. For write-access, the labels must match exactly.

2.2 Discretionary Access Control

HP-UX 10.24 has discretionary access controls (DAC) which are similar to those found on standard UNIX systems, specifically the read-write-execute mode-bits based on user and group identities. It is most effective when used in conjunction with Mandatory Access Control, for example, when used to restrict access to everyone but a special pseudo-user used by specific trusted processes.

2.3 Least Privilege

On a traditional UNIX system, the ability to override security controls rests entirely within the root-account. On trusted systems, this all-powerful ability has been broken down into a large number of individual privileges, each of which confers its holder the ability to perform a specific action. Every action that could affect security is restricted with a named privilege, and only processes with the appropriate privileges are allowed to perform these actions. Following the principle of least privilege, this finer level of granularity allows specific programs to be assigned with the minimum-required set of privileges in order for them to perform their assigned tasks.

3 IPSec Protocol

A significant fraction of commercial VPNs are based on IPSec technology which defines the encoding of the encrypted IP packets. Normal unencrypted IP packets are made up of an IP-header followed by a variable length payload in clear-text which may itself include headers for higher-level protocols, e.g. TCP or UDP.

IPSec packets appear to be normal IP packets to switching and routing hardware, but feature internal payloads which may be encrypted, authenticated or both. There are two formats used: the AH format [5] which allows data to be authenticated using a keyed hash-function, and the ESP format [4] in which data is encrypted and optionally authenticated, depending on which encryption algorithm is selected.

In addition, IPSec allows for two main modes of operation, namely transport and tunnel modes. Tunnel-mode allows entire IP packets of arbitrary protocols to be encapsulated within another IP header to allow for authentication of the original IP packet's header, and is typically used in creating an encrypted tunnel between gateways.

The IPSec protocol may be imagined to be operate at a level below or at the IP-layer of network protocol stacks. The secret keys used to encrypt the payloads are derived earlier using some higher key-agreement protocol (such as IKE [1]) or simply manually keyed.

4 Vaulted VPN Architecture

This section gives an overview of the architecture of the Vaulted VPN by describing the purpose and function of each of its components. An explanation is given on how packets are routed in and out of the kernel, where IPSec processing takes place and how Internet Key Exchange (IKE) [1] negotiations are triggered to form IPSec Security-Associations.

4.1 Design of Conventional Bump-in-the-stack VPNs

First, we consider the design of a conventional bump-in-the-stack VPN. Because the kernel on most operating systems is usually not re-entrant and not directly accessible by user-processes, it is usually designed in a monolithic fashion. The network protocol stack in such kernels is similar – packets flowing up or down the stack from various processes (some sensitive, some not) all go through the same monolithic stack. Therefore, a bump-in-the-stack VPN based on such systems would naturally share the problems of re-entrancy and lack of segmentation.

However, the nature of the HP-UX 10.24 operating system allows a finer level of granularity when handling network packets. Since HP-UX 10.24 applies sensitivity-labels to most objects and resources in the system (including files, processes and network packets), it becomes possible to segment the data present in such a system based on its sensitivity label. Since each process is placed at a given sensitivity level, and access to objects at different sensitivity-levels is controlled strictly by the operating system using MAC controls - one can imagine that a process at a given sensitivity-level to be in a separate ‘compartment’ from other processes of differing levels. The term ‘compartment’ is used loosely here to refer to the segmentation that arises as a result of applying MAC controls to labeled objects, and does not refer explicitly to the term used in defining sensitivity-labels. This usage is perhaps closer to what one might intuitively expect when reasoning about isolated groups of processes at different sensitivity levels.

This feature of segmenting data into different sensitivity levels (referred to from now on as compartments) can be used to isolate different processes from each other. For example, a sensitive private-key management agent could be placed in a separate compartment by itself to isolate it from other less sensitive processes.

4.2 Vaulted VPN: Compartmented Stacks

The principle of compartmentalization mentioned earlier yields an architecture for the Vaulted VPN where the processing of IPSec packets is broken into separate user-processes, each running in a separate compartment and handling network packets belonging to the same compartment as themselves.

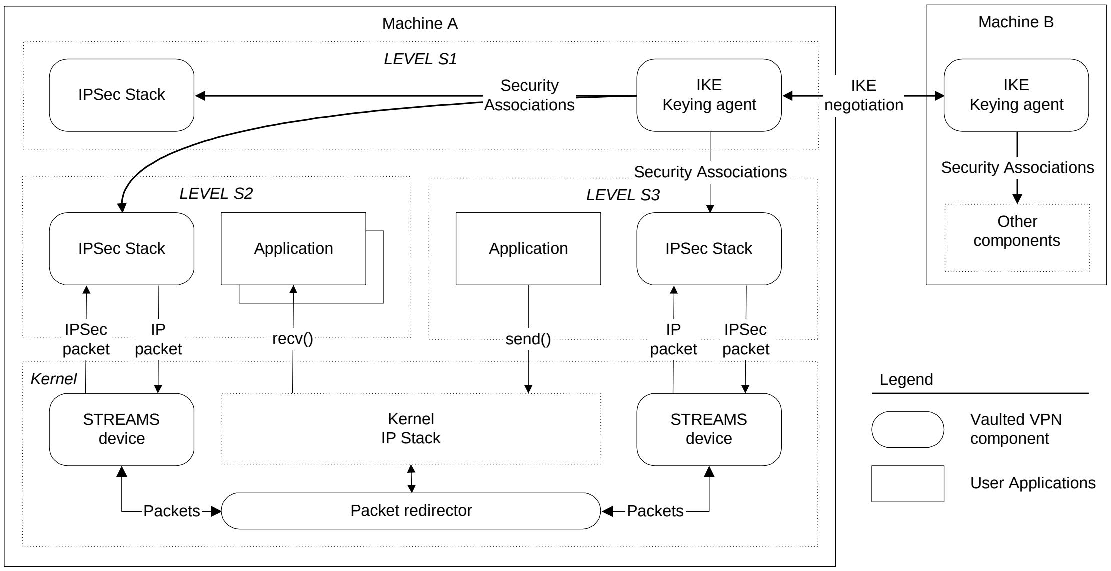

Figure 1 Vaulted VPN architecture showing stacks at different sensitivity levels

The traditional monolithic kernel-level IPSec stack has been changed instead to a series of compartmented, concurrent user-mode processes. In order to make possible the processing of network packets in compartmented user-mode processes, the target platform of the Vaulted VPN is a version of HP-UX 10.24 that has been modified to perform packet redirection by means of a simple kernel-level add-on. The kernel level add-on is necessary as this particular version of the HP-UX operating system uses a BSD-style [10] IP stack that needs to be modified to include packet redirectors.

This architecture (see Figure 1) of the Vaulted VPN consists of:

1. a kernel-level add-on for packet redirection from the kernel to a STREAMS [14] device

2. a series of concurrently executing user-mode IPSec stack-processes running in different compartments. These processes intercept redirected packets from the kernel and perform IP fragment re-assembly and the actual encryption and decryption of IPSec packets [4] [5].

3. a user-mode IKE [1] keying-agent, which performs negotiations with its remote peers in order to derive keying-material in the form of IPSec security-associations for use in encryption and decryption of IPSec packets. The IKE keying-agent performs the negotiations needed to derive the keying-material used in IPSec processing. This keying-material for IPSec security-associations is sent to the various IPSec stacks using a protected message-queue described later.

4.3 Packet Flow

In the Virtual Vault operating system, each packet travelling up or down the network stack is labeled with a sensitivity-label. The kernel has been slightly modified to intercept all incoming and outgoing packets and pass them to a STREAMS-based driver to be queued for further processing. Once queued at the STREAMS-driver, the packets will eventually be read by a user-mode IPSec stack and be processed there.

Each IPSec stack is a separate user process linked to the kernel IP-stack via a STREAMS-based driver. Typically, there is a separate stack for each sensitivity level on the system. Upon startup, the IPSec stacks send a message to the STREAMS-based driver in the kernel to indicate the level that the stack is at. This mechanism of run-time configuration can be abused and many ways exist to counteract this. As an example, one could check the system-supplied user-credentials in the STREAMS-based driver whenever a process attempts to open the driver.

Using this message, the kernel demultiplexes the stream of labeled IP packets to the various user-mode IPSec stacks depending on the value of the label registered previously. This process is purely a simple comparison - the label of each packet has to match the label of a previously registered IPSec stack before the packet can be routed to it. Duplicate IPSec stacks attempting to register using the same label are ignored.

The ability of HP-UX 10.24 to label network packets within the kernel allows the Vaulted VPN to distinguish between packets that are destined for different compartments and to route the packets accordingly. Incoming packets are labeled with the sensitivity-label of the network-interface from which the packet originated. Outgoing packets obtain their labels from the security-attributes of the socket that generated the packet. Because all network packets are labeled internally within the kernel, it is possible to redirect these packets to separate compartments based on their labels and operate on a compartmented basis.

This implies a use of MAC controls that has not yet been described fully. The notion of handling and routing packets on a compartmented basis is a logical extension of the segmented nature of trusted operating systems. Where packets within the kernel used to be processed together regardless of their labels, there now exists some form of segmentation within the network protocol stack of the operating system. This notion can be extended to many other system components, but this report does not explore it any further.

4.4 IPSec Stack - Basic Operation

Each IPSec stack imposes a security-filter on the incoming and outgoing packets. Each time a packet is received from the kernel, the security policy database [2] internal to the IPSec stack is consulted to see if the packet should be processed, dropped or bypass IPSec processing. If no matching entry is found in the policy database, the packet is dropped.

If the policy database specifies that an incoming packet requires IPSec processing, the packet is then inspected to see if IPSec decryption can take place based on the security-descriptors present in the packet. If the decryption processing completes successfully, the decoded packet is sent back to the kernel via the STREAMS-driver to travel up the network protocol stack, usually to a socket's receive-buffer.

For outgoing packets, the security policy database is consulted to see if traffic to the destination specified by the outgoing packet is allowed, and if so, what kind of processing to apply to it. If the outgoing packet is to be IPSec-encrypted, then the relevant encryption-processing is applied and the encrypted packet sent back to the kernel to be subsequently sent out the network interface. Packets banned from transmission by the security policy database are dropped.

4.5 Message Interfaces

The individual IPSec stacks use standard UNIX message-queues to communicate with the IKE keying-agent. These queues are used to trigger IKE negotiations and to pass keying-material from the IKE keying-agent to the individual IPSec stacks. The IPSec stacks also use a kernel-to-user-mode interface in the form of a STREAMS-based driver to send and receive IP packets from the kernel. These interfaces are protected using methods described later.

5 Compartmented Keying Material

Negotiations to form a new IPSec security-association are normally triggered by an outgoing packet that does not have a matching entry in the security-policy database. In such a case, a message is sent to the IKE keying-agent to trigger a new IKE negotiation, which occurs asynchronously to the processing in the IPSec stacks. The packet in question is queued within the IPSec stack in anticipation of a successful IKE negotiation. Once the IPSec security-association is formed, the queued packets are released to be processed internally by the IPSec stack.

The network packets from IKE negotiations are redirected by the kernel to the IPSec stacks, as it would do for any normal application. However, the Vaulted VPN is configured to behave as a pass-through for IKE packets, so that no special IPSec encryption is performed.

5.1 Labeling of IPSec keying material

The keying material derived as a result of an IKE negotiations is labeled according to the sensitivity-level of the outgoing packet that originally triggered the IKE negotiation. Figure 2 shows how sensitivity-labels of network packets flow through the Vaulted VPN and how they are used to determine the label of derived keying material.

Each IP packet inherits its sensitivity label from the socket that generated it and this label travels with the packet until it is intercepted by the IPSec stack and subsequently used to tag subsequent IKE negotiations.

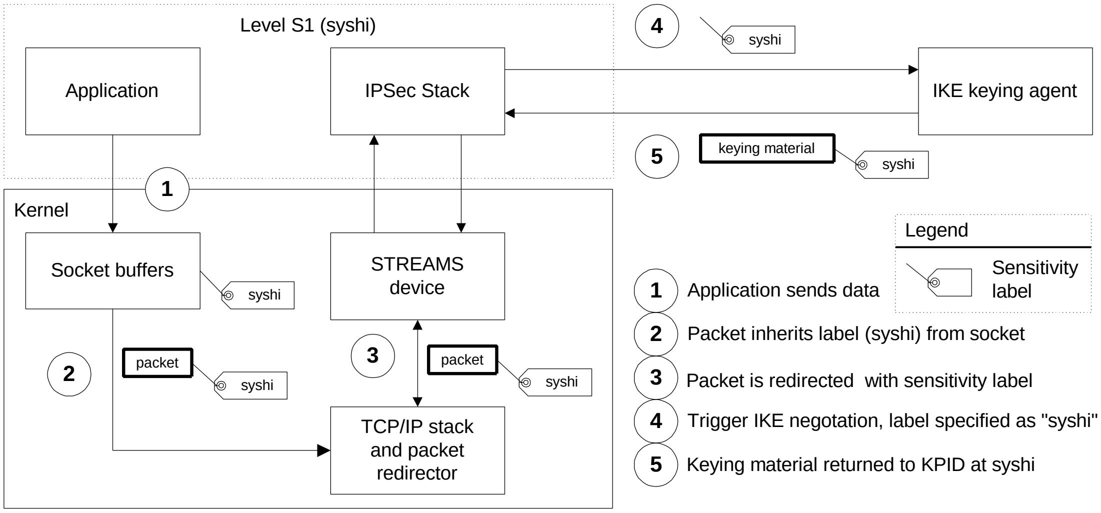
Level S1 (syshi)
Application
IPSec Stack
IKE keying agent
4
syshi
5
keying material
syshi
1
Kernel
Socket buffers
syshi
STREAMS device
2
packet
syshi
3
packet
syshi
TCP/IP stack and packet redirector
Legend
Sensitivity label
1 Application sends data
2 Packet inherits label (syshi) from socket
3 Packet is redirected with sensitivity label
4 Trigger IKE negotation, label specified as "syshi"
5 Keying material returned to KPID at syshi

Figure 2 Determination of sensitivity-level of IPSec keying material

5.2 Labeled Distribution of IPSec keying material

When IKE negotiations are successful, the keying-material derived in the form of IPSec security associations is sent via a private key-channel to the IPSec stack at the same sensitivity-level. The IKE keying-agent does not hold IPSec keying-material beyond the short duration required to derive it and pass it on to the individual stacks.

The net result is that keying-material is segmented according to its sensitivity-label (see Figure 3). The keying-material in any given stack is never duplicated in another, leading to cleanly segmented, non-intersecting subsets of the original keying-material.

5.3 Storage of non-volatile IKE private keys

Private keys used by the IKE keying-agent to digitally sign messages in IKE main-mode negotiations are stored as files and are protected by setting them to the highest sensitivity level, *syshi*. Like the IKE agent, a dedicated compartment would be preferable, but *syshi* is chosen given the static configuration of compartments in HP-UX 10.24. The appropriate permissions-bits are also used to disallow access by everyone else but the VPN pseudo-user. Further measures, including encrypting the keys with a passphrase that is supplied when the Vaulted VPN is started, could be employed but has not been implemented in the current version of the Vaulted VPN.

6 Security Analysis

This section presents a security analysis of various aspects of the operation of the Vaulted VPN. We show how the underlying security-features of a Trusted Operating System are used to isolate and protect the various components of the Vaulted VPN. This is done by examining the system from several aspects – the component processes, the messaging-interfaces between the components and the keying-material travelling between the components. Taken together, this view of security based on the components and their interfaces forms the basis of the security of the Vaulted VPN.

6.1 General security measures

- Each component is run with an effective user-id of a VPN pseudo-user. This user is specifically created for use by Vaulted VPN components only and cannot logon to the system under normal circumstances. This arrangement allows the use of UNIX-style permission-bits to disallow access by non-Vaulted VPN components. The use of system provided file-control databases ensures that only the Vaulted VPN components are assigned this pseudo-id, and allows the use of DAC controls as a more powerful discriminant that would otherwise be possible on standard UNIX systems.

- The *chroot* UNIX-command is used to place each executable component into a restricted filesystem to limit the scope of accessible system files should a security breach occur.

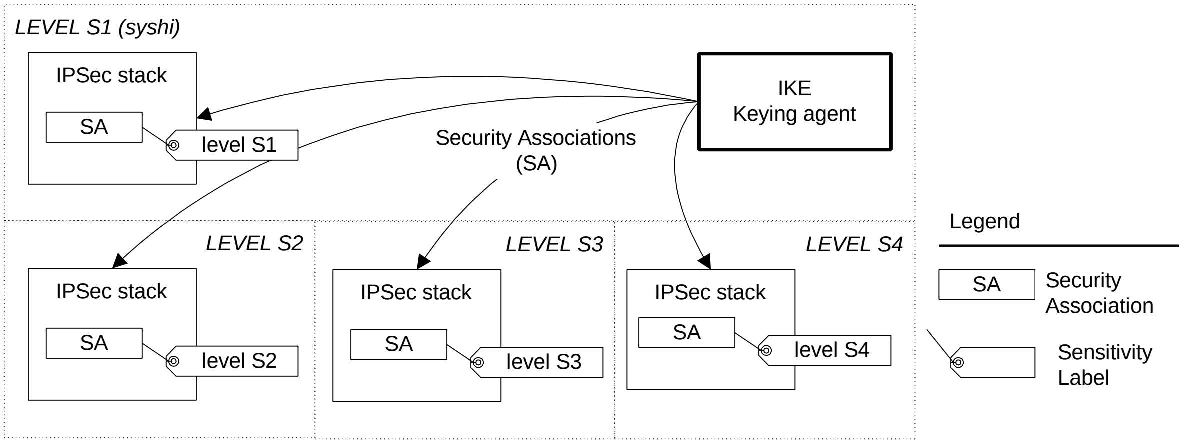

Figure 3 Distribution of labeled IPSec Keying Material

- In other ‘full-blown’ CMW systems, it is possible to place each component in a separate compartment of its own. However, HP-UX 10.24 offers a static configuration of compartments. Therefore, the safest choice is to place the sensitive components (like the IKE agent) at the highest sensitivity level possible, assuming that this level is not used by very many other processes in a properly configured system.

As a general rule, the IKE keying-agent is placed into the highest-classification possible to isolate it from other processes in compartments of a lower classification. The IKE agent executes with enough privileges to perform certain privileged network operations (such as binding to the IKE-reserved port number), but is otherwise relatively unprivileged.

The IPSec stacks are given almost no special privileges at all because their activities consist almost solely of reading and writing to a device file. Therefore, faults in the Vaulted VPN IPSec stacks do not give root-equivalent access, compared to the situation where a breach in an IPSec stack situated in the kernel allows access to the entire operating system.

6.2 Compartmented Component Processes

The approach taken here is to compartmentalize the various component processes of the Vaulted VPN to isolate them from each other and the rest of the system as much as possible.

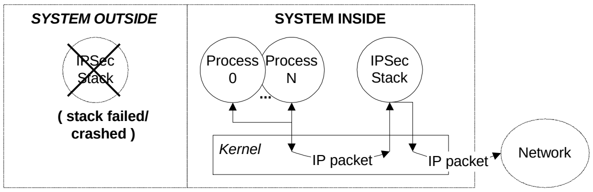

Figure 4 Crash-isolation between compartments

The individual IPSec stacks are also placed into separate compartments, each at different sensitivity levels – typically one stack per compartment. By using this approach, a fault in the IPSec stack in a given compartment does not automatically cause the processing of other stacks to fail (see Figure 4). More importantly, it does not lead to a breach in security for those other stacks.

6.3 Compartmented Keying-material

Keying-material derived from successful IKE negotiations is never kept in a single location with the IKE keying-agent. Instead, it is considered as labeled according to the sensitivity-level of the network packet that initially triggered the negotiation and once derived, is distributed to the appropriate IPSec stacks. The net effect is that the keying-material is only present in the compartment in which it is to be used. This reduces the visibility of keying-material associated with highly sensitive network connections and eliminates the mingling of keying-material found in systems where the entire IPSec stack resides in the kernel.

6.4 Protection of component interfaces

There are two main messaging interfaces in the Vaulted VPN which need to be protected:

1. The keying-channel between the IKE keying-agent and the various IPSec stacks

2. The kernel-to-user-mode STREAMS interface between the kernel IP stack and the various user-mode IPSec stacks

The keying-channel between the IKE keying-agent and the various stacks is a standard message-queue. Protection of this interface is through a combination of Mandatory and Discretionary Access Controls. The IKE keying-agent and stacks are launched under a special pseudo-user. The message-queues are then owned by this pseudo-user by virtue of inheritance and are created with permissions-bits that disallows access by everyone else. This restricts access to the queue to the Vaulted VPN components only, since they are the only components on the system that are configured to use this pseudo-user. In addition, the queue is labeled with the highest classification possible to protect it from access by processes from lower classifications.

The kernel-to-user-mode STREAMS interface used by the IPSec-stack to send and receive IP packets utilizes special device-files that also need to be protected. One device-file is created for each sensitivity level. This ensures that only one interface is accessible at each sensitivity-level. The other interfaces, which serve to handle IP packets of a different sensitivity-level are not visible (and hence, not accessible) from processes running at different sensitivity-levels.

The ownership and permissions-bits on each device-file is set to disallow access to everyone but the special VPN pseudo-user, thereby disallowing access to every other process but the IPSec stack allocated to this compartment. In addition, the kernel-level STREAMS driver checks the system-supplied user-context [14] when the device is opened in order to decide if the device is being opened by a VPN component.

6.5 Preservation of sensitivity-labels across machines

The notion of unauthenticated explicit sensitivity-labels to tag data in labeled IP traffic (e.g. IPSO/RIPSO [12][13], MaxSix [8]) has given way to cryptographic methods such a those used in IPSec. However, it would be desirable if the same net effect could be achieved within the IPSec framework. The solution that IKE provides is the ability to specify security-related attributes in an IKE quick-mode negotiation [15].

During IKE quick-mode negotiations, sensitivity-labels are passed between the negotiating IKE-peers to indicate the level of the incoming and outgoing data. This is stored for every security-association that is formed and is used by the IPSec stack to label processed packets before they move up the network stack. In this way, data that was labeled *syshi* on the originating machine is never “written down” on the receiving end, therefore preserving the sensitivity labels of data crossing the network.

This occurs without necessarily having to pass sensitivity labels in each transmitted IP packet because the sensitivity labels are negotiated first with IKE and stored in the corresponding IPSec security-association used in processing that packet. Packets processed using a particular security association are checked against the label stored with the security association and relabeled (or upgraded) as necessary.

6.6 Differences in security with conventional bump-in-the-stack VPNs

The benefits of this architecture are best understood by comparing it against the design of a conventional bump-in-the-stack IPSec VPN.

- In the Vaulted VPN, the various components and the interfaces between them are protected by the MAC and DAC controls of the Trusted Operating System. A conventional VPN running on a mainstream OS lacks the capability to compartmentalize the IPSec stack, and to isolate keying-material into separate compartments.

- In a conventional VPN, the IPSec stack is usually implemented as a STREAMS-driver or module (e.g. for Solaris) or as a kernel-level NDIS shim as in Windows NT. A fault in the stack (possibly caused by the receipt of a malformed packet) could easily lead to corruption in the kernel and lead the system to crash. In the case of the Vaulted VPN, the IPSec stack has been pulled from the kernel into user-mode and subsequently split into independent, compartmented processes. The kernel is less likely to be affected by a user-mode IPSec-stack crash.

- On a conventional UNIX system, the IKE-daemon has to run as root in order to bind its socket to privileged port number for IKE [16]. Similarly, in Windows NT, the daemon needs Administrator privileges to open a similar port. Following the principles of least-privilege, the keying-agent runs as a relatively unprivileged process. This means that breaking into the IKE keying-agent does not give root-equivalent access.

The last point mentioned above can be important as the IKE protocol is complex and IKE-implementations are still relatively new and lack sufficient code maturity to be considered stable. It is desirable that untrusted IKE daemons not be given root or Administrator access. The various IPSec stacks also execute as relatively unprivileged processes - reducing the scope of any damage from break-ins to the stack.

6.7 Other approaches

The choice of IPSec as the network protocol of the Vaulted VPN contrasts with the approach taken by the HP Praesidium Extranet VPN [17], where the SOCKSv5 protocol [18] is the choice. IPSec-based VPNs have traditionally been seen to be operating at a much lower level than SOCKS-based solutions - which operate at the level of each individual application and have access to higher-level security constructs, including those derived from application-specific domains.

This possible advantage is largely negated by the ability of the kernel-component of the Vaulted VPN to also pass fine-grained kernel security-attributes (derived from the user-processes) to the IPSec stacks. However, SOCKS-based solutions have the advantage of selectively excluding non-security sensitive applications from security-related processing, which allows for reduced overheads.

7 Extensions

This section describes extensions to the configuration of the Vaulted VPN to increase performance and robustness. We show how concurrently executing stacks can de-multiplex the stream of IP packets from the kernel in order to perform IPSec processing in parallel for greater performance. We also describe how the stacks can be dynamically configured to perform load balancing and to run in tandem to create multiple, redundant stacks.

7.1 Concurrent Execution of Stacks

Since the IPSec stacks exist as a series of independent user-mode processes, it is possible to launch as many copies of the stacks as desired. In particular, it is possible to launch two or more stacks to deal with IP packets for a particular compartment (see Figure 5). On platforms that support threads natively, it would be preferably to launch independent threads instead of processes to take advantage of the typically smaller overheads associated with context-switching amongst threads compared to processes.

This can be useful when traffic on a particular system is predominantly of a given sensitivity level. Most mainstream operating-systems feature support for SMP configurations and will distribute compute-intensive processes to idle CPUs, ensuring that speedups are close to being linear. Since the IPSec stacks operate by reading and writing packets via a special device-file tied to a STREAMS-driver [14], it becomes possible to demultiplex the stream of IP packets from the kernel evenly between any number of stacks.

This happens automatically as a result of the default processing at the STREAM-head which is supplied by the platform's STREAMS implementation. The individual stacks themselves are unaware that they are only receiving a fraction of the normal number of packets that they would have received if they were the only stack running at that level. Concurrency control is achieved purely through the operation of the STREAMS-head which is supplied by the platform's STREAMS implementation. The STREAMS-head typically operates at the granularity of individual STREAMS-messages (or IP packets in this case) and handles multiple-readers by queuing their requests internally. A later section lists some preliminary performance figures.

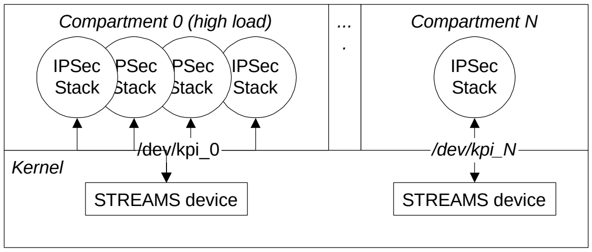

Figure 5 Configuration illustrating a variable number IPSec stacks per compartment

7.2 Multiple Redundant stacks

If it is assumed that IPSec stacks will suffer from the occasional bug, then the possibility exists of sending a corrupted packet to the kernel. To help prevent this, it is possible to implement several versions or implementations of the IPSec stack (possibly using several different languages or compilers).

Because different implementations will typically have different bugs present, the probability of any two (or more) implementations having the same bug under the same conditions is very small indeed (the probability decreases as the number of implementations increase). Therefore, if the multiple redundant implementations are used to "vote" on the output prior to it being accepted by the kernel then corrupt output can be greatly reduced or even virtually eliminated. Such a vote would have to be unanimous, although this approach would not likely solve subtle bugs, or bugs in the original protocol definition itself. As such, this extension although easy to develop, would be of rather less use than one might originally suppose.

7.3 Host-level Accreditation of Sensitivity Levels

It is possible to enforce a scheme where remote hosts are accredited with a range of sensitivity-levels of data that they are allowed to accept. This can be done by encoding the range of sensitivity-levels that a remote host is allowed to accept in extensions to X.509v3 certificates [6] and using these certificates during IKE negotiations.

Figure 6 below shows one such possible encoding for host-accreditation in an X509.v3 certificate. The actual encoding the X509.v3 extension is not described here, although choices could range from schemes using simple ASCII encoding to using table-lookups.

A check can be performed during IKE negotiations to see if the remote host in the negotiations has been accredited with receiving data of the current sensitivity-level of the packet. If the level of the packet falls outside the range specified in the certificate, the IKE negotiation can be programmed to fail. Hence, packets can only travel amongst hosts that have been properly accredited.

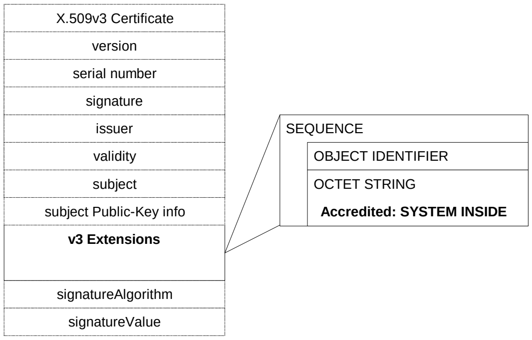

Figure 6 Placement of Host-Accreditation extension in an X509.v3 Certificate

8 Performance Measurements

As expected, the system experiences increased latency (which shows as increased times taken to connect) and lower throughput due to the longer processing pipeline introduced by user-mode stacks and overheads spent in synchronizing access to the STREAMS-device.

8.1 Web-Server Performance

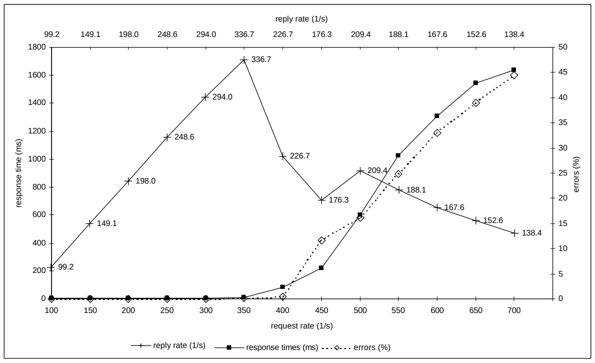

Figure 7 Performance graph-ESP with DES-CBC & HMAC-MD5

Figure 7 above shows the results of tests with *httperf* [22]. The test machine was a HP 9000/871 with 2x160Mhz PA-RISC CPUs with 256MB of RAM running the Apache 1.3.6 Web-server. The test-operation involved repeatedly retrieving a 1-kilobyte file using HTTP over a single TCP connection according to adjustable rates to show peak connection rates, connection-time latencies and errors due to timeouts.

The VPN was configured on the same machine on a 100Mbit/s link to use the IPSec ESP protocol with 8-byte DES-CBC encryption and HMAC-MD5 authentication. The table shows a peak connection rate of around 336 connections/sec after which response-times and errors (as a percentage of total connections) start to increase. This figure is well below what might be expected of this configuration which should be around 700 connections/sec.

The processing associated with DES encryption did not appear to be a significant factor as the figures recorded for a VPN operating in a 'pass-through' mode (i.e. no encryption calculations) were virtually identical as shown in Figure 8 below.

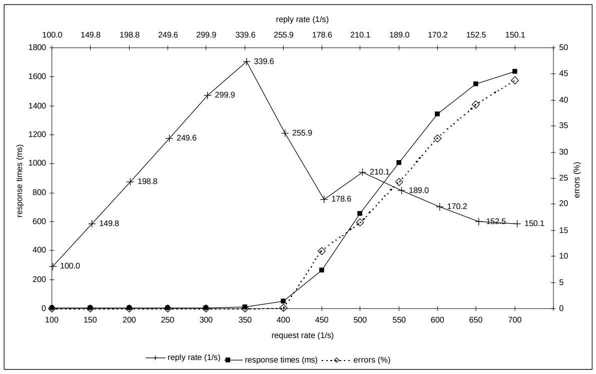

Figure 8 Performance graph – no encryption

8.2 Raw Throughput

Raw network throughput was measured with the *netperf*-tool [23] on a 100Mbit/s Ethernet link. Table 1 and Table 2 below show the results from tests performed on a dual 160Mhz CPU HP 9000/871 server with the VPN configured to use HMAC-MD5 authentication alternately with 8-byte DES-CBC or Triple-DES (3DES) with the IPSec ESP protocol. Tests were performed using both single and dual-stack configurations.

It is thought that the modest speedup on dual-stack configurations is due to the *netperf* client and server competing for CPU cycles with the IPSec stacks. Netperf was observed to consume significant amounts of CPU cycles (>40%) thus depriving the kernel and the IPSec stacks from effective use of the dual-CPU SMP configuration. It was not possible to perform these tests on SMP configurations with more than 2 CPUs due to lack of equipment.

# of stacks	**Configuration**	**Throughput (Mbits/s)**
1	DES-CBC & HMAC-MD5	10.0
2	DES-CBC & HMAC-MD5	11.5
1	3DES & HMAC-MD5	6.79
2	3DES & HMAC-MD5	9.25

Table 1 Netperf using unidirectional TCP_STREAM

# of stacks	**Configuration**	**Throughput (Mbits/s)**
1	DES-CBC & HMAC-MD5	13.1
2	DES-CBC & HMAC-MD5	13.8
1	3DES & HMAC-MD5	7.9
2	3DES & HMAC-MD5	8.0

Table 2 Netperf using unidirectional UDP_STREAM

The issue illustrated by running *netperf* in dual-stack configurations is one of resource-management. The run-time behaviour of user-mode applications impact directly on the performance and behaviour of the VPN because the system described does not offer adequate control over resource consumption, namely CPU-usage.

As noted in [21], conventional OSes (such as the system described here) currently treats individual processes as resource principals while failing to properly account for the kernel resources that they consume. The simple task of bandwidth-restriction is complicated by the need to calculate the accumulated CPU-usage of a packet as it flows from an interrupt-service routine in the kernel up to a user-mode handler and then down again. In such a scenario, one must view the IPSec stacks as extensions to the kernel and attribute the resources they consume to the originating resource principals.

To increase performance, the IPSec stacks could be restructured as a pool of kernel-level threads. However, the user-space design can be improved somewhat – perhaps by increasing the efficiency of event-driven servers (as each IPSec stack invokes *select*() repeatedly) as described in [24], or by using platform-dependent APIs which perform memory-transfers direct to-and-from user-space instead of the STREAMS interface currently being used.

9 Summary

Conventional bump-in-the-stack VPNs have well-known security hotspots such as running the IKE keying agent as root, or placing IPSec processing in the kernel which exposes the kernel to faults in the IPSec stack. A solution to some of these problems is to remap the design of a conventional IPSec VPN to a trusted operating system (such as HP-UX 10.24) to take advantage of the security-features present in such a system. Through the use of a trusted system, further refinements become possible – the IPSec stack and the associated keying-material can be compartmented to offer an even greater degree of security and robustness.

The Vaulted VPN architecture features multiple asynchronous user-mode IPSec stacks which execute concurrently and independently from each other. In HP-UX 10.24, these stacks are placed in separate compartments and handle only the IP packets that match the compartment that they're in. IKE keying-material is distributed to the relevant compartments and not kept in a single location. Sensitivity-labels of data crossing the network is preserved through the use of IKE negotiations without requiring the label to be transmitted in each encrypted IPSec packet.

The IKE keying-agent is never run as root to prevent faults in the agent from giving root-access. Because the relatively unprivileged IPSec stacks are placed in separate compartments, a failure in one does not necessarily lead to failures or security breaches in another. The Vaulted VPN IPSec stacks are compartmented, stripped of most privileges and do not possess a complete view of all the keying material on the system. The various messaging channels between Vaulted VPN components are protected by a combination of MAC and DAC to offer a higher level of security than would otherwise be present on a non-MultiLevel Secure operating system.

Because the IPSec stacks are user-mode processes they can be configured dynamically to execute in parallel for better performance and load balancing. They can also be configured to form multiple redundant stacks for checking the correctness of processed IPSec packets. Host-level accreditation in the style of IPSO/MaxSIX can be achieved by reinterpreting the original mechanisms and performing any needed authentication in the IKE negotiation phase. The failure or success of forming a security-association is directly tied to IKE negotiations that can be expanded to encompass other security-models extant.

References

[1] The Internet Key Exchange (IKE) protocol (RFC 2409)- D. Harkins, D. Carrel, Nov 1998

[2] Security Architecture for the Internet Protocol (RFC 2401) – S. Kent, R. Atkinson, November 1998

[3] IP Security Document Roadmap (RFC 2411) – R. Thayer, N. Doraswamy, R. Glenn, November 1998

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

[5] IP Authentication Header (RFC 2402) – S. Kent, R. Atkinson, November 1998

[6] Applying Military Grade Security to the Internet, Computer Networks and ISDN Systems, Volume 29 Number 15 – C. I. Dalton, J. F. Griffin, November 1997

[7] Internet X.509 Public Key Infrastructure Certificate & CRL Profile (RFC 2459)– R. Housley, W. Ford, W. Polk, D. Solo, January 1999

[8] HP-UX CMW MaxSix Administrator's Guide, Hewlett-Packard Co., 1995

[9] Compartmented Mode Workstation Evaluation Criteria, Report DDS-2600-6243-91, Defense Intelligence Agency, 1991

[10] TCP/IP Illustrated Vols 1, 2 and 3– Gary R. Wright, W. Richard Stevens, Addison-Wesley 1995.

[11] Secure Computer Systems: unified Exposition and Multics Interpretation, MITRE Technical Report MTR –1997 MITRE Corporation NTIS AD A023588/7– D. E. Bell, L. J. LaPadula March 1975

[12] U.S. Department of Defense security options for the Internet Protocol. (RFC1108) – S.T. Kent, November 1991

[13] Revised IP Security Options (RFC 1038) – M. St. Johns, January 1988

[14] STREAMS/UX for the HP 9000 Reference Manual, Hewlett-Packard Co, 1995

[15] The Internet IP Security Domain of Interpretation for ISAKMP (RFC 2407), D. Piper, November 1998

[16] IANA Assigned Port Numbers, ftp://ftp.isi.edu/in-notes/iana/assignments/port-numbers

[17] HP Praesidium Extranet VPN, Hewlett-Packard Co., 1998

[18] SOCKS Protocol Version 5 (RFC 1928), M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones, March 1996

[19] CERT Advisory CA-97.28 IP Denial-of-Service Attacks (Teardrop_Land), CERT Coordination Center, May 1998.

[20] CERT Advisory CA-96.26 Denial-of-Service Attack via PING, CERT Coordination Center, December 1997

[21] Resource containers: A new facility for resource management in server systems, G. Banga, P. Druschel, J. C. Mogul, USENIX Security Symposium 1997

[22] httperf – A Tool for Measuring Web Server Performance, D. Mosberger, Tai Jin, Hewlett-Packard Research Labs

[23] netperf 2.1– Network Performance Benchmark, http://www.netperf.org/

[24] Better Operating Systems Features for Faster Network Servers, G. Banga, P. Druschel, J. C. Mogul