IP Security Policy
Internet-Draft
Expires: October 15, 2006

M. Richardson
SSW
B. Sommerfeld
Sun

April 13, 2006

# Requirements for an IPsec API
# draft-ietf-btns-ipsec-apireq-00.txt

## Status of this Memo



## Copyright Notice



## Abstract

Given the open nature of the Internet today, application protocols require strong security. IPsec's wire protocols appear to meet the requirements of many protocols. The lack of a common model for application-layer interfaces has complicated use of IPsec by upper-layer protocols. This document provides an overview of facilities which a host IPsec implementation should provide to applications to allow them to both observe and influence how IPsec protects their

communications.

Table of Contents

## 1. Motivation for this work

Many protocols under development are considering the use of IPsec for security. Unfortunately, most existing IPsec implementations ([RFC2401] and [RFC4301]) do not give applications any visibility into what, if anything, they are doing on behalf of an application. This limitation only allows IPsec to do all-or-nothing access control, and requires two levels of authentication, with one within the application, and a second level within an IPsec key management protocol (most typically IKE [RFC2407][RFC2408][RFC2409] and IKEv2 [RFC4306][RFC4307]).

## 2. Terminology

The term "socket" will be used here to identify an application-layer communications endpoint; it does imply any specific API is to be used. For the purposes of this discussion, a socket may include:

A communications endpoint for a connectionless protocol
One end of an established connection for a connection-oriented protocol
A listening endpoint for a connection-oriented protocol

For the purposes of this document, the term "application" refers to programs implementing any client protocol using either IP or a transport protocol such as TCP or UDP running over IP. Note that this is in many ways somewhat broader than the traditional use of "application" within the IETF, as it may also include "infrastructure" protocols built on top of IP and IPsec, including routing, ICMP, etc.

## 3. Motivations for this work

Most protocols for application security, such as TLS [RFC2246] and SSH [RFC4251] operate at or above the transport layer. This renders the underlying transport connections vulnerable to denial of service attacks, including connection assassination [RFC3552]. IPsec offers the promise of protecting against many of these denial of service attacks.

There are other potential benefits. Conventional software-based IPsec implementations isolate applications from the cryptographic keys, improving security by making inadvertant or malicious key exposure more difficult. In addition, specialized hardware may allow encryption keys protected from disclosure within trusted cryptographic units. Also, custom hardware units may well allow for higher performance.

Areas where this is currently under active discussion include the set of block storage protocols being developed by the IP Storage working group [RFC3723] and NFS version 4 (XXX: need newer reference than target="I-D.ietf-nfsv4-ccm")

## 4. Goals

Separate policy and mechanism

## 5. Requirements

Here are some basic requirements for an IPsec application API:

An application should be able to determine HOW a communication was protected (or not).

An application should be able to determine WHO it is talking to.

If a communication is nominally authorized but fails, an application should be able to get an indication of WHY it failed, to help identify the configuration error causing the spurious failure.

An application should be able to influence HOW a communication is protected, subject to override or modification by system policy.

An application should be able to indicate WHO it wishes to talk to, again subject to override or modification by system policy.

These interfaces should be as independant as possible of the key management protocol being used; it should be possible to implement this with IKEv1, IKEv2, KINK, etc.,

## 6. System policy

Interactions with system policy:

System-level policy trumps all

By default, applications should be able to ask for *more* protection.

Applications wishing *less* protection may need appropriate local privileges. (example: ike bypass of UDP port 500; DHCP lease renewals...)

## 7. HOW

An application may have requirements for confidentiality and/or integrity; it should be able to determine if an inbound communication was protected and whether an outbound communication will be protected. In addition, there may well be a desire to express preferences for relative strength of algorithms, or specify the

specific algorithm to be used. Hard-coding algorithm names into applications should be actively discouraged; perhaps there should be generic "weak" or "strong" indications instead of specific algorithm identifiers.

## 8. WHO

This is perhaps the most tricky part of the problem. Existing IPsec key management protocols provide a wide variety of authentication methods -- preshared secrets, public key, Kerberos, X.509 certificates, etc.,

There are several potential uses for names provided by IPsec:

### 8.1. OPAQUE IDENTITY

It should be possible to determine that two IPsec-protected communications conducted within a short to medium time frame were with the same authenticated peer; it should be possible to use a received identity to initiate a communication back to that identity.

Example cases: connectionless replies; linking ftp control and data connections.

The application need only be able to determine if two identities are equal.

### 8.2. AUDITING

It should be possible for an application to construct a log entry naming the peer.

### 8.3. ACCESS CONTROL

While policy rules may allow traffic to be blocked entirely, it's often necessary for a program to provide services to mutually suspicious clients. It should be possible for a service to make appropriate access control decisions based on the identity of the peer; in addition, the peer's certificate may contain interesting SubjectAltName or other attributes which may have relevance for the application; it may also be possible for the system to derive other attributes from the peer's identity.

### 8.4. ATTRIBUTES/CREDENTIALS

[Mission Creep Alert] In many cases, an application is not so much interested in the peer's name, but rather in some other attribute of

the peer. Exactly where and how to map from long-term keys to these attributes needs to be nailed down; it may well be that this is best left as a local issue.

Some of this is probably out of scope for the working group; however, we should not preclude others from building on this.

## 9. Error reporting

There are a number of reasons why a communication may fail because of IPsec configuration mismatches..

These include, but are not limited to:

- Blocked by local or peer SPD.
- Local or peer key management protocols cannot establish an SA.
- Local or peer key management protocols cannot authenticate to each other.

It MAY be appropriate to map IPsec failures into existing error codes (e.g., "connection refused", "connection timed out"), so that existing applications use appropriate error recovery strategies; however, this does result in a loss of information. It SHOULD be possible for an IPsec-aware application to get additional information about the reasons that a communications failed.

## 10. Security Guarantees

Connection-oriented and connectionless communication often require different application structure. In many case, it will often be most convenient to do security checks once per connection, while for connectionless communications, per-message operations will be needed.

### 10.1. Connection-oriented communication

Packet boundaries are not, in general, visible to clients of stream protocols such as TCP, while IPsec protection is provided (or not) on a packet-by-packet basis,

In addition, it would be an unreasonable burden on applications to force them to continuously inquire about each individual packet.

It should be possible for an application to ensure that all traffic to a particular socket is protected appropriately; it should also be possible for an application to ensure that all traffic to a socket originates from the same authenticated identity.

A pair of communicating applications should be able to determine that the ipsec protection on a connection between them is end-to-end.

Note that it is common for datagram socket API's to allow a "connect" operation which sets a default destination and filters inbound packets based on source address; it should similarly be possible for the connection-oriented security guarantees to be applied to datagram sockets being used for 1:1 communications.

### 10.2. Connectionless communication

It is also common to use datagram sockets for many-to-many communication; it should be possible to get and set identity information on a packet-by-packet basis.

It may well be the case that a datagram-oriented client application will use the connection-oriented part of this API (because it is using a given datagram socket to talk to a specific server) while the server it is talking to use the connectionless API because it is using a single socket to receive requests from and send replies to a large number of clients.

## 11. Non-goals And Bad Ideas

Here are a few ideas which have popped up every so often which really seem to be bad ideas.. in other words, things which should not be exposed to applications because they can't be used reliably or which cause active harm.

### 11.1. Exposure of keys

There is absolutely no reason for applications to see the underlying encryption keys, or influence the choice of keys. This is to allow an IPsec implementation to have a clear boundary around its cryptographic components.

### 11.2. Exposure of IPsec SPI values

In general, there is no need for applications to see SPI values or keys; it's also the case that in many cases the exact algorithm used may not be of interest as long as it is appropriately strong.

Since both IKE and IPsec SA's may be short-lived, it is plausible that:

an application connection or association will outlive any given IPsec SA.

an application connection or association will outlive any given

IKE SA.
an application connection may be idle for extended periods, during which time there is no IKE or IPsec SA state between the peers.

It should be the case that any properties provided to applications regarding peer identity, protection, etc., should be able to survive rekeying.

It may be appropriate to use SPI values as temporary handles, but applications may last much longer than SA's, and SPI values may be recycled over time; it would be better for there to be a separate, local-use-only, space for (identity, params) pairs.

## 12. Other issues

Interface-specific vs. application-specific policy; deal with this as separate layers of filtering/intersections/etc?
Real-time notifications of both ends that rekey, etc., is having trouble (highly desirable for VoIP-type applications).
Balancing keeping full certificate handling out of applications while still providing full access to certificate attributes.

## 13. Security Considerations

## 14. Document TODO

Flesh out Other Issues section.
Flesh out Informative References with references to existing IPsec-related API's
Improve security considerations section.

## 15. References

### 15.1. Normative References

[RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[RFC2408] Maughan, D., Schneider, M., and M. Schertler, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4307] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, December 2005.

### 15.2. Informative References

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC3723] Aboba, B., Tseng, J., Walker, J., Rangan, V., and F. Travostino, "Securing Block Storage Protocols over IP", RFC 3723, April 2004.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

Authors' Addresses

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

Email: mcr@sandelman.ottawa.on.ca
URI: http://www.sandelman.ottawa.on.ca/

Bill Sommerfeld
Sun Microsystems
1 Network Drive
Burlington, MA 01803
US

Phone: +1 781 442 3458
Email: somerfeld@sun.com

## Intellectual Property Statement



## Disclaimer of Validity



## Copyright Statement



## Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.