Abstract

Usually network layer security through IPsec is transparent for applications. Usually, they cannot detect the presence of IPsec or affect its security properties. This document specifies extensions to increase the visibility of IPsec to applications through a low level sockets API. In addition, a higher layer APIs based on GSS is
also defined. The GSS based APIs increase security level of the application through the dual use of both transport and network layer security. It can be also used to improve protocol performance by reducing redundant authentications.
1. Introduction

The network communications of applications are usually secured explicitly with TLS [4] or using even higher layer interfaces such as GSS [3]. However, such interfaces do not exist for IPsec because it operates on lower layers and is mostly transparent for applications. Applying of IPsec to existing applications is therefore easier than with e.g. TLS because IPsec does not require changes in the application. However, it is difficult for the application to detect whether network connections are secured or not using IPsec.

In this document, we attempt to increase the visibility of the IPsec layer to the applications by introducing an API to IPsec based applications. The interface is an extension to the standard sockets API [1], similarly as with non-secured network applications. The benefit of this interface is that it does not require completely rewriting the networking part of an existing application as in GSS or TLS.

We also introduce an explicit way of enabling IPsec in applications that is based on the GSS API. This API allows the dual use of both IPsec and higher layer security mechanisms simultaneously which the following security and performance related benefits (as described in more detail in [11]):

- Robust security for application data through the use of dual-layer security mechanisms
- Performance optimization and reduced number of authorization configurations by removing unnecessarily redundant security features at different layers

Figure 1 illustrates three different applications, one of which uses GSS APIs, second which use the socket based IPsec APIs and a third which relies on transparent IPsec security.
Figure 1: Security API Layering

This document is an attempt to fulfill the BTNS requirements in [5]. Currently, it contains also some references to other similar work to provide a slightly broader view.
2. Sockets Layer API

The sockets API IPsec APIs are currently defined in [6]. The definitions consist of three main extensions to the standard sockets API which are summarized in this section briefly.

First, IPsec based socket APIs can only be used by an application only when it opens a socket with the protocol family (domain) set to PF_SHIM instead of AF_INET or AF_INET6. The new family enables the following new properties in an application:

- Protocol independent sockets for applications
- Detection of IPsec API support on the localhost

Enables application to use sockets API based communication interface with key management daemons and possibly IPsec modules. Allows querying of IPsec security properties using getsockopt interface. Setting of security properties is also possible, but should require access privileges verification because it can affect also the communications of other processes.

Second, the API proposes an abstraction mechanism that avoids exposing IPv4 and IPv6 addresses directly to the applications. Instead, application identifier is a "handle" (socket descriptor) which refers the actual transport layer identifier. The socket descriptors can used through the standard hostname-to-address resolver interface [2] with the use of a special flag. When the flag is set in the resolver arguments, the resolver returns socket address structures specific to socket descriptors instead of IPv4 or IPv6 socket structures. Some of the possible use cases for the socket descriptor abstraction mechanism are listed below:

- When transport layer identifiers are public keys, they may not fit into socket address structures. The constant-size socket descriptor is a convinient replacement for the public key.
- System access control mechanism for user specific identifiers (such as private keys) may be easier to implement using the socket descriptors as they resemble file descriptors
- May ease the implementation of various security and mobility features to protocols, such as opportunistic HIP mode [9], and maybe SHIM6 context forking [10].
- Isolates transport layer identifier changes from applications for future extensions. Might also benefit multicast APIs
Third, applications are allowed to listen for events reported by the key and mobility management daemons. The event interface is similar as in SCTP [8] that operates events through the sendmsg/recvmsg interfaces [2]. These interfaces allow sending and receiving regular networking data in a datagram oriented way, but also facilitate the use of ancillary data. For example, application can register to listen for events that are related to locator changes. The notification to the application arrives in a sendmsg/recvmsg call. The application can differentiate a notification from regular network data by examining a flag in the message structures. Please refer to [7] for a more detailed description of this.
3. Session Layer API

The GSS API for IPsec can make use the IPsec sockets API as depicted in figure Figure 1. The exact definitions are still TBD.
4. IANA Considerations

TBD
5. Security Considerations

TBD
6. Acknowledgements

Thanks for Michael Richardson and Love Hoernquist Aestrand for discussion on the topic.
7. References

7.1. Normative References


7.2. Informative References


