IPsec

Patrick McDaniel
AT&T Labs – Research
pdmcdan@research.att.com

Prior to the explosion of computer networks in the late 1980s, enterprise environments were largely isolated collections of hosts. The protocols used to connect those computers did not require much security. Indeed, few security issues were considered by original designers of the *Internet Protocol* (IP) suite upon which those and subsequent networks are based. While the openness of these protocols is a key ingredient to the Internet's success, the lack of security has led to many troublesome problems. For example, many otherwise safe systems have been compromised by an adversary who forges IP addresses. Such address "spoofing" is trivial on the current Internet. These and other security problems continue to confound the users and administrators of the Internet.

IPsec is a protocol suite that adds security to the existing IP protocols [KA98]. Standardized by the Internet Engineering Task Force [iet04], IPsec defines new IP message formats and the infrastructure used to define and manage security relevant state. IPsec is a general purpose architecture. Hosts, networks, and gateways define policies for each class of traffic they wish to secure. These policies define what security services they desire to apply to the traffic (e.g., authenticity, confidentiality).

IPsec provides security properties that are specific to the network medium. For example, the IPsec authentication service ensures that a host receiving a packet is able to determine that a packet was transmitted by the host or network that claims to have sent it. A related property, *integrity*, allows that same host to assert that the packet data, called payload, was not modified in transit. Authenticity and integrity are implemented in IPsec using the *authentication header* (AH) transform and optionally by the *encapsulating security payload*.

Confidentiality is also defined by IPsec with respect to the hosts and networks implementing it. Where confidentiality is configured, an IPsec host is guaranteed that the data transmitted between hosts (e.g., payload) is only visible to the intended recipient. To put it another way, the payload and optionally the IP header cannot be viewed by any intermediate node or external entity on the intermediate network. Confidentiality is implemented in IPsec using the *encapsulating security payload* (ESP) transform.

IPsec defines the requirements of end-point authentication (e.g., credentials and methods) and the protocols and procedures used for the creation and management of state. These services define not only how key agreeement is reached, but also how a concrete set of services and parameters is negotiated. These procedures are implemented by the abstract *Internet Security Association and Key Management Protocol* (ISAKMP) [MSST98] and concrete *Internet Key Exchange* (IKE) [HC98] protocols.

IPsec provides a general architecture for secure networks. It has been particularly useful in supporting interesting network services. For example, IPsec is an ideal technology for implementing virtual private networks (VPNs). This is principally because its ability to operate in tunnel mode, where intermediate gateways can securely transport traffic between private networks over untrusted networks like the Internet. IPsec has also been extremely useful in providing security in environments where the physical media is exceptionally vulnerable (e.g., wireless networks).

This remainder of this entry will explore the architecture and operation of the IPsec protocol suite. We begin in the next section by describing the entities and services comprising the IPsec infrastructure.

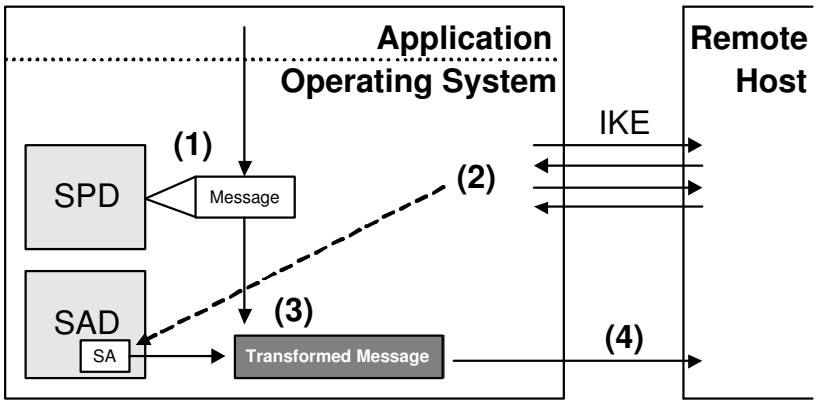

Figure 1: IPsec packet processing

The IPsec Architecture

The IPsec architecture [KA98] coordinates hosts and network elements to ensure that the information flowing between hosts is secured. How the security is defined for a particular environment is determined by policy. Policies are configured manually or obtained from the emerging IPsec policy system [BKRS03] and applied to packet traffic.

The IPsec the *Security Policy Database* (SPD) states processing rules for network traffic. The selector is a unique (to the SPD) collection of address, protocol, and type of service bits (TOS). The SPD maps selectors onto IPsec policies. These policies define the processing discipline applied to matching packets. The processing policies are similar to those found in firewalls, where a packet can be processed by IPsec, passed (without modification), or dropped. It is up to the host or network administration to determine which policy is most appropriate for particular network traffic.

A central and essential artifact of IPsec state is the *security association* (SA). An SA is a data structure used to define and track communication state and configuration between IPsec end-points. Each SA stores the packet-processing transform (e.g., AH or ESP), parameters (e.g., cryptographic algorithms) and other security-relevant data (e.g., keys, sequence numbers). The SA is created by the key management service as defined below. An SA is uniquely identified by its security parameter index (SPI). The SPI is negotiated at the start of an IPsec session and is later placed in the header field of all relevant IPsec packets. The SAs established by a host are held in the *Security Association Database* (SAD). Unlike most security protocols, IPsec defines unidirectional security associations. That is, the keys and policy are applied in a single direction. The advantage of this approach is that network administrators are free to establish different security policies for each direction or even to restrict the flow of data to one direction.

The following example illustrates the processing of IP traffic by the elements of the IPsec architecture. Assume initially that a local host A (in Figure 1) has not communicated with B recently.[1] An arbitrary application on A attempts to send data to some external host B via the User Data Protocol (UDP) [Pos80] (**1**). Upon reception at the IPsec implementation on A in the host operating system, the protocol(UDP), target address (B), and other information is mapped in the SPD to an IPsec policy. In this example, the policy mandates that ESP in confidentiality only mode and automated key management. In response, the IKE protocol is executed between the two hosts and session keys are established (**2**). The successful completion of the IKE protocol results in the creation of an SA defining the keying material, ESP policy, and SA lifetime. The original packet is transformed per the ESP policy specification (**3**) and transmitted to the remote host (**4**). Note that

[1] IPsec SAs have an explicit lifetime, after which the state is discarded. This example assumes the lifetime of any such previous communication between the hosts has been exceeded.

each subsequent packet sent during the SA's lifetime and matching the original selector will use the same SA (and hence, will not require further involvement of the key management protocol, IKE).

Transforms

An IPsec *transform* defines a packet format and a set of associated processing rules that address a particular set of security guarantees. IPsec defines two transforms, the authentication header (AH) and the encapsulating security payload (ESP). These transforms operate in either transport or tunnel mode. A transform operating *transport mode* secures the payload but not the IP header. This is useful when you need to apply security to the upper layer protocols only (e.g., authenticate TCP header and payload data only). *Tunnel mode* provides the guarantees over the entire IP packet. This is useful when you need to provide guarantees over the IP header fields (e.g., confidentiality of source and target addresses), and is very useful when dealing with operational issues (e.g., simplifies private addressing).

In transport mode, AH includes a message authentication code (MAC) in the IPsec header. The keyed MAC is calculated using the keys defined in the SA and over the payload data. ESP ensures confidentiality by encrypting the payload with the key defined in the SA. Where enabled, ESP includes an authenticating MAC similar to AH. Conversely, AH and ESP tunnel modes completely encapsulate the IP header and payload. The entirety of the IP packet is treated as payload, and a new IP packet is formed around it.

The original IP header data is protected (and not visible in the case of ESP) in tunnel mode. This has the advantage that an observer will (and again in ESP can only) see the packet as originating from the device that tunneled it. This allows network architects to use IPsec tunneling devices as *security gateways*. These gateways serve to separate and conceal sensitive traffic traveling across untrusted networks. Such gateways are a central element of contemporary virtual private networks (VPNs).

Another security guarantee supported by both transforms is replay protection (e.g., also known as anti-replay). Replay prevention ensures that packets are processed by the receiving host at most once, and hence are not "replayed" maliciously or accidentally. This feature is implemented by authenticating packet sequence numbers. Every SA has a sequence number that is initially set to 0 and incremented by one after a packet transmission. A transmission window set by policy to some agreed size (typically 64 or 128). Every time a packet is received, the sequence number is checked. Any correctly authenticated packet packet who's sequence number falls in the window and has not been seen before is accepted. If the packet it newer (has a sequence larger than the the right side of the window), the window is moved to the right to accept the packet. If the packet is older (to the left of the window), is is dropped.

Key Management

IPsec would not be very useful without key management. The purpose of key management facility in IPsec is to determine and distribute the keys used by the payload-processing transforms, and to secondarily negotiate the policy defining the SAs. All of these functions are implemented in the abstract ISAKMP architecture by IKE. Strictly speaking, ISAKMP, IKE, and IPsec are separate standards, but have in recent years have become largely inseparable.

IPsec provides for two kinds of key management, static and automatic. Environments with static key management simply identify the keys to be used to secure the communication between the end-points (e.g., the IPsec SAs). While avoiding the complexity and cost of implementing a key management protocol, this approach can be difficult to manage. Each end-point must be configured with an SA used to communicate with every other end-point. This is time-consuming and error prone where many end-points must be managed. Moreover, because it potentially exposes a large amount cipher-text (because the same key may be used for an indeterminate time), manual keying is not frequently viewed as good security practice.

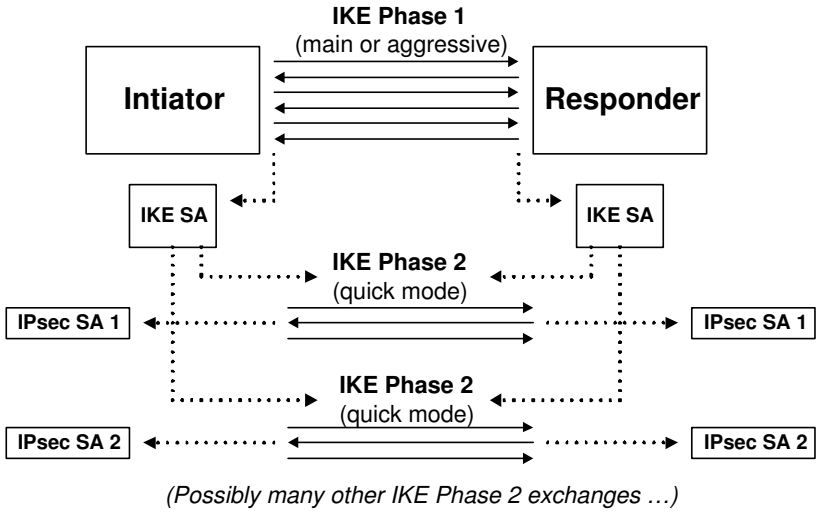

Figure 2: IKE Phase 1 and 2 protocol flow

ISAKMP is an abstract key management architecture. It defines the possible states, transitions, and (abstract) exchanges one uses to establish a shared key. Based on an authenticated Diffie-Hellman exchange and built on the ISAKMP architecture, IKE is a concrete protocol. IKE serves three purposes: (a) to establish the policy for an IKE session, (b) to establish a key for the IKE SA (see below), and (c) to establish particular SAs for the IPsec processing of IP and upper layer data.

As illustrated in Figure 2, IKE works in two phases. The first phase (creatively called *phase 1*) allows the two end-points to establish an *IKE SA*. An IKE SA defines the keys and policy used to establish regular payload processing SAs, called *IPsec SAs*. Phase one can operate in two modes. In *main-mode*, IKE implements a six message protocol which provides identity protection. Identity protection ensures the initiator's (the host that first tries to initiate communication) identity is not exposed to an attacker who is actively attacking the system (e.g., by hijacking the address of the intended host). Where such protection is not needed, the protocol can use a simpler, but less secure, three message *aggressive mode* protocol. The end result of either the main or aggressive mode protocol is the same, an IKE SA.

The simpler IKE Phase two exchange is appropriately named *quick mode*. This phase uses a previously established IKE SA to create an IPsec SA. The hosts perform the quick mode operation in three messages: an initial request with keying material, a response to the request, and an acknowledgment of the response. To simplify, the keys and configuration used to define the IPsec SA are generated from the values passed between the parties. If such a thing is deemed desireable, the parties may optionally engage in another Diffie-Hellman exchange. Note that a single IKE SA can be used to establish many different IPsec SA, even simultaneously.

Considerations

If all these modes and transforms seem complicated, they are. A central criticism of IPsec has been its complexity, mostly due to its attempt to be a suite of protocols that addresses the requirements of many constituents. The reasons for its broad task are obvious: because the world uses IP, IPsec must address the problems of the world. However, addressing such problems have evidently led to extremely difficult to implement (and often to manage) protocols. Recently, draft standards have surfaced within the IETF that attempt to simplify IKE, so there may be some relief on the horizon.

References

[BKRS03] M. Blaze, A. Keromytis, M. Richardson, and L. Sanchez. IP Security Policy (IPSP) Requirements. *Internet Engineering Task Force*, August 2003. RFC 3586.

[HC98] D. Harkins and D. Carrel. The Internet Key Exchange. *Internet Engineering Task Force*, November 1998. RFC 2409.

[iet04] Internet Engineering Task Force (IETF), 2004.

[KA98] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *Internet Engineering Task Force*, November 1998. RFC 2401.

[MSST98] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). *Internet Engineering Task Force*, November 1998. RFC 2408.

[Pos80] J. Postel. User Datagram Protocol. *Internet Engineering Task Force*, August 1980. RFC 768.