CHAPTER

IP Security

29

What You Will Learn

In this chapter, you will learn how IPSec adds another level of security to a TCP/IP network by adding IPSec to the MPLS-based VPN that we built in Chapter 26. We'll investigate the IPSec architecture and how its features are usually implemented.

You will learn about security associations and how authentication and encapsulation work in IPSec. We'll briefly mention the Internet key exchange (IKE) as a secure way to move keys around the network.

IPSec, as has been pointed out, is really a piece of IPv6 that was pressed into service for IPv4, mostly out of desperation after businesses began to use the Internet for more than just amusement. The formats for IPv4 and IPv6 IPSec are different, given the difference in header and address formats, but they are still very similar. Optional in IPv4, support for IPSec is mandatory in IPv6. IPSec is part of a public key infrastructure (PKI) architecture based on several things that we've talked about before: public key encryption, secure key exchange for the Internet (IKE), and several related concepts and protocols.

There are several key concepts in IPSec, as with anything else in TCP/IP. We'll talk about IPSec modes first, followed by security associations (SAs) and a closely related concept, the security parameter index (SPI). Then we'll focus on the three main "protocols" that make up IPSec: the authentication header (AH), the encapsulating security payload (ESP), and the IKE.

IPSec consists of two main "core protocols"—AH and ESP—although it is often pointed out that they are not really protocols at all because they cannot function on their own. AH allows a receiver to verify that the claimed originator of the message actually did send it, and that none of the data has been altered while in transit. It also prevents captured messages from being used again in the future (e.g., when a hacker cannot *read* the password but knows that this packet will log in the user when sent). This is called a *replay* attack.

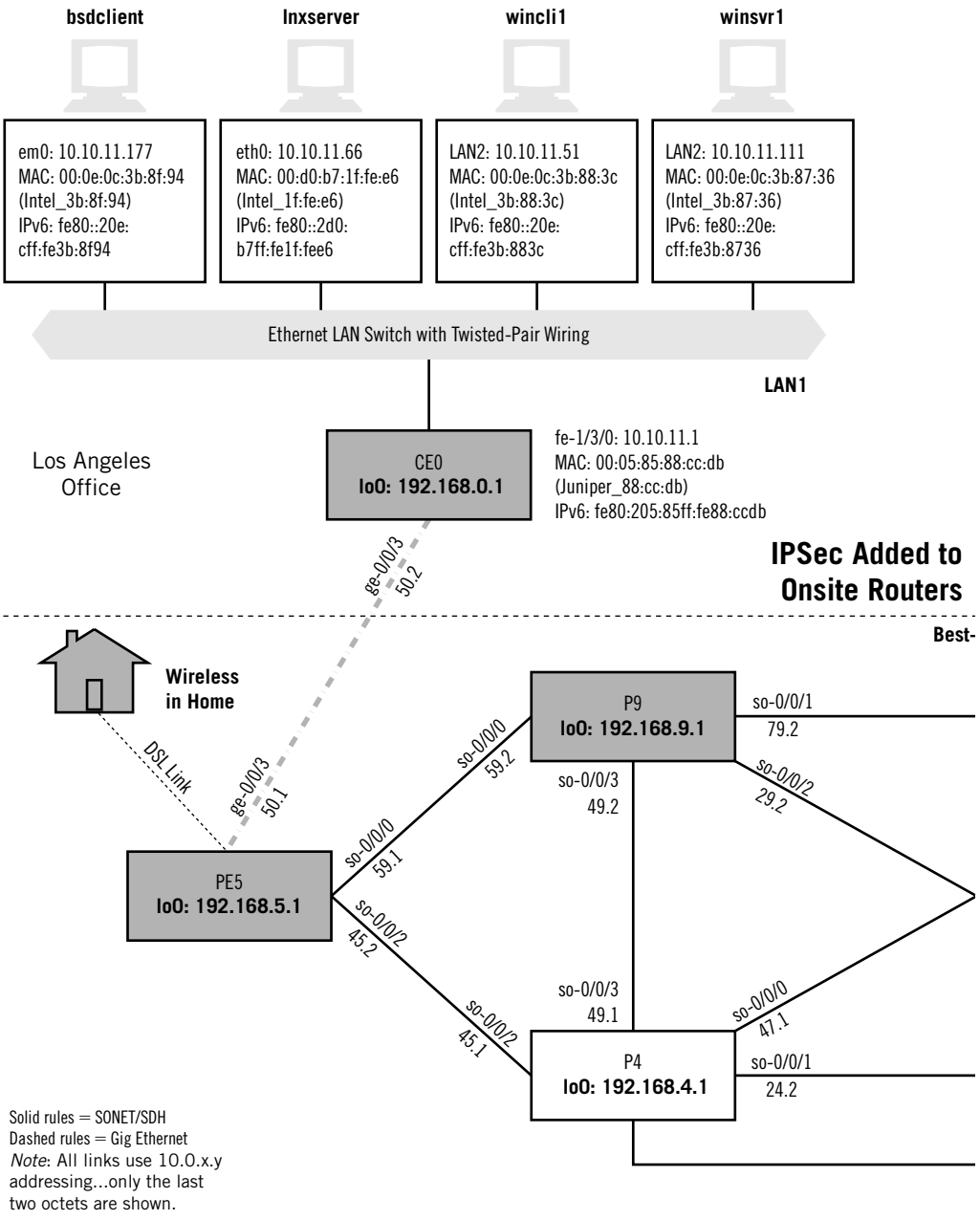
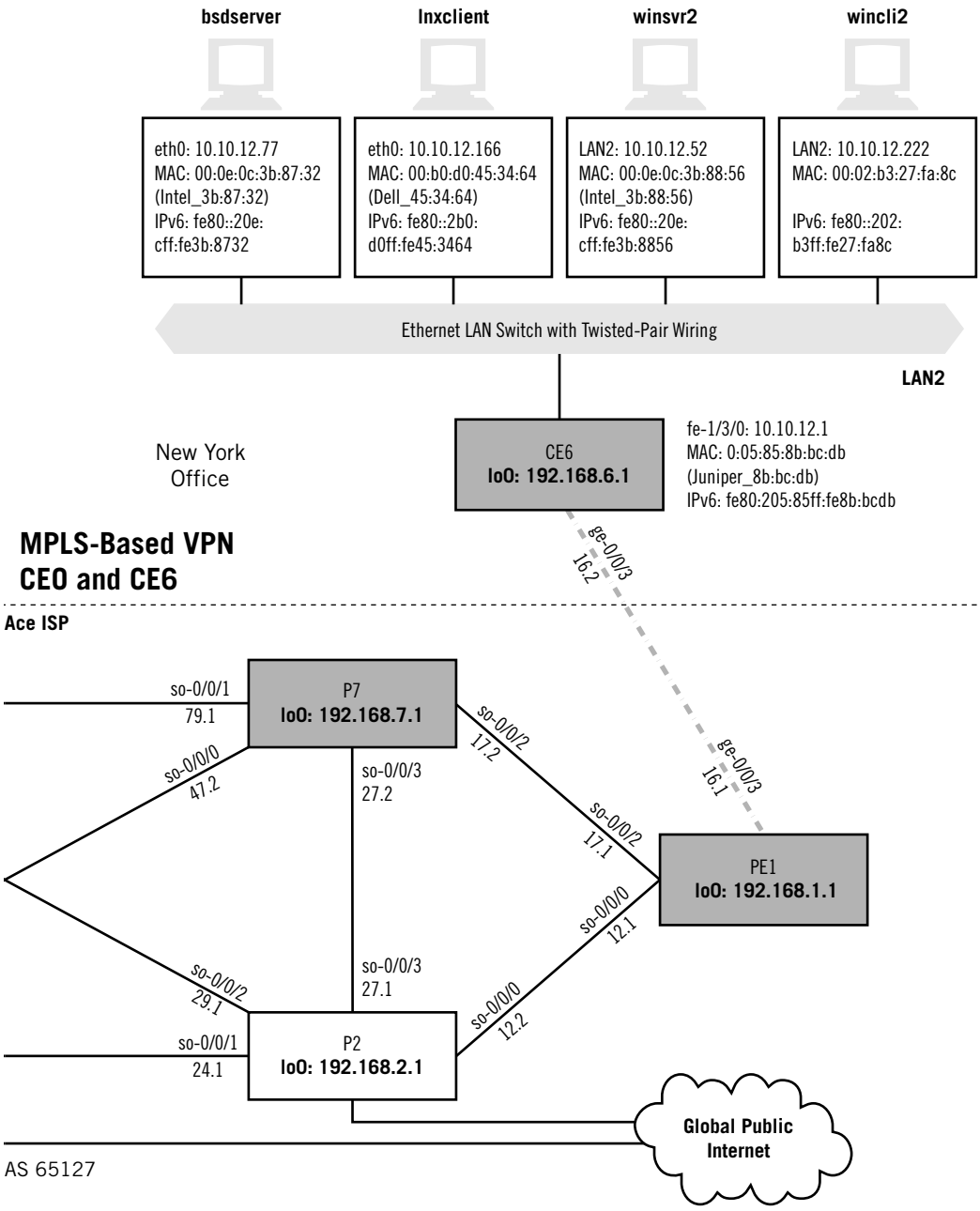


FIGURE 29.1

IPSec on the Illustrated Network, showing how IPSec adds security to the site routers connected by the MPLS-based VPN.


bsdserver
lnxclient
winsvr2
wincli2
eth0: 10.10.12.77
MAC: 00:0e:0c:3b:87:32
(Intel_3b:87:32)
IPv6: fe80::20e:
cff:fe3b:8732
eth0: 10.10.12.166
MAC: 00:b0:d0:45:34:64
(Dell_45:34:64)
IPv6: fe80::2b0:
d0ff:fe45:3464
LAN2: 10.10.12.52
MAC: 00:0e:0c:3b:88:56
(Intel_3b:88:56)
IPv6: fe80::20e:
cff:fe3b:8856
LAN2: 10.10.12.222
MAC: 00:02:b3:27:fa:8c
IPv6: fe80::202:
b3ff:fe27:fa8c
Ethernet LAN Switch with Twisted-Pair Wiring
LAN2
New York
Office
CE6
lo0: 192.168.6.1
fe-1/3/0: 10.10.12.1
MAC: 0:05:85:8b:bc:db
(Juniper_8b:bc:db)
IPv6: fe80:205:85ff:fe8b:bcdb
ge-0/0/3
16.2
MPLS-Based VPN
CE0 and CE6
Ace ISP
so-0/0/1
79.1
P7
lo0: 192.168.7.1
so-0/0/2
17.2
so-0/0/0
47.2
so-0/0/3
27.2
ge-0/0/3
16.1
so-0/0/2
17.1
PE1
lo0: 192.168.1.1
so-0/0/0
12.1
so-0/0/2
29.1
so-0/0/3
27.1
so-0/0/0
12.2
so-0/0/1
24.1
P2
lo0: 192.168.2.1
Global Public
Internet
AS 65127

ESP encrypts the payload of the message itself. It might sound odd that authentication and encryption are separate processes in IPSec, and in practice both are normally used together. Separating the processes allows them to evolve independently, however, so advances in encryption do not require changes in authentication (and vice versa).

We'll add IPSec to the MPLS-based VPN we created in the VPN chapter, as shown in Figure 29.1. We'll still use that same configuration on the routers, but add to it.

IPSec IN ACTION

As with NAT and stateful firewalls, the implementation of IPSec on the Juniper Networks routers used on the Illustrated Network depends on a special "internal interface" supported by an adaptive services physical interface card (AS PIC). All of the routers have these PICs, so we can build IPSec onto the configuration used for the MPLS-based VPN that we built for VPLS in Chapter 26.

Our goal here will be to add an IPSec tunnel using ESP between the CE0 and CE6 routers attached to LAN1 and LAN2, and at the same time preserve the VPLS VPN between routers PE5 at LAN1 and PE1 at LAN2. The packets flowing between LAN1 and LAN2 on the links between routers PE5 and PE1 will be encapsulated and encrypted (with IPSec), and then encapsulated again (for VPLS). Is this paranoia? Perhaps. But the idea is to raise the hacker work factor on these packets high enough so that the hackers give up and move on to less protected traffic.

We could configure manual SAs on each router and configure IKE to carry this information over the network, but such a procedure is overly complex for this chapter. We have to configure the SAs anyway, so we'll just (securely) configure manual SAs on routers CE0 and CE6 to run IPSec with ESP in tunnel mode between them, thereby dispensing with IKE. The VPLS is still there, but transparent to IPSec. The network topology appears as shown in Figure 29.2.

Then we'll show that the IPSec is up and running. (We could show some garbled Ethereal captures between the routers showing that IPSec encryption is in use, but these are not very enlightening.) Again, we'll show the configuration on each router, with comments.

CE0

This router has normal interface configurations, naturally. But we'll define a bidirectional manual SA in a "rule" called `rule-manual-SA-BiESP` and reference it to a "service set" associated with the interface. We'll use ESP, and a value of 261 for the SPI. We'll talk more about security algorithms later, but we'll also use HMAC-SHA1-96 for authentication, DES-CBC for encryption, a 20-bit ASCII authentication key for SHA-1, and an 8-bit ASCII key for DES-CBC authentication.

To get traffic onto the PIC and the IPSec tunnel, we have to match the LAN traffic with our IPSec VPN selector rule. Fortunately, this rule is already referenced in the

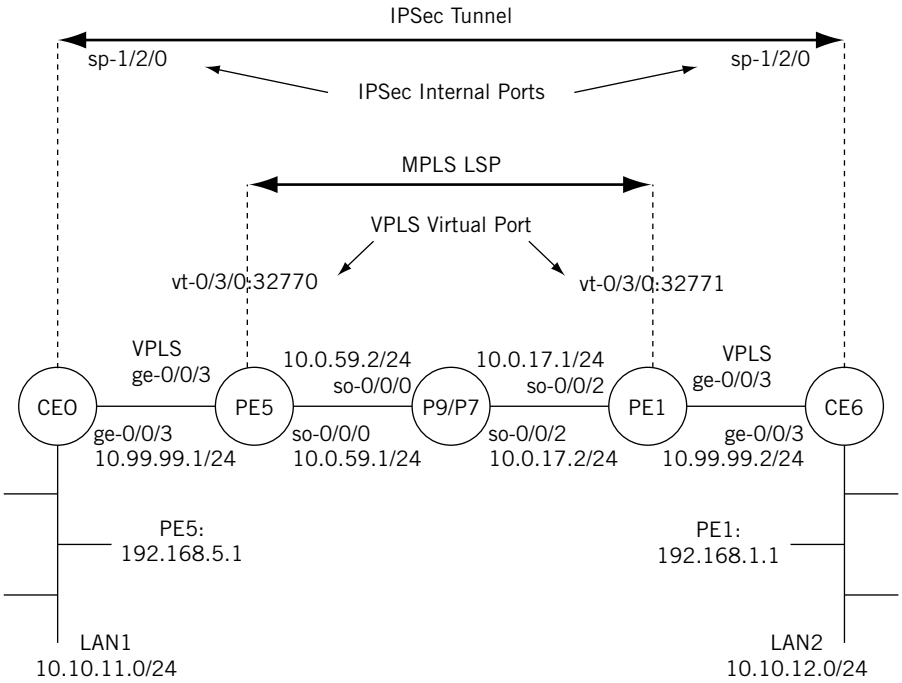


FIGURE 29.2

IPSec topology, showing how it relates to the MPLS LSP and VPLS.

service set from the VPN configuration. We'll also use a firewall filter to count the packets entering the IPSec tunnel.

```
set interfaces ge-0/0/3 vlan-tagging;
set interfaces ge-0/0/3 unit 0 vlan-id 600;
set interfaces ge-0/0/3 unit 0 family inet
    service input service-set service-set-manual-BiESP;
set interfaces ge-0/0/3 vlan-tagging unit 0 family inet
    service output service-set service-set-manual-BiESP;
    # applies the BiESP service set to input and output traffic
set interfaces ge-0/0/3 unit 0 family inet address 10.99.99.1/24;

set interface sp-1/2/0 unit 0 family inet filter input ipsec-tunnel;
    # configure the internal IPSec tunnel interface
set firewall filter ipsec-tunnel term 1 then count ipsec-tunnel;
set firewall filter ipsec-tunnel term 1 then accept;
    # configure a filter to count and process traffic

set services service-set service-set-manual-BiESP interface-service
    service-interface sp-1/2/0;
    # defines the main IPSec tunnel service set applied above
```

```
set services service-set service-set-manual-BiESP ipsec-vpn-options
    local-gateway 10.99.99.1; # the local IPSec tunnel addr
set services service-set service-set-manual-BiESP ipsec-vpn-rules
    rule-manual-SA-BiESP; # references the IPSec rule defined below

set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    from source address 10.10.11.0/24; # find LAN1 traffic for IPSec
set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    then remote-gateway 10.99.99.2; # far-end IPSec tunnel address
set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    then manual direction bidirectional protocol esp; # use ESP for IPSec
set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    then manual direction bidirectional spi 261; # the SPI is 261
set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    then manual direction bidirectional authentication algorithm hmac-sha1-96;
set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    then manual direction bidirectional authentication key ascii-text
    "$9$v.s8xd24Zk.5bs.5QFAtM8XNVYLGifT3goT369OBxNdw2ajHmFnCZUnCtuEh";
    # the authentication key was enters as 'juniperjuniperjunipe' (20 chars)
set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    then manual direction bidirectional encryption algorithm des-cbc;
set services ipsec-vpn rule rule-manual-SA-BiESP term term-manual-SA-BiESP
    then manual direction bidirectional encryption key ascii-text
    "$9$3LJW/AOEclLxdBlxdbsJZn/CpOR"; # entered as juniperj (8 characters)
set services ipsec-vpn rule rule-manual-SA-BiESP match-direction output;}
```

We need a manual SA key entry because this example is not using IKE. Note that although we type the key in plain text, the result is always displayed in encrypted form.

CE6

We can use exactly the same configuration on router CE6 by just swapping the local and remote gateway addresses on the `ge-0/0/3` interface and under `ipsec-vpn-options` and `ipsec-vpn`, so that `10.99.99.1` and `10.99.99.2` are swapped, and changing the `fe-1/3/0` address to `10.10.12.1`. So, in the interest of brevity, we won't show the CE6 listing.

How do we know that the IPSec VPN tunnel is working? Everything works as before, but that proves nothing. How do we know that traffic between LAN1 and LAN2 is now encrypted? An Ethereal trace can verify that, and we can display the value of the traffic counter (as long as it is non-zero) on the firewall filter we set up on the CE routers.

```
admin@CE6> show firewall filter ipsec-tunnel
Filter: ipsec-tunnel
Counters:
Name                Bytes          Packets
ipsec-tunnel          252                3
```

These counts reflect three pings that were sent from LAN1 to LAN2 over the IPSec tunnel. Other commands can be used to give parameters and details of the SA itself, but the latter just repeats information stored in the configuration file.

Let's see what the major portions of the configuration listing are accomplishing. To do that, we'll have to consider some concepts used in IPSec.

INTRODUCTION TO IPSEC

There are three IPSec *support components* in addition to the transport services provided by AH and ESP. One of these components is a set of encryption and hashing algorithms, most of which we've met already in the SSL and SSH chapters. AH and ESP are generic and do not mandate the use of any specific mechanism. IPSec endpoints on a *secure path* negotiate the ones they will use, as does SSH. For example, two common hashing methods are Message Digest 5 (MD5) and Secure Hash Alogrithm 1 (SHA-1), and the endpoints decide which to use with IPSec.

Other important support pieces are the security policies and the SAs that embody them. The flexibility allowed in IPSec still has to be managed, and security relationships between IPSec devices are tracked by the SA and its security policy.

Finally, an IPSec key exchange framework and mechanism (IKE) is defined so that endpoints can share the keys they need to decrypt data. A way to securely send SA information is provided as well. In summary, IPSec provides the following protection services at the IP layer itself:

- Authentication of message integrity to detect changes of the content on the network
- Encryption of data for privacy
- Protection against some forms of attacks, such as replay attacks
- Negotiation of security methods and keys used between devices
- Differing security modes, called transport and tunnel, for flexibility

IPSec RFCs

When it comes to RFCs, aspects of IPSec are covered in a collection of RFCs that define the architecture, services, and protocols used in IPSec. These are listed in Table 29.1.

IPSec Implementation

Okay, IPSec is wonderful and we all should have it and use it. But how? Where? There are two places (at least) and three ways that IPSec can be implemented on a network.

First, IPSec can be implemented host to host or end to end. Every host has IPSec capabilities, and no packets enter or leave the hosts with encryption and authentication. This seems like an obvious choice; however, the fact is that there are many hosts and, as with "personal" firewalls, this can be a maintenance and management nightmare.

Table 29.1 IPSec RFCs with Title and Purpose

RFC	Name	Purpose
2401	Security Architecture for the Internet Protocol	Main document, describes architecture and how components fit together
2402	IP Authentication Header	AH "protocol" for integrity
2403	The Use of HMAC-MD5-96 within ESP and AH	Describes a popular algorithm for use in AH and ESP
2404	The Use of HMAC-SHA-1-96 within ESP and AH	Describes another popular algorithm for use in AH and ESP
2406	IP Encapsulating Security Payload	The ESP "protocol" for privacy
2408	Internet Security Association and Key Management Protocol (ISAKMP)	Defines ISAKMP methods for key exchange and negotiating SAs
2409	The Internet Key Exchange (IKE)	Describes IKE as ISAKMP method
2412	The OAKLEY Key Determination Protocol	Describes a generic protocol for key exchange, which is used in IKE

And because most data are stored on servers in "plain text" formats, all of this work is often in vain if there is a way into the server itself.

IPSec can also be implemented from router to router, and this approach makes a lot of sense. There are few routers compared to hosts, and perhaps offsite packets are the only ones that really need protection. On the local LAN, the network risks are lower (or *should* be!), and more damage is caused by users leaving themselves logged in and leaving their work locations for breaks or lunch than sniffing "on the wire." When used in combination, IPSec VPNs are a formidable barrier to attacks originating on the Internet. (This is not to say that site security can be *ignored* when IPSec and VPNs are used between routers, but it certainly can be *different*.)

Ideally, in a host or a router, IPSec would be integrated into the architecture of the device. Where IPv6 is concerned, this is exactly the case. But IPSec is still an IPv4 "add-on" and so can be implemented in hosts and routers in different ways that mainly concern where in the network the actual IPSec protection actually kicks in.

There are two common ways to look at IPSec architecture in IPv4. These are sometimes called "bump in the stack" (BITS) and "bump in the wire" (BITW).

In the BITS architecture, IPSec bits are a separate layer between the IP layer and the frames. IPSec "intercepts" the IP packets inbound and outbound and processes them. The nice thing about this approach is that it can be easily added to (and upgraded on) IPv4 hosts.

The BITW technique is common when IPSec is implemented site to site by routers, and devices located next to routers. This architecture is shown in Figure 29.3.

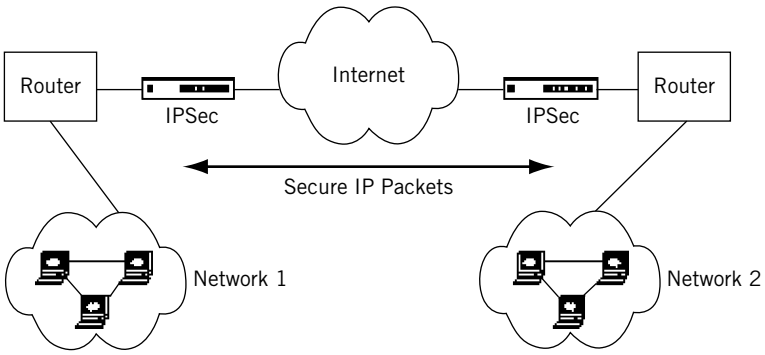


FIGURE 29.3

IPSec and routers, showing how separate devices can be used to apply IPSec to a network.

The IPSec "device" can be implemented in router software or as a separate appliance. The secure packets can be sent over a VPN or simply routed through the Internet, although a VPN adds another layer of protection to the data stream. The two approaches are similar, but have a different impact on each of the two IPSec modes.

IPSec Transport and Tunnel Mode

IPSec modes define the changes IPSec can make to a packet when it is processed for delivery. Modes in turn affect SAs, so the difference is not trivial by any means.

Transport mode—In this mode, the packet is handled as a unit from the transport layer (TCP/UDP). The segment is processed by AH/ESP and the appropriate header added along with a "normal" IP header before being passed down to the frame layer. The main point is that in transport mode, the IP header itself is *not* part of the AH/ESP process.

Tunnel mode—In this mode, IPSec performs its magic on an entire IP packet (original header included). The IPSec headers are placed in front of the encrypted IP packet and then a *new* IP header is placed in front of the entire construction. A nice feature is that the original IP address is encrypted and the new address can be seen as a form of NAT.

Transport mode is feasible only for host-to-host IPSec operation because only hosts have easy access to the transport layer segments. On the other hand, router implementations make use of tunnel mode because routers handle entire IP packets, tunnels are a familiar concept in the router world, and this form of IPSec works well with VPNs. (Some equipment vendors say that tunnel mode is "better" than transport mode, but that is really making a virtue out of necessity.)

SECURITY ASSOCIATIONS AND MORE

An IPSec device negotiates the precise methods and manages keys used for packets sent and received. Here comes a packet from somewhere else. So how will we decrypt it? What is its precise structure (mode)? The same issues come up with outbound packets. How do we know what was negotiated (or possible) for the partner at the other end of the secure path? This is turning out to be much more difficult in practice than in theory. We need help to keep it all straight. The following material describes how it's done in IPSec.

Security Policies

Security policies are general rules that tell IPSec how it can process packets. The security policy can also allow packets to pass untouched or link to places where yet more detail is provided. Security policies are stored in the device's *security policy database* (SPD).

SAs—This is a set of security information describing a particular type of secure path between one specific device and another. It is a type of "contractual agreement" that defines the security mechanisms used between the two endpoints. SAs are unidirectional, so there is one for each direction (inbound and outbound). So, there are at least *four* (and often eight!) SAs that apply to communications between a pair of devices. The SAs are kept in the device's *security association database* (SAD).

Selectors—Which packets does a given SA apply to? The rule sets are called *selectors*. A selector might be configured that applies a certain SA to a packet from a particular range of source IP addresses, or that is going to a certain destination network. SAs don't have names, however. SAs are *indexed* by number, and the number is really a representation (a "triple") of three parameters and not just the SPI.

Security parameter index—The SPI is a 32-bit number picked to uniquely identify an SA for a connected device. The SPI is placed in the AH or ESP headers and links the packet to a particular SA. Once the receiver knows some general information about the packet content, the SPI provides a clue to the rest of it.

IP destination address—The IP address of the device at the "other end" of the SA path.

Security protocol identifier—Tells whether this SA is for AH or ESP. If both are used, they need separate SAs.

The nice thing about using this combination is that any one of the parameters can change to form a "new" entity based on existing pieces. But it can still be confusing.

Authentication Header

AH authenticates by associating a header with a piece of data. The scope of the operation, and the exact placement of the header, depends on the IP version (IPv4 or IPv6) and mode (transport or tunnel). As with many other authentication schemes, AH relies on a hash operation similar in concept to the CRC used on frames. The specific hash (called an *integrity check value* [ICV]) used is stored in the SA and is known only to source and destination. The AH provides authentication, but not privacy. No direct content encryption is used in the AH operation.

AH authentication is simpler for IPv6 than for IPv4 because it was designed for IPv6. In IPv6, the AH is inserted as an extension header using the usual rules for extension header linking. The AH value of 51 is inserted into the IPv6 `Next Header` field. In transport mode, the AH is in the *main* IP header and precedes any destination options and follows an ESP header (if present). In tunnel mode, the AH is an extension header in the *new* IP packet header. These differences are shown in Figure 29.4, with routing (43) and destination option (60) headers in use with a TCP segment.

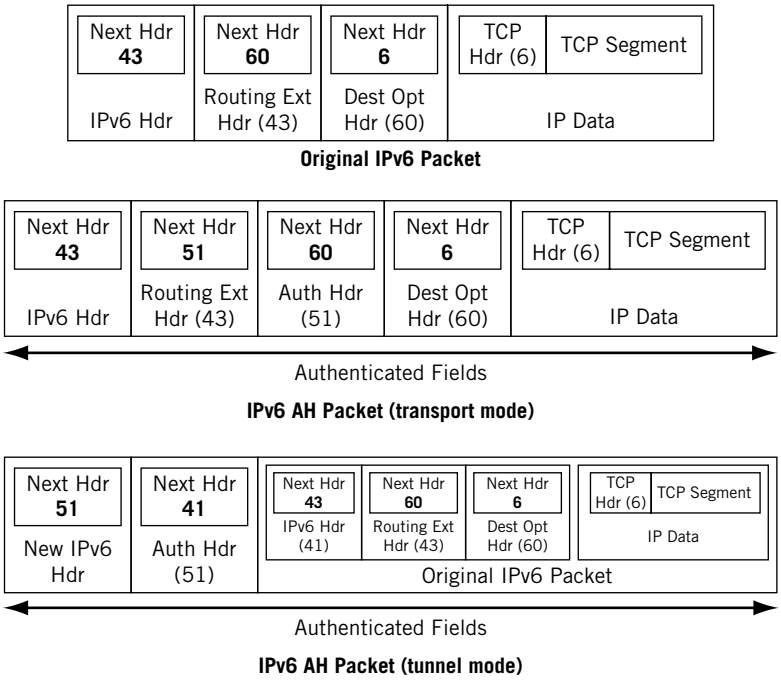


FIGURE 29.4

IPv6 AH packet formats, showing how the various fields and headers relate to one another.

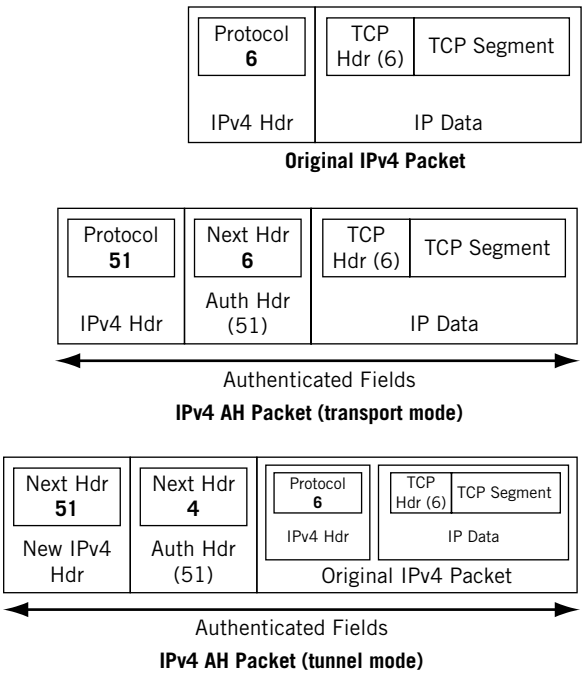


FIGURE 29.5

IPv4 AH packet formats showing how the various fields and headers relate to one another.

In IPv4, the AH has to follow the IPv4 header one way or the other (as shown in Figure 29.5).The fields of the AH itself are described next and shown in Figure 29.6.

Next Header—This 1-byte field gives the protocol number of the *next* header after the AH, *not* the protocol number of the current one.

Payload Length—This 1-byte field measures the length of the AH itself, not really the "payload." It is expressed in 32-bit units, minus 2 for consistency with other IPv6 header calculations.

Reserved—These 2 bytes must be set to all zeros.

Security Parameter Index (SPI)—A 32-bit number that combines with the destination address and type (AH in this case) to identify the SA used for this packet.

Sequence Number—A 32-bit counter that starts at zero when the SA is formed and increments with each packet sent using that SA. This prevents replay attacks with captured packets.

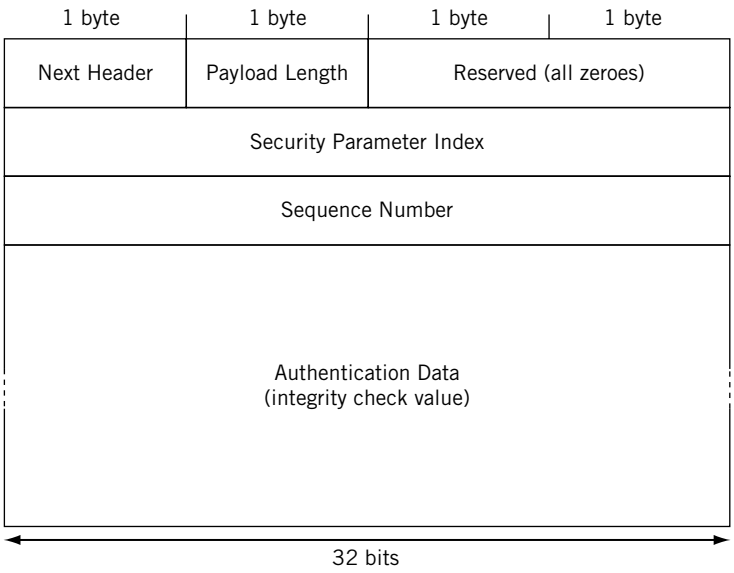


FIGURE 29.6

IPSec AH fields.

Authentication Data—This is the ICV hash and varies in size depending on hashing algorithm used. It must end on a 32-bit (IPv4) or 64-bit (IPv6) boundary, and so is padded with zeros as needed.

Encapsulating Security Payload

ESP encrypts data and adds a header and trailer to the result. ESP has its *own* optional authentication scheme, and can be used in conjunction with AH or not. Unlike the AH "unit," ESP is split up into three distinct pieces. The ESP *header* precedes the encrypted data, and its placement depends on whether IPv6 or IPv4 is used and on mode. The ESP *trailer* follows the encrypted data because some encryption algorithms require that any needed padding follow the encryption. The ESP authentication data with ICV is optional (and redundant when AH is used), so its separation makes sense. It authenticates the ESP header and trailer (and so cannot appear *in* them). This field follows *everything* else.

Placing the ESP headers is different in IPv6 and IPv4, but similar to AH. The trick is finding the ESP trailer because there is no field in the ESP header to give length to or location of the ESP trailer. If it sounds difficult to figure out where the trailer is, that's one of the points. But it can be done, given the correct SA, and the ESP trailer *does* have a next header field to "point back" to the front of the data. Figure 29.7 might make this clearer for IPv6. In transport mode, the ESP trailer value of 60 "points" (it's really in no

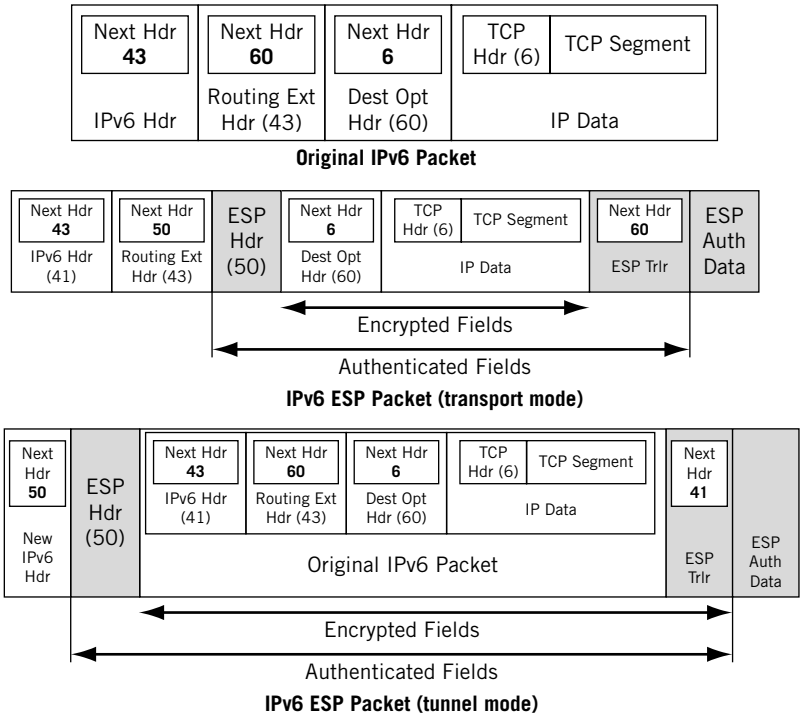


FIGURE 29.7

IPv6 ESP packet formats, showing how the various fields and headers relate to one another.

sense a pointer) to the Destination Options field (value 60) and from there to the TCP header (IP protocol value 6). In tunnel mode, the ESP trailer next header value is 41 and indicates that an IPv6 header comes next.

Figure 29.8 shows the same process for IPv4. In this case, the ESP trailer next header value is 6 for transport mode (TCP header comes next). The value is 4 in tunnel mode, to indicate that an Ipv4 packet is between the ESP header and trailer.

How it all fits together in ESP is shown in Figure 29.9. Note that several fields are only authenticated and not encrypted.

SPI—This 32-bit number is part of the ESP header and is used with destination address and type (ESP, in this case) to be used for this packet.

Sequence Number—This 32-bit number is part of the ESP header and is initialized to zero when the SA is formed and incremented to prevent replay attacks (the same is true in AH).

Payload Data—This is the encrypted data itself and varies in size. Sometimes it contains an *initialization vector,* depending on encryption method.

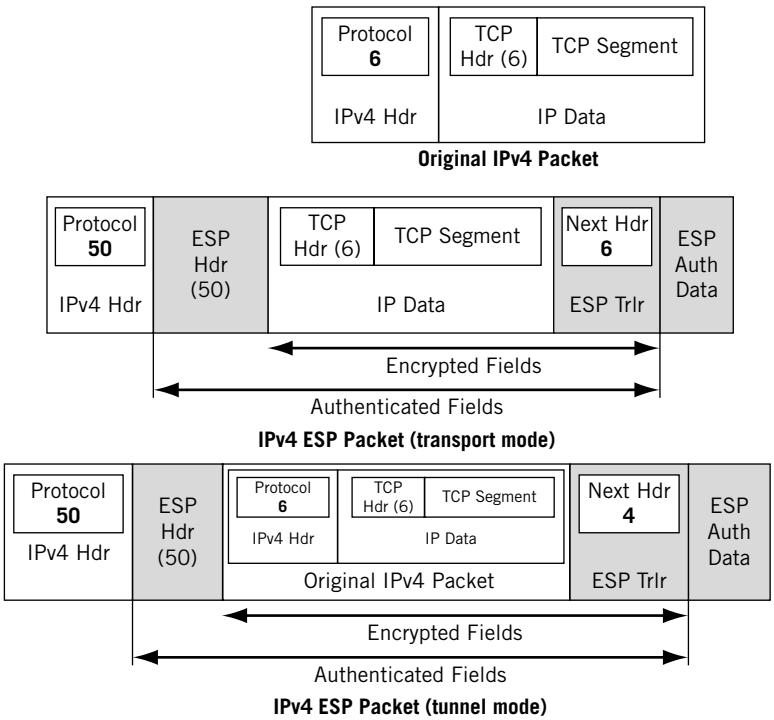


FIGURE 29.8

IPv4 ESP packet formats, showing how the various fields and headers relate to one another.

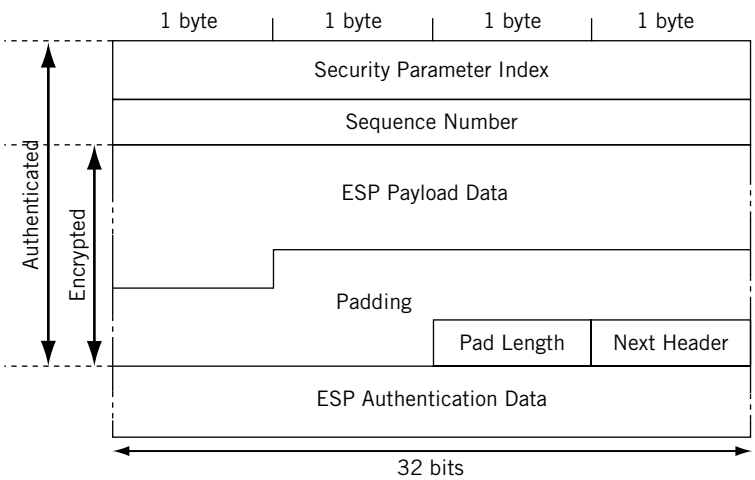


FIGURE 29.9

IPSec ESP fields, showing which fields are authenticated and encrypted.

Padding—This field, from 0 to 255 bytes long, is part of the ESP trailer and is used to align the data as needed.

Pad Length—This 1-byte field is part of the ESP trailer and gives the length of the padding.

Next Header—This 1-byte field is part of the ESP trailer and often "points" to the TCP header (6).

ESP Authentication Data—A variable-length ICV (authentication is optional).

Internet Key Exchange

Our journey through IPSec is almost complete. We've found a way for the endpoints to decide what the formats of the IPSec packets are (the SAs). But what about the keys? Like SSH, IPSec depends on shared secret keys for encryption and decryption. Obviously, the entire method is as secure as the steps taken to secure the keys. That's what IKE is for.

IPSec was actually used before IKE was implemented. So how did the keys get into the SAs and the SAs get everywhere they were needed? An "off-Net" method had to be used. Large organizations used to fly everyone who needed them to a central location and simply hand them out (in sealed envelopes, of course). Smaller organizations used FedEx or some other delivery service. Usually multiple keys, often a great many, were distributed this way, and they changed on a basis known only to those who had to change them.

This method of manual SA definition is still valid and widely used. Sometimes security personnel fly around the country configuring the SAs locally on each router. Few trust "secure" remote access methods for this sensitive task because many millions in financial resources might be at risk. For example, IPSec might have to protect corporate payroll records sent to the banks for employee direct deposit.

IKE is one of the most baffling protocols to understand and explain without a fairly deep knowledge of mathematics and cryptography. Some pieces are not that bad: Diffie-Hellman is the obvious choice for shared secret key exchange, although it says nothing about private/public key distribution. But other components are far beyond the abilities of generalists to understand, let alone know how to explain easily. And there are those who say that you don't really understand something until you can explain it in simple terms to someone else. If that is true, I have yet to find anyone who really understands IKE.

IKE allows IPSec devices to simply send their SAs securely over the Internet to each other. In other words, IKE populates the SAD so that both ends know what to do to send and receive with IPSec. IKE combines (and adds to) the functions of three other protocols.

ISAKMP—The Internet Security Association and Key Management Protocol *is a general* framework protocol for exchanging SAs and key information by negotiation and in phases. Many different methods can be used.

OAKLEY—This extends ISAKMP by describing a specific mechanism for key exchange through different defined "modes." Most of IKE's key exchange is directly based on OAKLEY.

SKEME—This defines a key exchange process different from that of OAKLEY. IKE uses some SKEME features, such as public key encryption methods and the "fast rekeying" feature.

IKE takes ISAKMP and adds the details of OAKLEY and SKEME to perform its magic. IKE has the two ISAKMP phases.

Phase 1—The first stage is a "setup" process in which two devices agree on how they will exchange further information securely. This creates an SA for IKE itself, although it's called an ISAKMP SA. This special bidirectional SA is used for Phase 2.

Phase 2—Now the ISAKMP SA is used to create the *other* SAs for the two devices. This is where the parameters such as secret keys are negotiated and shared.

Why two phases? Phase 1 typically uses public key encryption and is slow, but technically only has to be done once. Phase 2 is faster and can conjure different but very secure secret keys every hour or every 10 minutes (or more frequently for very sensitive transactions).

QUESTIONS FOR READERS

Figure 29.10 shows some of the concepts discussed in this chapter and can be used to answer the following questions.

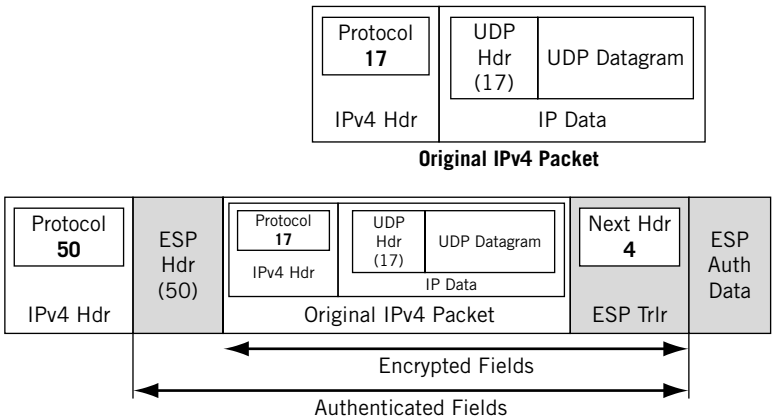


FIGURE 29.10

IPSec ESP used with an IPv4 packet.

1. Which IPSec ESP mode is used in the figure—transport or tunnel?
2. Which IP protocol is being tunneled?
3. What does the ESP trailer next header value of 4 indicate?
4. Could NAT also be used with IPSec to substitute the IPv4 addresses *and* encrypt them?
5. Is the SPI field encrypted? Is it authenticated?