5

Virtual Private Networks

5.1 Introduction

In Part 2, we discuss *virtual private networks* (VPNs). Chapter 4 had some examples of VPNs, but we were interested mainly in their tunneling aspects and didn't dwell on their security and authentication features.

In this chapter, we define VPN and briefly revisit the VPNs from Chapter 4. In the rest of Part 2, we study several types of VPNs, see how they are used, and take note of their strengths and weaknesses. As we shall see, these VPNs can operate at any layer in the TCP/IP stack. As usual, we will be less concerned with the administrative details of configuring the VPNs than with developing an appreciation for the protocols themselves and the manifestation of those protocols on the wire.

Before beginning our discussion of VPNs, we should agree on a definition for them. We already have an implicit definition from our study of MPLS VPNs in Chapter 4. We might say that according to that definition, a VPN is a method of using tunneling to build a private overlay network on top of a public network in such a way that the security of the private network is equivalent to that provided by leased lines. But this definition suffers from a lack of precision as to the meaning of "security equivalent to that provided by leased lines" and is a bit too general for our purposes.

Instead, let us say that a virtual private network is an overlay network built with tunnels in which the tunnel payloads are encrypted and authenticated. Given that we use robust encryption and authentication, such a VPN would certainly provide security as good as or better than that provided by leased lines, so this definition is consistent, if more restrictive, than that for MPLS VPNs. The underlying notion of both definitions is that we are trying to create the illusion of a private network while using a public network, such as the Internet.

It's worth dwelling, for a moment, on the differences between a "real" private network and a virtual private network. As a first approximation, we could say that real private networks provide security by physical separation of the underlying communication media. Separate leased lines are dedicated to the network, and these leased lines carry traffic only for that network. This means that short of a physical wiretap, an attacker does not have access to network traffic.

As we saw in our study of MPLS VPNs, the security of real private networks does not necessarily depend on the actual separation of physical media. We can also achieve segregation of network traffic through routing, or by multiplexing several data channels onto a single physical cable.

MPLS VPNs are an example of providing a private network by using routing to ensure that a private network's traffic is delivered only to intended recipients. Because the assignment of the MPLS label, and thus the route, takes place within the MPLS cloud, an attacker on the edge of the MPLS network has no way to capture traffic from another private network or to inject packets into it.

In a typical transcontinental leased-line deployment, the customer is provided with a partial or whole T1 line. Even if the entire T1 line is dedicated to a single private network, the traffic from the user's T1 line is multiplexed onto a higher-bandwidth backbone, such as a T3 or OC4 line, for transport across the continent to the other endpoint's T1 line. Thus, traffic from the private network is carried on the same physical media as traffic from other networks. Nevertheless, from the point of view of a user of the private network, this data is inaccessible and as a practical matter does not exist.

Despite the realities of the previous paragraphs, our normal conceptual model for a leased-line connection is a dedicated wire from one site to another. This model includes the notion of a physical connection, and when we're told that the network is down, we imagine that a physical event, perhaps involving a backhoe, has taken place. A virtual private network, on the other hand, is just that: virtual.

As with a TCP connection, a VPN's tunnel is a purely notional construct consisting of shared state at the tunnel endpoints. When told that the VPN is down, our first thought is not that a cable has been cut but that the shared state has become desynchronized. Once one of the VPN's packets enters the Internet, it is like any other IP datagram in the Internet. A malefactor can use a flooding attack to cause a router to drop it or can inject phony packets into the VPN by forging some of the packet's header fields. To protect itself from these and other attacks, a VPN relies on encryption and authentication to secure its data.

The advantages of a VPN over an actual private network should be clear. Instead of expensive leased lines or other infrastructure, we can make use of the relatively inexpensive, high-bandwidth Internet. More important in many instances is the ubiquity of the Internet. In most developed areas, access to the Internet is readily available without special provisioning or long waiting times. Given a VPN with robust cryptographic primitives and protocols, we could argue that a VPN is, in fact, more secure than a dedicated leased line, even if we accept our conceptual model of such a line as real.

In our definition of VPN, we said that the tunnel payload is protected by encryption and authentication. As we study the various types of VPNs, we will see that the meaning of *payload* depends on the class of VPN. In Chapter 6, for example, we study SSL

tunnels, which operate primarily at the application layer. Thus, the payloads that they encrypt and authenticate are usually application data. At the other end of the spectrum, tunnel-mode ESP in IPsec (Chapter 12) operates at the network layer, so its payloads are entire IP datagrams.

5.2 PPTP

We studied the tunneling aspects of PPTP in Chapter 4, where we viewed it as a type of remote access server that doesn't require expensive capital expenditures on modem banks and telco lines. As we mentioned, users view PPTP primarily as a VPN technology—they perceive its main benefit as secure communications. In this section, we take a brief look at PPTP's VPN properties.

If we reexamine the PPTP message types, we see that none of them deals with encryption or authentication. That's because PPTP really *is* a tunneling protocol, not a VPN protocol. PPTP relies on the underlying PPP protocol for its encryption and authentication services: In practice, that means Microsoft's *Microsoft Point-to-Point Encryption* (MPPE) [Pall and Zorn 2001] and *Microsoft Challenge Handshake Authentication Protocol* (MS-CHAP) [Zorn and Cobb 1998].

At first glance, these seem like reasonable cryptographic protocols. MPPE uses RC4, which we saw and remarked on in Chapter 3. MS-CHAP is Microsoft's version of PPP CHAP, a typical challenge/response protocol. Unfortunately, Microsoft's implementation of these protocols has several problems.

The MS-CHAPv1 protocol has several weaknesses that make recovery of the user's password comparatively easy using a dictionary attack. MS-CHAPv2 fixes the worst of these weaknesses but is still susceptible to an attack where a dictionary of N trial passwords can be checked at a cost of about $N/2^{16}$ attempts and some precomputation.

The version of MPPE used with MS-CHAPv1 has a fatal flaw: It uses the same RC4 key for both the PAC and PNS, resulting in reuse of the key stream. As we saw in Chapter 3, this leads directly to the recovery of both plaintext streams. With MS-CHAPv2, Microsoft changed MPPE to avoid the key stream reuse, but serious problems remain. The main problem is that the RC4 session keys are derived in a deterministic way from the user's password and information passed in the clear. Thus, the keys have the same entropy as the password instead of the 128 bits that a randomly generated session key would have. Because user passwords are generally low entropy and, with MS-CHAP, susceptible to a particularly effective dictionary attack, PPTP cannot be considered to have robust encryption.

Finally, there is no per packet authentication. This means that various bit-flipping attacks are possible on the encrypted data.

> We can "flip," or change, any bit without detection when using RC4 and similar stream ciphers. If we know that a certain bit in a message is important for some reason, we can easily change it. Suppose, for example, that we know that the most significant bit of byte X in a message enables certain features that make it more difficult for an attacker to compromise the security of the system. If we know that the bit is turned on, we can turn it off by merely exclusive-ORing 0x80 with byte X of the ciphertext. Note that we don't have to know the encryption key

> or even the plaintext value of byte *X* to do this: only that we want to change the most significant bit. It's easy to see how we can generalize this to alter larger units of data if we know their plaintext values and positions in the data stream.

The PPP control protocols, such as LCP, are particularly vulnerable, and it may be possible to convince the client and server to use the older MS-CHAPv1 protocol, resulting in fairly easy user password recovery and compromise of the RC4 cipher stream.

The fact that MS-CHAPv2 considerably strengthened the Microsoft version of PPTP notwithstanding, almost all experts recommend against using PPTP. The specifics of the MS-CHAP and MPPE weaknesses are detailed in [Schneier and Mudge 1998] and [Schneier, Mudge, and Wagner 1999].

The weaknesses we have been discussing are specific to the Microsoft implementation, of course, but this implementation is the preponderant one. Virtually all other PPTP implementations are written to interoperate with Microsoft's, so they are likely to share the same weaknesses.

5.3 L2TP

L2TP has minimal built-in security. The LAC and LNS can authenticate each other during tunnel setup, and most AVPs can be encrypted, but L2TP, like PPTP, depends on PPP to protect the user data in the tunnel. This default security has several problems.

Let's put aside the problems with MPPE and MS-CHAP for the moment and assume that we have robust versions of CHAP and an encryption protocol installed in our PPP implementation. Unfortunately, we are still far from secure. Let's take a look at some of the problems that remain.

First, the control channel, once set up, is unprotected except for any AVPs that are encrypted. This means that an attacker can easily disrupt the tunnel by, for example, sending a forged StopCCN message to the LAC or LNS. The attacker will have to know the tunnel ID and proper sequence numbers, of course, but they are easily obtained by snooping the tunnel or even by informed trial and error.

A parallel weakness exists in PPP. Although packets carrying user data are encrypted, control packets, such as LCP, CHAP, and IPCP packets, are not. This means that PPP can leak information, such as the internal IP addresses of the enterprise network. It also means that active attacks, such as sending a forged packet pointing to a compromised DNS server, are possible. Because PPP packets are not authenticated—whether or not encryption is in use—these types of attacks are particularly easy.

This last point is an important one. Because neither PPP nor L2TP messages are authenticated, they are subject to various manipulations.

> We must distinguish here between the endpoint authentication, which happens at L2TP tunnel setup and PPP session establishment, and message authentication. Endpoint authentication typically involves a CHAP-like mechanism to convince each side of the proposed connection that its peer is who he says he is. Message authentication, on the other hand, refers to providing each message with a message authentication code, such as one of the HMACs we discussed in Chapter 3, to guarantee that the message is from the peer, not a forged message injected into the message stream by an attacker, and that it has not been modified since the

> MAC was calculated. Message authentication is sometimes called message integrity to distinguish it from endpoint authentication. Similarly, MACs are sometimes called *message integrity codes* (MICs).

Those who are not well versed in security and cryptographic protocols often believe that encryption alone provides message authentication. After all, if the message is not from the peer, how did the attacker encrypt it? Similarly, how could an attacker alter a message's plaintext without knowing the encryption key? Unfortunately, even encrypted messages are subject to manipulation by an attacker. Data that is encrypted with a stream cipher is subject to bit-flipping attacks, as we noted in our discussion of PPTP. Data encrypted with block ciphers is subject to cut-and-paste attacks, as we'll see in Chapter 12. Thus, the lack of message authentication in L2TP is a serious security shortcoming.

> As we mentioned in Chapter 4, the forthcoming L2TPv3 will provide optional authentication for all messages in the control channel.

A consequence of L2TP's lack of message authentication is that there is no replay protection. That is, an attacker can replay previous messages in order to confuse the protocol or end application. We would not, for example, want to allow a previous request for a bank fund transfer to be replayed. Replay attacks can easily be prevented by adding a sequence number to each message and then authenticating the message. Notice that the sequence number alone is insufficient, as the attacker could merely supply the expected sequence number of an unauthenticated message. It is the authentication that prevents such trivial attacks.

Another problem with L2TP and PPP encryption and endpoint authentication is that they are based on a single shared secret. There are two issues with this: First, this shared secret could be compromised in some manner outside of the protocol, and once compromised, all previous and future traffic can be read, and the enterprise network that it is protecting is subject to immediate attack. Second, the shared secret is long lived, allowing an attacker to accumulate a large amount of data encrypted with it. The more data that a cryptanalyst has, the easier it is to break the encryption and discover the key.

VPNs generally have more robust key-management protocols that change keys often and prevent the accumulation of data that a cryptanalyst can work with. MPPE does this to a limited extent—the session key can change with every packet or after 256 packets—but because these keys are derived from a single shared secret and information sent in the clear, their security is no better than that of the shared secret. If the shared secret is compromised, all messages—past and future—can be read.

The ideal key-management protocol has a property called *perfect forward secrecy* (PFS). This means that each session key is independent and that the compromise of one such key does not compromise any of the others. Thus, our complaint about MPPE in the previous paragraph is that although it does change keys frequently, thereby making brute-force attacks more difficult, it does not enjoy the PFS property. This is a problem with PPP in general; none of the encryption protocols defined for use with it enjoy PFS or have robust key-management protocols. Indeed, as of this writing, they all depend on a shared secret.

L2TP and IPsec

Because L2TP has such weak native security, many experts consider it a remote access technology rather than a VPN. On the other hand, users tend to think of it as a VPN technology and are interested primarily in protecting their communications. One way of reconciling these two views is to combine L2TP with an external security protocol. The most popular way of doing this is to run L2TP over IPsec (Part 3).

IPsec provides encryption, authentication, and other security services at the network layer. IPsec can run in several modes and provide differing security services, but for now, let us merely stipulate that IPsec can provide encryption and authentication for IP packets. In particular, we shall be interested in ESP transport mode (Chapter 12), in which the payload of IP datagrams is encrypted and authenticated while in transit between, in this case, the LAC and LNS. Figure 5.1 shows the encapsulation of the L2TP header and message within IPsec, UDP, and IP.

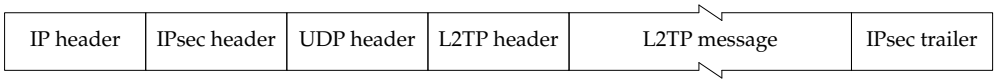

Figure 5.1 Encapsulation of L2TP Within IPsec

With this encapsulation, the UDP header, the L2TP header, the L2TP message, and parts of the IPsec header and trailers are encrypted and authenticated, giving complete protection for everything in the L2TP tunnel. That is, the control channel and all the data channels, including the PPP control protocols, such as LCP and IPCP, are protected from snooping and alteration. This solves the problem that we have with plain L2TP of leaving these protocols unprotected. Finally, ESP provides replay protection, so L2TP/IPsec is also safe from replay attacks.

Given a connection over an L2TP/IPsec tunnel from a remote host to a host on the enterprise network, it is important to understand what parts of the path between the two hosts are protected. In the most common case of the road warrior with a voluntary connection to the enterprise network, we have the situation shown in Figure 5.2. If we assume that the remote host is communicating with one of the servers on the enterprise network, we almost have an end-to-end VPN. The L2TP/IPsec tunnel ends at the LNS on the enterprise network, so the final hop to the server is unprotected, but this won't matter for most applications, because the LNS and the server are both on the enterprise network, which is presumably protected by firewalls and other means from outside interference. Notice though, that the connection is still subject to snooping or manipulation by a host inside the enterprise. If this is a concern, another VPN could be established between the LNS and the server.

Another typical application of an L2TP/IPsec VPN is shown in Figure 5.3. Here, a remote network, such as a branch office, is connected to the enterprise network through an L2TP/IPsec tunnel. We assume that a host on the remote network has a connection to a server on the enterprise network.

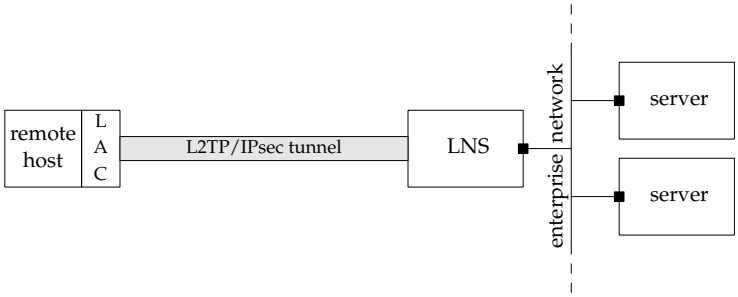

Figure 5.2 An L2TP/IPsec VPN Between a Road Warrior and the Enterprise Network

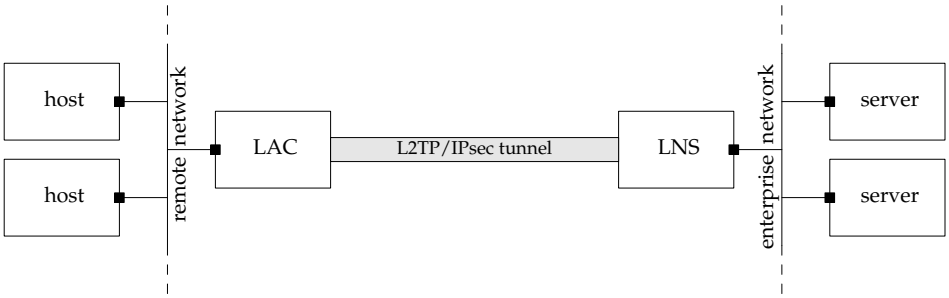

Figure 5.3 An L2TP/IPsec VPN Between Networks

The security situation at the enterprise network is exactly like that in Figure 5.2: The connection is unprotected from the LNS to the server. The situation at the remote network is a little more complicated. If we assume that the host is talking to the LAC over a PPP connection, we can arrange that the data between the host and LNS is encrypted by PPP. That means, of course, that the data is doubly encrypted over most of the path between the host and the server, but more important, there is limited protection for the conversation on the remote network. Thus, such a connection would be resistant to snooping by a host on the remote network.

Other configurations are possible, of course, so it is important to evaluate the security of each leg in any proposed L2TP/IPsec VPN topology. If we trust the hosts and servers on the private networks, we needn't be too concerned about the lack of security there. If we don't trust them, we must take steps to ensure that any sensitive data is protected from snooping or alteration from within the private networks.

Microsoft uses the combination of L2TP/IPsec as its VPN solution (replacing PPTP), so we can expect to see this topology frequently. Nonetheless, not everyone agrees that this is a good solution.

First, it reintroduces the NAT problem. Recall that L2TP uses UDP encapsulation rather than GRE, as PPTP does, and that the rationale for this was that it allowed L2TP

to interoperate with NAT. When L2TP is combined with IPsec, the UDP header is encrypted and thus unavailable to NAT. This means that the most common NAT mode, PAT, cannot be used. Fortunately, this problem is being solved. The IETF is developing a standard for a technology, called *NAT traversal* (NAT-T), that allows IPsec to interoperate with NAT. The Microsoft implementation of L2TP/IPsec includes a version of NAT-T. We discuss NAT-T further in Chapter 14.

The other common complaint about L2TP/IPsec is that it's a solution in search of a problem. Critics complain that L2TP not only adds overhead and does not scale well, but also fails to solve any problems that IPsec alone can't solve. Microsoft and other L2TP adherents counter that L2TP enables the use of existing session authentication protocols, such as MS-CHAP, and makes use of PPP's ability to assign IP addresses and DNS servers to the remote host. [Messmer 2000] discusses the pros and cons of L2TP/IPsec deployment. Chapter 10 of [Shea 2000] has an excellent summary of L2TP security concerns and a discussion of L2TP/IPsec. RFC 3193 [Patel, Aboba, Dixon et al. 2001] discusses securing L2TP with IPsec in detail.

5.4 Other VPNs

In the remainder of Part 2, we examine several other VPN technologies. We begin with examinations of the SSL/TLS (Chapter 6) and SSH (Chapter 7) protocols. Because these protocols operate at the application layer, some might consider them merely secure applications and not real VPNs. We will see, however, that they meet our definition for a VPN, and that we can, in fact, use them to build traditional network-to-network VPNs.

Regardless of whether SSL/TLS and SSH are "real" VPNs, they solve the central problems of privacy, authentication, and key management that every VPN must address. By studying their solutions to these problems and by noting where they succeed and where they fail, we will gain a deeper appreciation for both the problems and their solutions. Indeed, we will see that the design sets we introduce in our examination of SSL/TLS and SSH are used again and again in other types of VPNs. For these reasons alone, our study of these protocols will pay handsome dividends.

Next, we introduce and study some lightweight VPN technologies. When we say that they are lightweight, we mean that they are simpler and easier to deploy than, for example, the more comprehensive IPsec protocols (Part 3). In some applications—especially ad hoc applications—using one of these protocols might make sense.

We begin with an examination of VTun, a very simple VPN that illustrates the difficulty of engineering robust security protocols. At the same time, VTun also illustrates the use of a common framework for building lightweight VPNs. Because of its simplicity, VTun exposes this framework in a way that makes it easy to see and understand.

After VTun, we take a quick look at CIPE, a VPN running only on Linux and Windows NT. Because CIPE depends on a kernel module, porting it to other platforms is difficult. As we'll see, CIPE solves some of the security problems in VTun but still has flaws.

Next, we examine tinc, a VPN using the same framework as VTun. We'll see that it solves most of the problems that CIPE did not resolve. Tinc is interesting because it is designed as a *network* of VPNs, where a set of tinc gateway nodes securely connect a series of networks by maintaining encrypted tunnels between the nodes. Within this network, tinc manages routing and the decryption and reencryption of IP datagrams as they pass through intermediate nodes to their destination node.

Finally, we study OpenVPN, an excellent VPN that appears to offer security comparable to that of IPsec. OpenVPN achieves this by reusing the TLS/SSL protocol (Chapter 6) for endpoint authentication and key exchange, and by closely mimicking ESP (Chapter 12) for its data channel. Although it uses the same simple framework as VTun and tinc, OpenVPN provides robust security by leveraging the proven SSL and ESP protocols.

Again, the study of these VPNs will deepen our appreciation for the problems that all VPNs must solve. It will also help us to understand their limitations and enable us to make informed decisions as to whether they are appropriate for any given application.

5.5 Summary

In this chapter, we agreed on a definition of VPN and acknowledged that even with our definition, there can still be disagreement about whether a particular technology is a VPN or merely a secure application. We saw how this definition applies to PPTP and L2TP and examined the extent to which those protocols provide a robust VPN.

We ended the chapter by providing a road map for the rest of Part 2. We indicated that we will study SSL, SSH, and four lightweight VPNs.

Exercises

5.1 Verify the statements concerning bit flipping in stream ciphers. Specifically, show that exclusive-ORing a 1-bit into the ciphertext changes the corresponding plaintext bit.

5.2 Would it make sense to reverse the L2TP/IPsec encapsulation? That is, what would be the advantages and disadvantages of establishing an IPsec connection between a host on a remote network and a server on the enterprise network, and then running that through an L2TP tunnel between the LAC and LNS? The topology would be similar to that in Figure 5.3, but IPsec would be encapsulated in L2TP messages instead of the other way around.

5.3 Make the argument that a VPN is more secure than a leased line.

5.4 Given a VPN with robust cryptographic primitives and protocols, or a dedicated leased line, what are the most vulnerable points in the network? That is, if a malefactor were tasked with compromising the network's data, how should the attacker proceed?