12

Tunneling and VPNs

"Because of the Alderson drive we need never consider the space between the stars. Because we can shunt between stellar systems in zero time, our ships and ships' drives need cover only interplanetary distances. We say that the Second Empire of Man rules two hundred worlds and all the space between, over fifteen million cubic parsecs. . .

"Consider the true picture. Think of myriads of tiny bubbles, very sparsely scattered, rising through a vast black sea. We rule some of the bubbles. Of the waters we know nothing. . ."

Dr. Anthony Horvath in *The Mote in God's Eye*
—LARRY NIVEN AND JERRY POURNELLE

The Internet offers the potential for IP connectivity between almost any pair of computers, except where they are blocked by a firewall or the moral equivalent thereof. This connectivity is both a bug and a feature. It is extremely convenient to use the Internet as transport in many situations. For example, instead of making a long-distance call to connect to a home server, it is cheaper to make a local call and then use the "free" Internet to connect back. Conversely, a direct dial-up line is a safer way to communicate with your server: phone lines are harder to tap.

Enter the *tunnel*. Tunneling is defined by Yuan and Strayer as "an architectural concept in which one or more protocol layers are repeated so that a virtual topology is created on top of the physical topology" [Yuan and Strayer, 2001]. This is overly restrictive. We use the word to include any *encapsulation* of one protocol within another. Less formally, a tunnel is any sort of virtual wire that is somehow implemented over the Internet. In this chapter, we take a broad look at tunnels and *virtual private networks (VPNs)*. Tunnels don't always use cryptography, but they usually should. If you would like a peek at the details of the cryptography involved in tunneling, the inner workings are explained in Section 18.3.

Think of a tunnel as a special, high-tech channel that can connect various services, programs, hosts, and networks through an existing internet without running new wires.

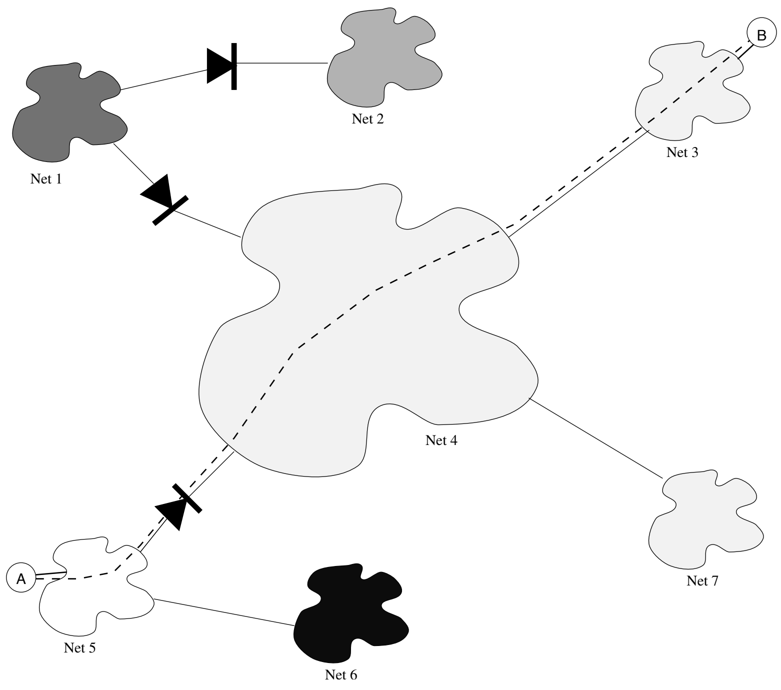

Figure 12.1: Tunneling past a firewall.

12.1 Tunnels

Think of a tunnel as a funky link layer. It's below IP, but generally recurses, with a bit of glue (preferably secure glue) in between. That makes the tunnel a fundamental building block for enabling security on the Internet: It lets you avoid obstructions and hide when traversing dangerous places. As such, tunnels are discussed in various places throughout this book.

12.1.1 Tunnels Good and Bad

Although firewalls offer strong protection, tunnels (see Figure 12.1) can be used to bypass them. As with most technologies, these tunnels can be used in good or bad ways.

As previously stated, tunneling refers to the practice of *encapsulating* a message from one protocol in another, and using the facilities of the second protocol to traverse some number of network hops. At the destination point, the encapsulation is stripped off, and the original message is reinjected into the network. In a sense, the packet burrows under the intervening network nodes, and never actually sees them. There are many uses for such a facility, such as encrypting links, connecting branch offices to headquarters without running a new network, telecommuting, and supporting mobile hosts. More uses are described in [Bellovin, 1990].

IP packets can be tunneled in several ways. They are quite often encapsulated within themselves: *IP over IP*. Another important one is *Point-to-Point Tunneling Protocol (PPTP)* or its IETF variant, *Layer Two Tunneling Protocol (L2TP)* [Townsley *et al.*, 1999].

The point is that IP may be tunneled in many parts of its own protocol suite. That is the situation we are concerned about here. If a firewall permits user packets to be sent, a tunnel can be used to bypass the firewall. The implications of this are profound.

Suppose that an internal user with a friend on the outside dislikes the firewall and wishes to bypass it. The two of them can construct (dig?) a tunnel between an inside host and an outside host, thereby allowing the free flow of packets. This is far worse than a simple outgoing call, as incoming calls are permitted as well.

44 Almost any sort of mechanism can be used to build a tunnel. At least one vendor of a *Point-to-Point Protocol (PPP)* package [Simpson, 1994] supports TCP tunneling. There are reports of *telnet* connections and even DNS messages being used to carry IP packets. Almost any gateway can be abused in this fashion (but see RFC 1149 and RFC 2549 [Waitzman, 1990, 1999]). Even pairs of FTP file transfer connections can provide a bidirectional data path. An extreme example of tunneling is Microsoft's *Simple Object Access Protocol (SOAP)*, which can be used to wrap any arbitrary content over HTTP, a protocol that is permitted by many firewalls. In fact, the use of SOAP by peer-to-peer systems, such as Groove Networks, is becoming quite common.

SOAP has been submitted to the *World Wide Web Consortium (W3C)*. The document[1] identifies privacy and integrity as important security concerns, but does not address them. Authentication, perhaps the most important in this context, is not mentioned. The protocol is, fundamentally, RPC over HTTP. That makes it hard for proxies to filter it safely. Such attempts by vendors to evade firewalls are irresponsible. The right path for vendors is to make protocols open and easy to analyze, and to document the security implications of opening the port(s), not to evade the administrator's security policy. A related example is the *Internet Printing Protocol (IPP)*. It uses HTTP, but over port 631. The designers wanted it to run on port 80, and in fact, that is happening. Because port 80 is open on many firewalls, vendors have taken advantage and multiplex traffic over that port. If every protocol were to do that, the firewall would be of little use, basically only filtering on addresses. (Some of the blame is shared by administrators who reflexively say "no" without analyzing the business case. If this keeps up, we'll end up in a world of firewalls as placebos, installed in a network to make upper management feel secure.)

The extent of the damage done by a tunnel depends on how routing information is propagated. Denial of routing information is almost as effective as full isolation. If the tunnel does not leak your routes to the outside, the damage is less than might be feared at first glance: Only the host at the end of the tunnel can access the Internet. But it does provide a hacking target. If it

1. See `http://www.w3c.org/TR/SOAP/`.

has weaknesses, someone from the Internet can connect to and conquer it, and then access your intranet from that host. This is a *host leak*, discussed below.

Conversely, routing filters are difficult to deploy in complex topologies; if the moles choose to pass connectivity information, it is hard to block them. On the Internet, the routers generally filter incoming route announcements. Failure to do so has caused all kinds of mayhem in the past, so most ISPs are pretty good about keeping an eye on this. Thus, if your internal networks are not administratively authorized for connection to the Internet, routes to them will not propagate past that point, but they may be widely known within a wide-open organization, such as a university.

Often, such a tunnel can be detected. If you are using an application- or a circuit-level gateway, and an external router knows a path to any internal network except the gateway's, *something* is leaking.

Standard network management tools may be able to hunt down the source, at which time standard people management tools should be able to deal with the root cause. Unauthorized tunnels are a management problem, not a technical one. If insiders will not accept the need for information security, firewalls and gateways are likely to be futile. (Of course, it is worth asking if your protective measures are too stringent. Certainly that happens as well.)

Host leaks often occur at the ends of tunnels. They can often be detected with specially designed spoofed packets sent to one end of the tunnel. Packets finding their way through the tunnel may be collected by hosts connected to the other network.

Once suspected or spotted, a tunnel should be monitored. We know a number of cases in which hackers have actively used unauthorized tunnels, or abused authorized ones. In September 2000, a tunnel into Microsoft made front-page news. Others companies have made similar but less-publicized discoveries.

Tunnels have their good side as well. When properly employed, they can be used to bypass the limitations of a topology. For example, a tunnel could link two separate sites that are connected only via a commercial network provider. Firewalls at each location would provide protection from the outside, while the tunnel provides connectivity. If the tunnel traffic is cryptographically protected (see Section 18.3), the risks are low and the benefits high.

45 Note that such tunnels may be subject to denial-of-service attacks even if the cryptology and implementation are perfect. The protected packets pass through an untrusted network. A swarm of packets from other sources could choke the channel over which the tunnel flows.

If the connection is vital, don't use a public network.

12.2 Virtual Private Networks (VPNs)

A private network used to be defined as a set of computers protected from the Internet via a firewall (if they were connected at all.) These machines were more secure from outside attack because the money, expertise, and paranoia were all focused at keeping the gateway secure. Sites with multiple locations had to be linked privately: The Internet did not offer services secure enough to link locations. Each site was protected by a firewall, but there was no way for machines at different locations to communicate securely. In fact, due to the firewall, it was unlikely that they could communicate at all.

Virtual private networks extend the boundary of a protected domain through the use of cryptography. There are three kinds of VPNs. The first type enables remote branch offices to share a security perimeter and even an address space. The second is used by companies that do not wish to open up their entire networks to each other but wish to have some shared services and data—these VPNs implement a DMZ between them. The third kind enables remote users to connect to their work location from home, hotel, or coffeeshop.

12.2.1 Remote Branch Offices

So, you did not learn your lesson yet, and you decided to start your own Internet software business. Surprisingly, you have been quite successful, and you now have offices around the world. The sales and marketing departments are headquartered in New York, development is split between Silicon Valley and India, and product packaging and shipping takes place in Memphis. You decide to buck convention and attempt to establish close ties between software development and marketing. Marketing folks need to see demos of the latest releases of the products, and the techies would like to see the latest business plan so that they can try to at least simulate the promised features. Travel budgets are tight, and the two development arms need to be able to share file systems and their environments, as well as video conference using NetMeeting. At the same time, you would prefer not to share your business plan and development code with the rest of the world.

This is a not uncommon scenario, and it screams for a VPN solution. Once you've established your security policy, it is time to enforce it. The best way to do this is to define each remote security perimeter, and deploy firewalls, intrusion detection, and network monitoring. Once New York, San Jose, Bangalore, and Memphis are online, VPNs can be established between the different locations. The most common and sensible way to do this is to set up firewall-to-firewall tunnels. The tunnels should use IPsec in tunnel mode to encapsulate IP packets, as described in Section 18.3.

When a machine in Memphis sends packets to a machine in New York, local routing gets the packets to the firewall in Memphis. There, the packets are encapsulated, encrypted, and MACed (see Appendix A). The packets are sent over the Internet to the firewall in New York. There they are unencapsulated, decrypted, and verified. Finally, local routing in the New York network gets the packets to their destination. When the packets travel through the Memphis and New York networks, respectively, they are unprotected, as these are (presumably) trusted networks. When the packets travel in the wild, IPsec ensures that they are not tampered with, and that the contents are not exposed.

Figure 12.2 shows how a packet flows from one remote branch to another by tunneling from firewall to firewall.

If you use the same address space in all of your locations, applications can access remote resources just as easily as ones in the same physical location. This can be handy for such things as mounting file systems, remote login, and video conferencing.

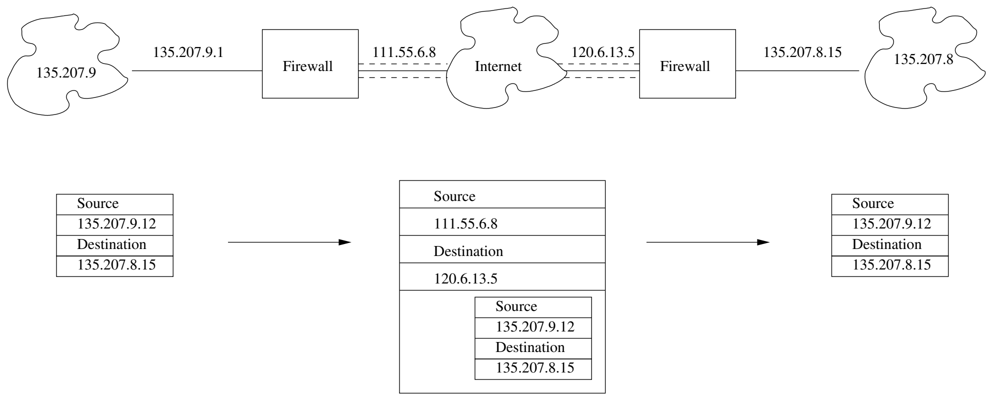

Figure 12.2: Two branch offices tunnel from firewall to firewall. The firewall corresponding to the source encapsulates the packet in a new IP packet, destined for the destination firewall. The receiving firewall unencapsulates the packet and sends it to the private-side destination.

12.2.2 Joint Ventures

VPNs can be used when two organizations wish to give each other limited access to resources while excluding the rest of the world. For example, two companies may wish to share several databases. Each company can dedicate a part of the network that is partitioned from the rest of the network by a firewall. The two companies can then exchange keys and use a VPN to link the two private networks containing the databases.

The nice thing about a VPN is that it can be configured to a fine level of granularity. For example, company A can enforce a policy that company B may access a database for a certain amount of time, or at certain hours during the day. Application-level VPNs, such as tunneling over *ssh*, also exist, and they can be used to make user-level access control decisions. Managing a VPN is not unlike managing a firewall in that respect. IPsec-based VPNs are analogous to packet-filtering firewalls, and application-level VPNs resemble application level gateways.

When we wrote the second edition of this book, we used *cvs* over *ssh* as a secure solution for keeping copies of the manuscript up-to-date. The master copy was kept on a server in one of our homes. We all had logins on the machine, but we all used *cvs* to check copies in and out. At the same time, this configuration did not permit the authors access to any other part of that home network, nor did the setup give access to any of our employers' networks.

> Of course, we all had shell access on this server machine. We could have abused it to create tunnels or other mayhem. *At some point, security comes down to personal trust.*

A joint venture can be in only one direction, whereby one entity provides access to another; or it can be mutual, in which case, it is not unlike the branch office example given above.

12.2.3 Telecommuting

Telecommuting from home is common. Telecommuters save the time and trouble of the commute, but sacrifice the personal interactions at work. This book was written mostly at home or in hotel rooms, where isolation helped the writing process.

A home network is easy to install: An Ethernet is just some twisted-pair wires leading into a cheap hub; wireless cards and access points are showing up in homes where lath, plaster, and inconvenient beams frustrate hardwired installations. Fast Ethernet cards are ridiculously cheap, and employers have found that it is worth paying for a data line into the home: workaholics are often pleased to use the line to work a few extra hours at night or on weekends. Home networking is sure to put a strain on many a marriage.

How should a home network be linked to work? There are two options: link to work, or link to a nearby ISP and run through the Internet. Either link may be run over a hardwired line of some sort (leased line, cable modem, ADSL modem, dark fiber, packet radio, satellite base station, African or European swallow, and so on) or through a dial-up line (analog, ISDN, and so on).

One of the more important issues to consider when connecting remotely over a VPN is how DNS is employed. When networking through someone else's network, it is clearly unacceptable to rely on the ordinary external DNS for name resolution within the organization. At a minimum, DNSsec (see Section 2.2.2) must be employed. Although DNSsec may not be adopted globally due to PKI issues (see Section 7.10), it can and should be adopted locally within organizations. IPsec, after all, protects conversations at the IP level; if a host is deceived about the IP address of its peer, it will set up a secure conversation to the enemy.

But DNSsec does not solve the problem of concealing internal host names. If that type of protection is desired, a split DNS of the type described in Section 10.1.1 can be used. This scheme is problematic for laptops and the like; they live on the outside, and will normally see only the external DNS.

The solution here lies in the nature of the tunnel that is set up between the telecommuter's machine and the firewall. If, once the tunnel is established, all DNS queries are resolved via an internal server, the internal version of the organization's DNS tree will be used. Better yet, configure the tunnel so that *all* traffic from the user's machine flows *only* through the tunnel. That way, the machine will have all of the privileges, protections, and restrictions of any other inside host. That is, the firewall will protect the laptop against attack; similarly, corporate restrictions on outbound connections can be enforced.

There are some disadvantages to the configuration. Traffic from the telecommuter to the outside takes the scenic route, via the corporate firewall. For low-bandwidth, dial-up users, that isn't a serious loss; as higher-speed connectivity becomes the norm, it will be a more pressing point. (This paragraph was written from a hotel that provides Ethernet connectivity to guest rooms.) Worse yet, the protection is deceptive; both before and after the tunnel exists, the laptop has unrestricted connectivity to the Internet. And this, in common with other forms of serial monogamy, has its share of risks.

Address assignment for these machines is a related issue. Suppose, as is generally the case, that dial-up users are assigned more-or-less random IP addresses each time they connect to an ISP. How can the central site route packets to them?

If the firewall and the encrypting gateway are combined in one box or operate in series, addressing isn't a problem. But if the two operate in parallel, measures must be taken to ensure that outbound packets destined for such computers reach the encryptor, rather than the firewall.

The most obvious solution is to have the encryptor advertise routes to those destinations. That's messy and unpleasant. For one thing, those would be host routes, potentially very many of them. For another, there's no obvious way to know when to stop advertising a route to a given laptop. The obvious answer is "when it has disconnected," but the encryptor has no way of knowing when that occurs, short of continual *ping*s—another messy possibility.

A better strategy is to assign such telecommuting machines internal addresses on a given subnet. Then, a static route for that subnet can be established. It would be nice if these machines' addresses were dynamic; unfortunately, that is difficult at present. For one thing, there is again no way to know when an address can be reassigned, though here at least it can be bound to the lifetime of the security associations set up. More seriously, no standard protocol exists to transmit such assignments. When one is devised and deployed—a security equivalent, in a sense, to DHCP—it will be the preferred choice. Conversely, the computer needs a route to the encryptor for all internal networks, as opposed to the rest of the Internet. Again, this is difficult for large organizations. It's much easier if all Internet traffic is routed back through the home network; that way, a simple "default" route suffices.

(Similar considerations apply when two or more networks are connected via IPsec. Here, though, it may be impractical to coordinate address assignments; as a result, each partner must have a list of the networks assigned to the others. Fortunately, these are network routes, not host routes; furthermore, there are no concerns about address lifetime.)

A more general case is when IPsec is widely available. What if a random inside machine wishes to make a protected call to an outside peer—for example, to a Web server? A typical firewall today might permit such outgoing calls, and use the TCP header to distinguish between reply packets and an attempt to initiate a new incoming call *from* the Web server. With IPsec in place, that becomes quite problematic. The firewall will not be able to distinguish between the two cases, as the TCP header will be encrypted. But relying on the client host to protect itself contravenes the entire rationale for the firewall.

At this time, there are no good solutions to this dilemma if you use traditional firewalls. At best, such uses of IPsec can be barred, except to trusted machines. A client wanting a connection that is encrypted even on the internal network would employ IPsec to the firewall; it in turn would encrypt it the rest of the way. This means that the packets must be in the clear on the firewall, a disadvantage.

A better solution is per-connection keying. That is, every time a new TCP connection is established, a new key, and hence a new *Security Parameter Index (SPI)*, is allocated. The firewall could simply keep track of inbound versus outbound SPIs; a packet encrypted with the wrong SPI would be dropped, because its port numbers would not match those bound to the SPI. Unfortunately, most current IPsec implementations do not support such fine-grained keying.

The best answer is to use a full-blown distributed firewall, as discussed in Section 9.5.

For now, the most likely scenario is the trusted machine case. That is, end-to-end IPsec would be used only to machines on the VPN. In that case, there would be two layers of IPsec, one end-to-end, and one between the firewall and the outside machine.

When IPsec becomes ubiquitous, hosts will face another challenge: learning the address of the proper IPsec gateway for a destination. A possible solution is some sort of DNS "key exchange" (KX) record, analogous to MX records; see [Atkinson, 1997] for one proposed definition. But there are some serious problems that must be resolved. Clearly, KX information must be strongly authenticated, perhaps via DNSsec. In fact, DNSsec must be used to indicate that no KX records exist; otherwise, an enemy may be able to force a conversation into the clear, by deleting the proper KX response.

If the DNS is used for KX records, there is another problem: mapping the KX hierarchy to the topological relationship between the two endpoints. For example, a host outside the firewall that wants to talk to an inside server should use the firewall as the IPsec gateway; a host on the inside that wants to contact the same server could speak directly. Again, a split DNS solution appears to be the best alternative.

Note that KX records can be spoofed by the firewall, in much the same way as other DNS records. This technique can be used to force IPsec sessions to terminate at the firewall, as described earlier. In this case, though, the DNSsec keying hierarchy must be spoofed as well.

It is also necessary to deal with the problem of multiple layers of IPsec. At this time, how best to communicate such policies to end systems is unclear.

Direct Connection to a Company

A connection to work makes the home network an extension of the corporate network. This places the home network "behind" the corporate firewall, which may make it a bit safer from attack by random strangers.

If the home machine is used strictly for official business, this arrangement is fine. If the kids cruise the Web, or the spouse works for a competitor, this link can be a problem for corporate policymakers. On the one hand, most institutions have clear policies limiting use of institutional facilities. On the other hand, information workers are expensive and often hard to hire and keep. Does it hurt the company if an employee's kid harmlessly cruises the Web during off hours, when network use is down?

The answer is *yes.* Children are like employees: They can do stupid things without understanding the security ramifications. They can acquire viruses, worms, foistware, and back doors.

Corporate policy might require that employees purchase their own link to the Internet for personal use. Now there is a new problem: There are two links into the home. If they are linked, intentionally or otherwise, the corporation has just acquired a new link to the Internet, possibly without firewall protection. A policy may state that this is not allowed, but this is difficult to police. Home LAN security is hard.

If the spouse (or other domestic partner or partners) works for another company, there may be a third link into the home. Is it reasonable to assume that the three connections will remain separate, especially if there are shared resources such as printers and wireless access points? In one case we know of, a home had two networked computers, each with a link to a separate company. Routing information propagated over these links. Both companies' routers learned of a "direct" connection between them, and started routing packets through the couple's home network. Most arrangements like this don't have such an obvious result.

Connecting Through an ISP

It is usually cheaper to connect to an ISP and reach the workplace over the Internet. Some people telecommute across a continent. One usually needs some sort of encryption and authentication to make this arrangement safe. IPsec and PPTP are popular options for connecting to the corporate firewall or a server inside; there are other cryptographically protected tunnel mechanisms as well. (PPTP is also an example of what can go wrong when a security protocol is implemented incorrectly [Schneier and Mudge, 1998], but it is still very popular.)

This way, the home can have separate clients for Mom, Dad, and the kids. Mom's client has the key to tunnel into her company, Dad's to his, and the kids' computers (if they have their own) have no keys. Numerous security problems are still possible, but at least the home network itself is outside of both corporate networks. Only a single client tunnels in. If the client is secure from the kids, spouse, and the Internet in general, then the company is safe as well. A standard client these days tends to be harder to invade than a server.

Note, though, that this raises another issue: Are the work machines *always* tunneled? If so, they're always inside the corporate net, and hence always protected by the corporate firewall. But then how does the employee do recreational Web surfing? Saying "naughty, naughty, don't do that" doesn't work, but that's what most corporate policies demand. Of course, that leaves the machine naked to the Internet, and that same machine will soon be tunneling back in.

Networking on the Road

When you are on the road with a laptop, you typically have three networking options. You can dial in directly to your office; you can dial a local number and then tunnel across the Internet; or you can connect directly to a foreign (*i.e.,* hostile) network and tunnel across the Internet. As such, the technology for securing your networking is not that different from your options when networking from home.

The nature of these foreign accesses are changing. Many hotels now supply Ethernet connections in the rooms. These tend to use private address space and NAT to access the Internet at very nice rates. Some tunnels, notably IPsec, do not interact well with these arrangements. *Ssh* works well in this scenario.

Hotel lobbies, airports, and even some coffeeshops offer 802.11 wireless service, often free. Again, these often employ NAT and private address space.

Finally, in high-tech areas and cities, one can war drive: travel around looking for wireless base stations with DHCP, and steal their Internet access. There are local co-ops that offer this service for free; these have the advantage of being legal.

None of these networks are trustworthy by themselves; end-to-end encryption is the only safe way to use them.

12.3 Software vs. Hardware

Up to this point, we have discussed how tunnels work and what we mean by virtual private networks. Now it's time to look at how we instantiate this idea. There are basically two options: You can run the tunneling software on your machine or you can attach a separate hardware device between your machine and the network. Each option has its share of advantages and disadvantages.

12.3.1 VPN in Software

The main advantages to running a VPN directly on your client machine are flexibility, cost, and convenience. You do not need to do any wiring, or lugging around of any external hardware device. Software cryptographic processing has been one of the chief beneficiaries of steadily increasing CPU speeds; most strong crypto (symmetric operations) is unnoticeable on modern CPUs when supporting a single user. Of course, as hardware VPNs get smaller and the form factor improves (e.g., PC card VPN boxes), this advantage diminishes.

Software VPNs offer flexibility in the choice of protocol. An IPsec VPN in software may consist of a *shim* in the protocol stack between the IP layer and the link layer. Another option is to set up an *ssh* tunnel and forward IP packets over the protected connection. You may choose to have both options on your laptop and decide which one to use based on what you want to do.

Having the software on the machine puts the user in control of it. This may or may not be good, depending on the user's level of sophistication. If you've ever tried to configure a software IPsec product on a PC, you know that there are many more ways to do it wrong than right, and that you need a pretty good understanding of the protocol to get it working. Most users are not at this level, nor should they have to be.

IPsec carries a promise of strong security without user hassle. Although authentication can annoy a user, there is really no excuse for IPsec to be difficult. The crypto portion of security, at least, ought to be easily hidden behind the scenes.

One problem that arises with software IPsec is that many Windows applications reinstall portions of the IP stack. So, for example, you can have your secure tunnel up and running. Then, after installing a new financial package, you suddenly find that IPsec is no longer working. That package may have installed its own communications and broken your IPsec configuration.

They aren't trying to be malicious when they do this, but their priority is ensuring that you can talk to them, preferably without invoking their help desk. Reconfiguring your network to talk securely to theirs is often the easiest way to proceed. (This points out another problem with software VPNs: they don't compose very well.)

However, you do not need to install a financial package to get a Windows machine to stop working. While attempting to work with a well-known IPsec software package on a Windows machine, we found that the software itself took care of crashing the computer. In fact, uninstalling the software did not correct this, and ultimately, we had to reformat the disk to fix the problem. At that point, we switched to a hardware solution.

One final note about software VPNs regarding security. If your crypto software is running on the same machine that you use for everything else, it is exposed to viruses and Trojan horses. In Windows, this means that a descendant of Melissa and the Love Bug could potentially extract your IPsec or *ssh* private keys and disclose them to an adversary. Many modern viruses and worms disable your security software. A hardware solution is not totally immune to these, but if the VPN hardware is protected from the client to some degree, the probability of exposure is much less.

12.3.2 VPN in Hardware

Many of the advantages of VPN hardware can be inferred from the preceding software section. Hardware solutions are more secure because they do not expose sensitive information to the client to the same degree as software solutions. They are less likely to crash your machine. They also offer a different kind of flexibility: You can set up an entire network behind a hardware VPN, and assume that the protection from this device is shared by all of the machines behind it. This makes administration of a site much simpler because only the one hardware device needs your attention. (It becomes a firewall for the site.)

At AT&T Labs, we've set up a hardware VPN solution for telecommuters [Denker *et al.*, 1999]. Researchers contract with whatever local ISP they want for connectivity. Then, behind the DSL or cable modem, they connect a small box that we call a *YourKey,* which contains two Ethernet interfaces and a flash card. Some versions support a modem and/or an 802.11b card. Inside is a StrongARM processor running Linux. The whole thing weighs less than a pound. Users connect one of the Ethernet interfaces to the modem from the ISP (the wild side), and the other to a PC or to a hub. The flash card contains the users' keys, and the YourKey provides an IPsec tunnel to a back-end server on our firewall that handles all of the connections.

Remote administration of the YourKey is done via *ssh*, and there is no other way to talk directly to the box. The system works very well and allows researchers to use the internal address space in their houses. The YourKey can be taken on the road to provide IPsec tunneling from virtually anywhere. It even works through NAT.