CHAPTER 1

INTRODUCTION TO NETWORK PROTECTION

When was the last time you used a computer that was not connected to a network? I am not talking about using your laptop on an airplane. That is a planned disconnection from the network, and we can copy necessary information to the machine before we disconnect. We are talking about getting to a hotel room only to discover that they do not have a high-speed Internet connection or a data port on the phone (and the phone cable is connected using screws, inside the phone!); about moving into a new house before you get the cable modem or DSL line installed; about the network going down unexpectedly. Remember that feeling of helplessness? That feeling that the computer in front of you is just a pile of useless plastics and silicon, more useful as a boat anchor than a business (or entertainment) tool.

In today's environment, a computer that is not connected to the network is about as useful as a car without gasoline. It is pretty. The stereo still makes cool sounds—until the battery dies. The seats even lean back, but the car does not exactly do very much.

A computer today is only as useful as the network(s) it is attached to. This book is about how to protect the network and the computers attached to it so that you, its rightful owner or operator, can get maximum benefit out of it. In the end, information technology is most valuable when it is used to aggregate data from multiple sources, perform some really interesting task with that data, and then share it with someone else. The infrastructure that makes this all happen is the network. Several years ago, Microsoft launched a marketing campaign themed around the "Digital Nervous System." The digital nervous system was the network. It sounds corny to those of us who do not spend all day thinking about how to sell something, but it does make some sense. The network is what allows data to flow from the place where it is stored to the place where it has some

impact. In the end, it is all about data; data that you convert into information and then share in such a way that you get maximum benefit from it. Network protection is about ensuring that the infrastructure where all this happens is available, that data and information does not leak into the wrong hands, and that the data and information arrives at its destination intact.

When we first proposed this book, someone asked, "So, is it a book about how to build a secure network?" Our answer was no. Network security as an end state is a pipe dream, an impossible reality that we cannot attain. We constantly get asked how to make a network secure, but that really is the wrong question to ask. The concept of "security" denotes some finite state, some end goal. "Security" is defined as "freedom from risk or danger; safety." It is obvious that "security" in computers can never attain this lofty goal. Computer security is more "management of risk." In fact, is *secure* or *security* the right word to even use? Nothing is truly secure or has security if we look at the true definition. Secure means you can stop working because the network is now secure. Network security is a process, a task description, not an end state. Put another way, security is a journey, not a destination. Therefore, we like to talk about network protection as the goal, and network security as a task description. The task (as shown in Figure 1-1) is to detect problems and, preferably before someone else does, respond to those problems in a way that prevents them from becoming security vulnerabilities. At that point, the process repeats, and we look for more problems to prevent.

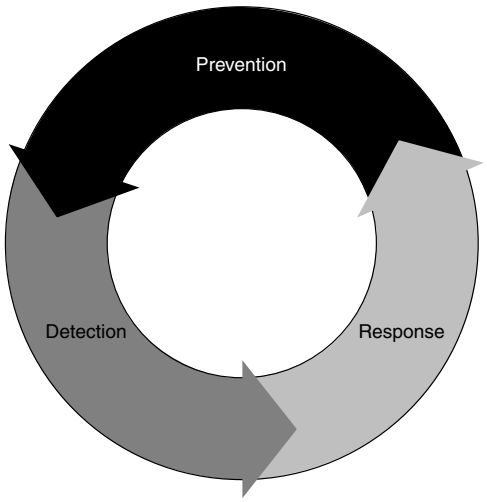

Figure 1-1 The security process.

NOTE: Note that this book is about the broader area of network security, not the more narrowly defined "distributed systems security." A distributed system is one where systems cooperatively share processing and data in order to appear to the user as a single system—in essence, abstracting a lot of the implementation details of the network design from the user. Distributed systems were popular in the 1990s, and we still get asked about the concept. Network security is a broader topic, because the network includes a lot more components. In addition, distributed systems lead to some interesting security problems that we address in more detail in Chapter 8, "Security Dependencies."

We often get asked the "big question" as our colleague Ben Smith calls it: Is my network secure? Contrary to Ben, however, we have not been able to make an entire 75-minute presentation out of it, because the answer is so simple: No. Your network is not secure. The state of being secure is typically considered an absolute. Consider the corollary: Can someone break into our network? Obviously, if the answer to that is no, your network is secure. The problem is that you can never conclusively answer no to that question, and because that is the bar we use for measuring whether something is secure, you will never have a secure network. You may have a "secure enough" network though. For the time being, know that we are aiming to protect the network, to have good enough security for our purposes. What does that mean? Well, it could mean a lot of things. One way to look at it is by comparing it to a car alarm. Does a car alarm make it harder to steal a car? No, not really. Even ignition killers can be bypassed easily by those who know what they are doing. Does it prevent theft? Well, that depends. If you have an alarm but the car next to you does not, it is likely that a thief may just steal the car next to yours (unless he really wants yours). It is kind of like the old story about a camping trip. Two guys are sitting by the fire and one of them asks what they will do if a bear comes. The other guy says, "That's why I am wearing sneakers." The first guy asks, "Do you really think you can outrun a bear though?" The second responds, "No, but I don't need to. I just need to outrun you!" In some cases, it is simply enough to be a more difficult target than someone else!

As long as people are not out to get you specifically, if you protect your network sufficiently, it is likely that the attackers will attack a network that is less secure, unless they really want something on your network. So, we face two challenges: protecting our network from the casual attacker or virus that does not care which network it destroys, and protecting our

network from the determined attacker who wants your information. The latter is definitely much more difficult. However, if you take some fundamental steps, you will have accomplished the former as well as make the job of the determined attacker much harder. This frees you up to focus on the rest of the job, which is staying far enough ahead of the determined attacker so your network, and the data on it, remains protected. In a sense, protection is like temporal security. It makes sure that you are secure until the bad guys learn enough to break down your defenses. At that time, you had better have additional defenses in place.

Why Would Someone Attack Me?

One key question we hear a lot is why someone would attack you. The people that attack networks and systems that do not belong to them are criminals, pure and simple. You will often hear the description of them couched in terms such as *script kiddie* and *hacker*, but why beat around the bush? They are criminals.

WHAT IS A HACKER?

We particularly dislike the use of the term *hacker* to describe a criminal. We both grew up in a time before the term hacker was appropriated by misinformed journalists to describe a criminal. A hacker is someone who is genuinely interested in computers and everything about them and who knows how to manipulate the machine in sophisticated ways, often even beyond what the designers envisioned and thought possible. Webster's dictionary offers several definitions of the term hacker:

> An expert at programming and solving problems with a computer
> A person who illegally gains access to and sometimes tampers with information in a computer system

The original definition of a hacker is the former. At some point in the late 1980s a misguided journalist co-opted the term, equating it with *cybercriminal*. This is particularly galling for those of us who have proudly called ourselves hackers since long before the term was appropriated and used to brand computer criminals. From now on, let it be known that when we use the term hacker, it is in the original, proud sense of the word, meaning someone who loves computers and tries to learn as much as possible about them.

The vast majority of people who attack networks today are not hackers under the original definition of that term—they are merely criminals. Therefore, the real explanation of why they do these things delves into the mind of criminals, and is best answered by a psychiatrist. However, there are several relatively obvious reasons. To understand the reasons, let us first look at the types of attacks you may see.

Types of Network Attacks

Essentially, network attacks can be distinguished on two dimensions: passive versus active and automated versus manual. A passive attack is one that uses network tools, such as a sniffer to capture network traffic, that simply listen on the network. These tools may capture traffic that contains sensitive information.

Active attacks, by contrast, are where the attacker is actively going after the protected resource and trying to get access to it, possibly by modifying or injecting traffic into the network.

On the other dimension, we have automated attacks. The vast majority of the attacks we hear about today are automated attacks, where the attacker creates some tool that attacks a network all by itself. The tool may have some intelligence built in, but fundamentally, if the network is not configured the same way as the one the tool was written for, the tool fails. Worms are methods of automated attack. In most cases, automated attacks are based on a known vulnerability in a system. The best method of defense against an automated attack is simply to keep the system fully patched at all times and monitor your network for suspicious events or messages. That is easier said than done, but Chapter 3, "Rule Number 1: Patch Your Systems," gives you some hints on how.

A manual attack occurs when the attacker is actually executing the attack without using automated tools. In this case, the attacker is actively analyzing the network and responding to its inputs. These types of attacks are much rarer, largely because the ratio of expert attackers to networks is relatively small. When we think of an attacker breaking into a network and stealing or modifying information, we are typically thinking of a manual attack.

Table 1-1 Types of Attack Against a Network

	Passive	Active
Automated	Hard to pull off, unlikely to generate much value	Reaches thousands of systems, but (relatively) easy to defeat
Manual	Sometimes fruitful, but takes longer than an active attack	Extremely dangerous, but rarer than the others

Consider these four types of attacks; we have four intersections, ordered roughly in order of severity from least to most severe, as shown in Table 1-1.

1. *Passive-automated*—This type of attack is usually some kind of sniffer that captures particular types of data. For instance, a keystroke logger that automatically sends data to the attacker falls into this category. So does a sniffer that captures and automatically replays an authentication sequence. It is pretty unlikely that these will generate a large percentage of useful data for the attacker, and it would require more skill than some of the other types that generate more access faster.
2. *Passive-manual*—In this type of attack, the attacker is just sniffing everything. A packet sniffer that logs everything falls into this category. We worry a lot about these attacks, but as discussed in Chapter 10, "Preventing Rogue Access Inside the Network," they are not nearly as important as we make them out to be. An attacker who can perpetrate these can usually, with some notable exceptions such as wireless networks, perpetrate other more serious attacks.
3. *Active-automated*—At first, it appears these attacks do not exist. How could an automated attack, such as a worm, involve an active attacker? However, into this category also falls attacks from attackers with sophisticated tools at their disposal. Most network worms fall into this category. For instance, a worm that searches for machines that are missing a particular patch, exploits it, and then uses the compromised machine to find additional targets falls into this category. Another example of this is an attack that uses thousands of hosts to target a single network to cause a denial-of-service condition. Tools now exist that can exploit hundreds,

maybe even thousands, of systems at the click of a button and return information to the attacker about exactly which attacks succeeded. These attacks are very disturbing, but they are usually also very noisy. In addition, they usually rely on exploiting unpatched vulnerabilities. When doing this, the risk of crashing systems is pretty high, and that would be very noticeable.

4. *Active-manual*—This is the most worrisome attack. Many people ask how this could be more worrisome than a tool that can exploit thousands of systems at the click of a button. The reason is that if you are subject to one of these, you are up against someone with at least a basic, probably more, knowledge of systems and how to attack them in general, and your network in particular. In this type of attack, the attacker manually attacks a particular network, adjusting the techniques and tools as necessary to counter your defenses. This attacker is probably out to get you, or someone you do business with. They have the time, skill, and resources to do the job thoroughly and to hide their tracks. If the attacker behind one of these attacks is skilled, you may never even know you got attacked!

We frequently discuss the types of attacks that worry us. It is not the first two, and to some extent, not even the third. We know pretty well how to stop worms. (Patch your stuff, and then see the discussion on isolation in Chapter 10.) We also know how to detect mass automated attacks, not to mention how well we know how to stop e-mail worms. The attack that worries us is the one where someone adds himself to your payroll system; the attack where someone gets access to all the patient records at Mass General Hospital; the attack where someone modifies all trades on the New York Stock Exchange by one cent and funnels the proceeds into a Cayman Islands bank account; the attack where someone gets access to the intercontinental ballistic missile systems and obliterates Minneapolis! Those are the types of attacks that worry us. This book is about what we need to do to protect ourselves against those types of attacks.

All the attacks can cause incredible amounts of damage. However, an active-manual attack can cause more targeted damage. An active-automated attack, in the form of a worm, is designed to cause widespread damage; but because it is designed to attack as many systems as possible, it is by necessity generic in nature. The basic principle behind worms is usually to cause the maximum amount of harm to the greatest number of people.

Thus, the damage it can inflict is often more generic. In an active-manual attack, the damage can be much more specific and designed to cause maximum harm to the current victim. There is one notable exception to this: the active-automated attack that is designed to use the maximum number of people to cause the greatest amount of harm possible to one victim. Microsoft, along with others, has been the victim of these types of attacks several times. In them, some criminal wrote a worm designed to infect as many systems as possible and then use them to disrupt access to Microsoft's Web sites. However, these attacks still pale in comparison to what a dedicated active-manual attack can do.

Types of Damage

Generally speaking, four kinds of damage can be inflicted on a network or its data: denial of service (DoS), data destruction, information disclosure, and data modification. You will often see these discussed under the CIA acronym: confidentiality, integrity, and availability. However, data destruction and data modification, although they both fall under integrity, have vastly different consequences, and deserve to be separated. In essence, CIA fails to capture the nuances of what modern criminals do.

The simplest, and most obvious type of damage, is where an attacker slows down, or disrupts completely, the services of your infrastructure or some portion thereof. This is a typical DoS attack. The aforementioned attack on Microsoft's Web presence is an example of this type of attack. In some cases, the damage results from an attack that crashes or destroys a system. In other cases, a DoS attack can consist simply of flooding the network with so much data that it is incapable of servicing legitimate requests. In a flooding attack, it usually comes down to a matter of bandwidth or speed. Whoever has the fattest pipes or fastest computers usually wins. In other cases, particularly in the case of an automated attack, simply moving the computers to a different IP address mitigates the attack.

Of potentially much more serious consequence than a DoS attack is a data-destruction attack. In this type of attack, you are not merely prevented from accessing your resources, they are actually destroyed. Perhaps database files are corrupted, perhaps operating systems are corrupted, or perhaps information is simply deleted. Imagine if someone deleted your accounts receivable database? This type of attack can be extremely damaging, but can be mitigated by maintaining backup copies of both data and equipment.

NOTE: You will commonly see statistics that claim that DoS attacks cause more damage than any other attack type. These statistics are probably true. However, that is because of the sheer number of those attacks, and the fact that many organizations subject to data-destruction attacks will not acknowledge that fact. If you ask yourself truthfully, you will probably choose to have your systems crash any day if the alternative is to have all your data destroyed.

Damage can also result from information disclosure. This damage may be more serious than data destruction, particularly because it is much less obvious. For instance, in February 2004, someone posted portions of Microsoft Windows source code on the Internet.[1] This was an information-disclosure attack that involved portions of intellectual property. In a sophisticated information-disclosure attack, the victim may not know for years whether any data was disclosed. This is often the objective of government spies—to steal information such that they get an advantage while the enemy is unaware of what is happening. One extremely famous example of this happened during World War II. In 1942, the United States had accessed some of the Japanese naval codes, including the code used by Admiral Yamamoto, head of the Japanese combined fleet. The Americans knew that Yamamoto was planning an assault on a location designated as "AF." The problem was that they did not know what the designation AF meant, although they suspected it designated Midway. Commander Rochefort, of the code-breaking command at Pearl Harbor, and Captain Edwin Layton, Admiral Nimitz's fleet intelligence officer, devised a plan to determine whether AF actually did mean Midway. They sent a message via underwater line to Midway asking them to transmit a message in the clear stating that their desalination facility used to produce fresh water was broken. Shortly after the message was sent, the Japanese transmitted a new coded message indicating that AF was short on fresh water and that the conditions for an attack were favorable. Nimitz now had all the information he needed and was able to position the fleet to intercept the Japanese attack at Midway, leading to one of the most spectacular victories of World War II; a definitive turning point in the war in the Pacific.

A covert information-disclosure attack could either leave the victim with a false sense of security, or a nagging feeling of insecurity, both of which can be damaging in the long run. When information is disclosed, an

1. The source code released was portions of a pre-release version of Windows 2000 and portions of Windows NT 4.0 Service Pack 3. It was not leaked from Microsoft but rather from a source-code licensee.

attacker may be able to use it for malicious purposes. For example, confidential trade secrets can be used to undermine market share, to cause embarrassment, or to obtain access to money. Many people think that destruction of data is more damaging than an attacker reading the data, and, of course, whether it is depends on the data and whether regulatory confidentiality requirements are involved. (Some locales, notably California, now have regulatory requirements regarding confidentiality of all data, and virtually all jurisdictions are subject to regulatory confidentiality requirements of at least some data.) However, since we usually have some form of backup, disclosure is typically more severe. If you still have doubts, ask victims of identity theft if they would have rather had the criminal destroy their bank records rather than steal them.

Data modification may cause the most serious damage of all. The reason, as in the case of information disclosure, is that it is very difficult to detect. For example, suppose that the perpetrators broke into your payroll system and added themselves to the payroll? How long would it take you to notice? If you work in a small organization, it probably would be discovered during the next pay period; in a company with thousands of employees, however, it may go undiscovered for years. When writing this book, we were told of a story (no word on the truth of it) about a company that made all employees come pick up their paychecks one week instead of getting them automatically deposited. Apparently, several fake employees were discovered in the process.

When the Microsoft source code mentioned earlier was discovered on the Internet, the immediate concern was whether the perpetrators had also been able to insert back doors into the source code. (This is always the concern when a large software vendor is attacked, even if, as in this case, it was not actually the vendor that was attacked. The news reports immediately stated that "there is no word yet on whether any back doors have been inserted.") Data modification can be used to cause all kinds of damage, some of which may never be discovered, and some of which may only be discovered in very rare events, when the altered data are actually put to use. If someone wants to cause huge amounts of destruction to IT systems, obviously attacking a large software vendor and modifying the source code represents an efficient way to achieve that objective. If we may say so ourselves (after all, we helped design the protection), the Microsoft source code is extraordinarily well protected. However, back doors and Trojans have been discovered in several open source projects to date. Examples

from other realms can easily be constructed. Consider, for instance, what would happen if attackers modified patient blood type data in a medical database, or tax information in an accounting database, or whatever data you consider important in your line of business.

VIRUSES VS. WORMS

There is a long-standing debate about what constitutes a virus and what constitutes a worm. We would much rather not enter that debate because it really is a bit like pornography—we recognize it when we see it (not that we see it very much, mind you!)—and the debate over its nature is largely wasted effort anyway. However, as a very simple definition, a ***virus*** is a piece of malicious code that spreads within a machine, and needs user action to spread to other machines; whereas a ***worm*** is a piece of malicious code that spreads from system to system without extraordinary user action. However, it is basically immaterial to the rest of the book to define better than that.

Most of Us Are Just Roadkill

A friend of ours describes most of the victims of viruses and worms today as "roadkill." They just happen to be standing in front of the truck when it, in the form of the latest worm, comes barreling down the information superhighway. (Yes, this will be the last time we refer to the "information superhighway," and you may complain loudly if we break that promise!) Although it may be true that the person who wrote the worm was not out to attack you specifically, roadkill is still just as dead as if it had been shot with a high-precision weapon. There is an important lesson in that: *Do not become roadkill*. More specifically, there are some very simple things we can do—such as patching—to avoid being roadkill. If we can just avoid being creamed by the latest worm, we can devote our attention to protecting ourselves against the attacks that are actually targeting us.

There Are People Out There Who Are Really Targeting You

Many of the people who are causing damage on our networks today are best compared to the people who spray-paint highway overpasses. They are in it for the sheer joy of destruction and to broadcast their pseudonym.

They may not be out to attack you specifically. As long as they ruin someone's day, that is sufficient. In some cases, they may not actually be after you at all. They may be after the vendor from whom you purchased your software or hardware. By causing damage to you, they discredit the vendor by making it seem as if the vendor's products are more insecure or cause more problems than some other vendor's systems.

The people you really have to worry about are the ones who are directly targeting you. In some cases, they are attacking you actively only because you use some technology that they know how to take advantage of, and taking advantage of it will earn them money, fame, or prestige in a community of like-minded deviants. In other cases, they are after you because you have something they want. You may, for example, have a list of customers. If competitors steal it, they can target your customers. You may have an accounts receivable database. If someone destroys it, you do not know how much money to ask people for, and you will not get paid. You may have a payroll system. If someone destroys it, how long before your employees leave when they do not get paid?

It really does not matter what business you are in. Every organization has something that is of value to someone else. You need to consider what those things are, how much they are worth, and how much money you should spend protecting them. Think of it this way: We all have insurance. Some large companies are self-insured, but they still have to set aside money to pay for claims. Although we can buy insurance for our information technologies, we still have to take reasonable measures to protect them. In Chapter 4, "Developing Security Policies," we discuss how to analyze how much money to spend protecting information and technology assets. Until then, keep in mind that the value of technology is not the technology itself; it is what you do with it. Technology is replaceable, but the services and data you are using it for are not. If your systems are down, the services they would have rendered while they are down are lost forever.

Nobody Will Ever Call You to Tell You How Well the Network Is Working

Between the two of us, we have spent about 20 years administering networks and systems. Throughout those 20 years, one thing became increasingly obvious: *Nobody will ever call you to let you know how well the*

network is working. Never in 20 years of network administration did we get a phone call to tell us that the e-mail system was working, that users could print without glitches, and that files were available without problems. The phone calls system administrators receive seem to always come at 0500 on Saturday morning, with the caller screaming about the network being down. That experience taught us two things:

1. The people who called at 0500 were usually the ones who broke the network in the first place.
2. Information technology is working properly only when users can stop thinking about how or why it works.

Although there is no sustained learning in the first observation (other than that we always seem to work for the wrong people), the second is an example of what we call the "principle of transparency." Users, unlike us, are not interested in technology for technology's sake. In fact, strange as it may seem, they are not interested in technology at all. The users just want the technology to work so they can get their jobs done without having to think of why or how. The ultimate challenge for information technology is to be invisible—completely transparent to the user. Every time users have to think about the technology, it is because something is not working the way it should (or the way they think it should) or because they cannot access resources they want. When a manager has to think about technology, it is usually because she needs to spend more money on it, or because it is no longer working after her eight-year-old downloaded a virus onto the manager's laptop while surfing the Internet as an admin last night. Neither experience is all that pleasant. Fundamentally, the network administrator's job is to make him or herself invisible. This is what makes getting managers to spend money on security so hard. Fundamentally, security management is about spending good money to have nothing happen. Success is measured by the absence of events, not by the presence of them. If nothing happened, you were probably successful protecting the network, or you were just lucky, and you really do not know which!

NOTE: Security management is about spending good money to have nothing happen.

But, Security Will Break Stuff!

So, how does all this relate to network protection? The problem is that whereas network administration is about ensuring that users can get to everything they need, security is about restricting access to things. A colleague of ours used to quip, "Got an access denied? Good, the security is working." At a basic level, that means that security administration at its core is fundamentally opposed to network administration—they have, in fact, conflicting goals. Hence, we have an elemental tradeoff that we need to consider.

As mentioned previously, technology must be transparent to users. Transparency can take many forms. The technology should be easy to use. However, the technology-acceptance research in management information systems has proven that technology also needs to be useful—that it needs to have some kind of compelling functionality—to be accepted by users. (For simplicity's sake, we sometimes group usability and usefulness into the single term *usability*.) Essentially, the tradeoff is between security and usability or usefulness. An old cliché says that the most secure system is one that is disconnected and locked in a safe, then dropped to the bottom of the ocean. Of course, if you have an availability goal, in addition to confidentiality and integrity, this is a suboptimal approach.

This has implications for all software technologies. Take the operating system (OS), for example. The only perfectly secure OS is one that is still in the shrink wrap. After you break the shrink wrap and install the OS, confidentiality and integrity can be compromised. When you install an application on to any operating system, you enable additional functionality that may make the system less secure because it increases the attack surface of the system. The more complexity the system has, the more potential weak points there are. In Chapter 9, "Network Threat Modeling," we discuss the environmental aspects of security hardening and look at how you analyze the usage scenario to optimally harden a system.

We can make any technology more secure, but by doing so we will probably make it less usable. So, how do we make it more secure *and* more usable? This is where the third axis of the tradeoff comes into play. Any good engineer is familiar with the principle of "good, fast, and cheap." You get to pick any two.

Last year, Jesper was visiting a customer to help them design a network architecture for security. During the discussion, it became clear that people were struggling with the tradeoff between security and usability/usefulness. By making the network more secure in some ways, they would

have to make it less usable in other ways. After about 15 minutes of this discussion, he went up to the whiteboard and drew the triangle in Figure 1-2.

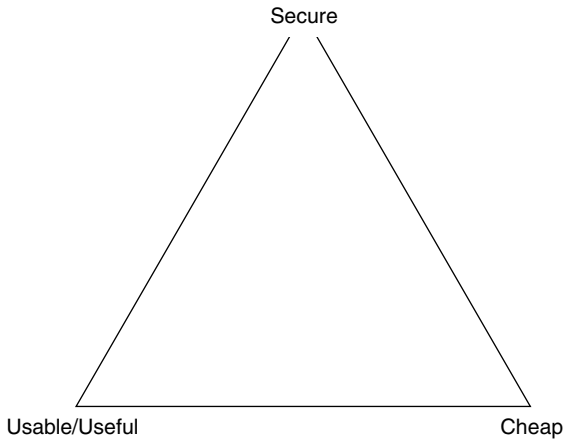

Figure 1-2 The fundamental tradeoffs.

Then he turned to the CIO and told him he gets to pick any two of those. The CIO thought about it for a few seconds and then said, "Ok. I'll pick secure and usable." All of a sudden, everyone knew what they had to work with, and the discussion turned toward what resources they needed to make the system both secure *and* usable.

This fundamental tradeoff between security, usability/usefulness, and cost is extremely important to recognize. Yes, it is possible to have both security and usability/usefulness, but there is a cost, in terms of money, in terms of time, and in terms of personnel. It is possible to make something both cost-efficient and usable, and making something secure and cost-efficient is not very hard. However, making something both secure and usable takes a lot of effort and thinking. *Security is not something you can add on to a fundamentally insecure design; the design itself must incorporate security*. It is not some kind of holy water you can sprinkle on an existing implementation to anoint it to a higher state of security. Security takes planning, and it takes resources. In addition, you will never be able to, or even want to, become completely secure. What you want is to be *secure enough*—to be protected against the threats you care about. That would represent an optimal design in your environment.

A note of interest here is that this book is about designing to a security policy, not to a resource constraint. We all live within resource constraints. However, when you design security strategies, you need to stop thinking about resources. If you start your design by limiting yourself to the options that fit within your resource constraint, you will almost certainly end up with a suboptimal design, because you will dismiss options that may be important before they are fully understood. Furthermore, you will almost certainly end up with a design that cannot ever become optimal. A much better option is to design a strategy that gets you where you want to be. Then you figure out what the resources are that you have to work with. After you have those, you can rank order the components of the design according to benefit and choose which ones to implement now and which to leave for later. Doing the analysis this way also helps you explain to those who control the resources why what they have given you is insufficient.

System Administrator ≠ Security Administrator

Making system or network administrators manage security is counterproductive; those job categories then would have conflicting incentives. As a system or network administrator, your job should be to make systems work, make the technology function without users having to think about it, making the technology transparent. As a security administrator, your job is to put up barriers to prevent people from transparently accessing things they should not. Trying to please both masters at the same time is extremely difficult. Dr. Jekyll/Mr. Hyde may succeed at it (for a time at least), but for the rest of us, it is a huge challenge. The things that will get you a good performance review in one area are exactly what will cost points in the other area. This can be an issue today because many who manage infosec are network or system administrators who are also part-time security administrators. Ideally, a security administrator should be someone who understands system and network administration, but whose job it is to think about security first, and usability/usefulness second. This person would need to work closely with the network/system administrator, and obviously the two roles must be staffed by people who can work together. However, conflict is a necessity in the intersection between security and usability/usefulness. Chances are that only by having two people with different objectives will you be able to find the optimal location on the continuum between security and usability/usefulness for your environment.

How Vendors Can Change the Tradeoff

There are actually several ways to address this tradeoff. Each vendor's technology is used in many different organizations. If we use "effort" as a proxy for the "cheap" axis on the tradeoff, we can see that the amount of effort the vendor expends in making their technology usable as well as secure will offset the amount of effort customers have to expend on the same task. The equation is effectively as follows:

$$\frac{VendorEffort}{CustomerEffort} \times \frac{CustomerEfficiencyCoefficent}{VendorEfficiencyCoefficient} = 1$$

The relationship is not directly one to one because the efficiency and effectiveness of the resources applied to the problem differ. In other words, not everything the vendor does to make the product more secure and usable will actually benefit the customer. However, some portion of the effort that a vendor expends on making the product more secure and usable will benefit customers.

To see an example of this, one needs to look no further than IPsec in Windows 2000 and higher. IPsec is arguably one of the most useful security technologies available in Windows and many other non-Windows operating systems. For example, IPsec was one of the fundamental protection mechanisms used in Microsoft's successful entry in eWeek's OpenHack IV competition in 2002.

OPENHACK

OpenHack is a recurring competition organized by eWeek magazine. One or more systems are configured and connected to the Internet, and the public is invited to try to break into them. Microsoft has participated in three of these and has come unscathed out of all three.

For more information on how the Microsoft entry in OpenHack IV was protected, see `http://msdn.microsoft.com/library/en-us/dnnetsec/html/openhack.asp`.

The IPsec protocol is incredibly versatile. It is also, at least in Windows, the poster child for user unfriendliness. Most people never get over the clunky user interface. If you manage to get over that, you usually

run into one of the truisms about IPsec: It is a lot better at blocking traffic than it is at allowing traffic. There are few analysis tools to help figure out why traffic is not making it through. In Windows Server 2003, the network monitor was enhanced to allow it to parse IPsec traffic, greatly decreasing the troubleshooting effort customers need to invest to understand IPsec. With more effort expended by Microsoft at making IPsec usable, the deployment effort expended by customers would go down greatly, thus decreasing the cost to make networks both secure and usable. What we have is a teeter-totter effect between vendor cost and customer cost (see Figure 1-3).

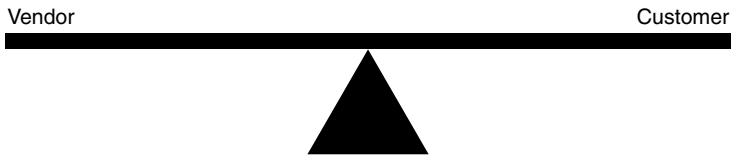

Figure 1-3 Balancing between vendor cost and customer cost.

What this really means is that you very often get what you pay for. A product that costs more should also be more secure and usable/useful than a product that costs less. Other factors come into play here, but these tradeoffs hold in general.

Introduction to the Defense-in-Depth Model

One of the fundamental guiding principles for network protection is defense in depth. As a general rule, it is always a good idea to have multiple prevention mechanisms, at multiple levels, to guard your network and the resources on it. Defense in depth, however, must be done to some threat model. Far too often, we deal with people who want to put some "security" measure in place but do not know what potential problems it mitigates or blocks. In that case, they usually call it a defense-in-depth measure. Unfortunately, one of the results that we need to consider from the fundamental tradeoff above is that any security measure impacts on usability or usefulness or both. At best, this impact is obvious. At worst, it is undefined. Defense in depth is a good thing, but instituting security measures to guard against a nonexistent threat at best has no effect and at worst makes the network less secure or stable and causes grief to users.

Figure 1-4 shows the defense-in-depth model we follow. This model is based loosely on the seven-layer Open Systems Interconnect (OSI) model. The OSI model is old by now, and some people scoff at us when we bring it up. However, the OSI model is still an incredibly useful abstraction for how networking works. It is still one of the most concise and tractable ways to demonstrate functionality in a network, notwithstanding the fact that efforts to build a network that directly implements the abstraction usually fail spectacularly.

NOTE: Many people actually argue that there is an eighth layer on the OSI model: the political layer. (Yes, that is a joke, and we have heard it called other things! Other potential layers include the financial and religious layers.) This is actually accounted for in this defense-in-depth model. It is in the people, policies, and process layer.

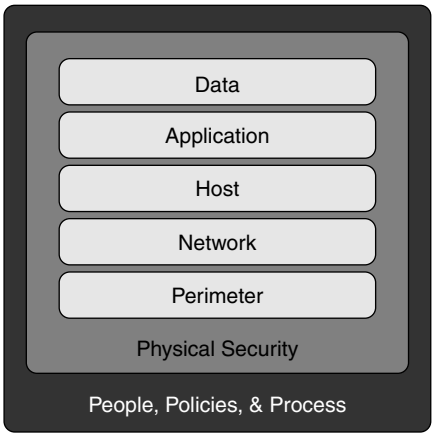

Figure 1-4 Defense-in-depth model.

Defense in depth starts with proper policies, procedures, and, perhaps most importantly, the right people and awareness among the general user population. Without a policy to guide you, you have no hope of developing procedures to implement the policy. Without a policy to guide your users, they will never be able to do the right thing. *Users actually want to do the right thing*. We know this statement comes as a shock to many system administrators, but it really is true. Users are actually interested in being

good citizens and do not want to cause undue grief for their colleagues. Ok, there are some exceptions, but if you tell users how to do the right thing, as long as it makes sense to them, they will usually give it a sporting chance. As for the exceptions, well, if you have a policy, you can always turn them into ex-employees in accordance with the rules for breaking the policy! Part 2 of the book deals with policies, procedures, and how to educate users.

After you have established the policy, you must establish physical security. A network that is not physically secure will never be secure, period. (See Appendix E, "10 Immutable Laws of Security," for more information.) There are things that you can do at a logical level to make up for poor physical security, but in the end, if a bad guy can get to your systems, and he does not care whether you find out that he was there, it is not your system any longer. Chapter 6, "If You Do Not Have Physical Security, You Do Not Have Security," goes into more depth on physical security.

Going beyond physical security, we get into the actual technology that makes up our network. The first step is to ensure that we have a secure perimeter. The perimeter is the interface between your network and the rest of the world. It is where you meet your customers, and your attackers. If we had a dime for every time we had heard someone say "we have a firewall, so perimeter security is taken care of" … well, let's just say we would not have to write this book if that were true. In Chapter 7, "Protecting Your Perimeter," we will explain more about why a firewall does not a perimeter make.

Within the perimeter, there is the network. This is all the gear, wires, and systems that make your organization tick. It is the place where all the attackers want to be. Why? Because most networks are built on the eggshell principle. They have a hard crunchy outside and a soft chewy inside. In Chapters 8 through 10 ("Security Dependencies," "Network Threat Modeling," and "Preventing Rogue Access Inside the Network," respectively), we will look into how to make your network less soft and chewy.

Of course, no network is a network without hosts (the computers that are used on the network). The hosts are the actual systems that store and process information on a network. Hosts are running some form of operating system. In this book, we are primarily concerned with Windows operating systems, by which we mean modern Windows NT-based operating systems (i.e., Windows 2000, Windows XP, and Windows Server 2003 being the current versions). That is not because those are less or more

secure than any other operating system. Windows is not any less secure, or securable, than other operating systems. In the hands of the right people, any platform can be adequately protected, and in the wrong hands, any platform can be successfully compromised. The only differentiators are (a) how much money and effort went into protecting it, and (b) how much functionality you gave up in the process. The reason we discuss primarily Windows is actually much more pragmatic than that. We know our way around Windows. It is what we deal with every day. It is what we have spent a significant part of the past 10 years both securing and hacking. Although most of the principles in the book apply to a network on any platform, some of it is specific; and Chapter 12, "Server and Client Hardening," deals specifically with hardening Windows.

A host without applications is not particularly useful. Applications are what make a system valuable to us. We can have the coolest computer in the world, but without applications, it is only interesting to someone who is willing to write the applications. Unfortunately, at least with common off-the-shelf systems, the more applications you install on a computer, the less secure that computer usually becomes, and by extension, the less secure the network that computer is a part of becomes. Therefore, application hardening is critical. In Part 6 we look at hardening several different kinds of applications, all running on Windows. We also look at how to evaluate the security of additional applications you are using, or considering using, in Chapter 16, "Evaluating Application Security."

Finally, we have the most important part of it all, the raison-d'être of your network: data. Without data, you would not need a network to transmit it. What should you to do protect your data? Turn to Part 7 to find out.

The Defender's Dilemma

When we perform penetration tests, we try to enumerate as many weak points of the target network as possible. Do you think a real attacker would do that? Most likely the answer is no. The attacker only needs to find one way into the network. So, why do we look for more than one when we do a penetration test? The answer is that whereas the attacker needs only one entry point, the defender must defend all points. This is known as the defender's dilemma. It is not enough to close down some holes. You have to close down all the holes an attacker would use.

DEFINITION: A penetration test is where someone tries to break into your network to evaluate how well protected it is and how well the defensive mechanisms work.

Remember the Unicorns

Remember what we said earlier about network security? The defender's dilemma is that network security is not an end state. We have seen a number of companies contract for a penetration test, sometimes as part of a security assessment, only to receive a report that concludes that their network is secure. The accurate conclusion to such a report many times is that the firm that performed the assessment was incapable of getting in, likely because they are not very competent attackers. It is impossible to say that a network is secure. To understand why, remember your unicorns.

If you ever took a symbolic logic class in college, this will bring back fond memories, to be sure. If you did not, you missed out on what may be the most useful course ever offered. Symbolic logic is one of those great philosophy courses in which you learn how to analyze truth. It is a great course for Monday morning, because you probably would do better after a rough weekend!

In symbolic logic, we learn that you can never prove that there are no unicorns. To do so, we would have to go to every possible place where there might be a unicorn and prove that there is not one there. Oh, but we need to go to all these places at the same time; otherwise, the unicorns might just move from one place to the next while we are moving. To extend that to network security, *the only way to demonstrate that a network is secure is to enumerate all the places where it might be insecure, and demonstrate that it is not insecure in any of them*. If you can figure out how to do that, you should write a book.

By contrast, to prove that there are unicorns, all you have to do is find one. That is a lot easier than demonstrating that there are none. In network security, to prove that a network is insecure, all you have to do is find one vulnerability. This is why we say that "network security" as a state is impossible. We can never prove that, so what we will work toward is network protection, the absence of any unmitigated vulnerabilities. The remainder of this book focuses on network protection.

NOTE: A protected network is one with an ***absence of unmitigated vulnerabilities that can be used to compromise the network***.

Summary

Networks are the lifeblood of business today; they are the nervous systems of organizations across the world. They are also the world's most interesting targets to a class of criminals who would love nothing more than to deny you the services of the technology you paid for. This book is about protecting your network. Notice that we did not say "securing your network." You can never hope to secure your network in the sense that it is impervious to attack. That is, you can never secure your network to that extent if you are not willing to turn off the network. The best you can hope for is some measure of protection. *Someone really is out to get you*. They may not even know that you are the one they are causing damage to. They may just want to harm anyone who happens to get in the way. Of course, then there are the people who are out there to get *you*. A healthy level of paranoia turns out to be a useful asset for security administrators.

As security administrators, we face some interesting tradeoffs. Fundamentally, the choice to be made is one between a system that is secure and usable, one that is secure and cheap, or one that is cheap and usable. We cannot have everything. This also means that, in general, it is inappropriate to make the same person responsible for both security and system administration. The goals of those two tasks are far too often in conflict to make this a job that someone can become successful at. Finally, it is critical to evaluate vendor offerings based on the amount of effort expended to make the product secure and usable/useful. The amount of effort the vendor has expended toward that goal will directly offset the amount of effort you will need to expend to implement the product. Whether the effort is expended by the vendor or by you, the customer, it should be carefully considered within an appropriate framework that ensures maximum protection for your network. The remainder of the book is structured around a defense-in-depth model that distinguishes between the various places where you can put protection mechanisms in place.

What You Should Do Today

In our presentations, we often tell the audience that "you should stop whatever you are doing and go fix this right now." The equivalent in this book is the "What You Should Do Today" section at the end of each chapter; these sections provide prescriptive guidance. These are things you should do as soon as you get a chance to.