1 Introduction

Why We Wrote This Book

There are a lot of security books—books about risk management, cryptography, cracking, hacking, and writing better software, along with many firewall "cookbooks" and "bibles." We wrote this book because there are not many books out there that focus specifically on what to do when something goes wrong with your firewall or that explain how to fix errant firewalls. We wanted to put together a book that would combine the practical elements of fixing specific and common problems with Linux firewalls, along with how to figure out what might be causing a problem we might not have foreseen when we set out to write this book. In our exploration, we found that there was no book that did these things. Nothing existed that reduced all these disparate pieces of knowledge, the Tao of firewall security, the Zen of troubleshooting, and the nitty-gritty, step-by-step instructions to fix a problem. We hope you will agree that this book presents a simple and easy-to-follow methodology for solving problems, along with a practical manual that will give you the tools and the knowledge to fix some of the most common problems users experience when building and maintaining Linux-based firewalls.

When reading this book, realize that our intent is to first provide a methodology that can serve as the baseline for solving problems. We believe that having a good mindset is the most critical tool you can have at your disposal when addressing firewall issues. We cannot cover every possible problem, but we have combed our resources to provide as many common problems encountered with Linux netfilter-based firewalls and the solutions to those problems. We arrived at this list of common problems by researching all of the public Linux firewall mailing lists, by speaking with several large Linux customers,

and by reflecting on our own experiences with Linux firewalls spanning over a decade of experience working with Linux. Hopefully we will have covered any problems you might have, but if not, we present our methodology for solving problems in Chapter 5, "The OSI Model: Start from the Beginning."

We have several goals in writing this book, but our chief intent is to make sure you can solve your firewall problems quickly and safely. Plans, strategies, and methodologies, while useful, are no replacement for cold, hard execution. With that intent in mind, this book is really two books. The first book teaches methods, concepts, and abstract ideas to help you learn how to diagnose a problem, to collect information about it, to arrive at a root cause, as well as what tools you can apply to that problem. The second book is a "grab it off the shelf," skim the Table of Contents, find the system/problem, flip to the appropriate page, follow the instructions, and fix the firewall type of book. We want to make sure you can pick this book up and flip to the troubleshooting chapters without having to read the entire book. We're all busy people, and we want to help you—not create more work for you.

In short, this book seeks to make engineers into mechanics with a whole tool box full of tools, a shelf full of easy to read manuals, and a mind filled with the necessary knowledge and scientific thinking to fix any problem a firewall might have. If you don't want to take the time to fully understand the mechanics of a problem, it's possible we have the solution already documented in this book. If you want to understand more, you can read the first half of the book.

HOW THIS BOOK IS ORGANIZED

This book is organized into three sections. Section 1 is a brief introduction to our principles of security and risk management in which we explain how firewalls work, how they should be set up, and some sample recipes for various firewall configurations. If you're new to firewalls or need a refresher, this is a good section for you to read. If you're an old hand with firewalls, you can probably skip this section and move on to Sections 2 and 3.

Section 2 is about troubleshooting and diagnostic methodologies. The intent here is to pass on troubleshooting methods and tools to reduce the amount of effort involved with troubleshooting and implementing a solution. The goal for this section is to teach you how to figure things out for yourself, to do it quickly, and to be able to repeat that process in the future. In Section 2 we explain how the key element to solving problems is to methodically reduce variables and to start with the simplest explanation first.

Section 3 contains the specific troubleshooting chapters in the book. This is where the troubleshooting guides reside. It should be possible to just flip open the book to any part of Section 3 and follow the instructions to diagnose and fix the problem. The goal of

the section is to be a fix-it manual for even the least technically adept user. We believe this gradual procession to the final section of our book provides enough background information to make the process of troubleshooting second nature to the reader.

With regard to the issues of making this material as approachable as possible, we make no assumption about the reader's knowledge about good firewalling, risk management, and computer security practices. An important thought hopefully not lost on the reader is that firewalls and other security devices should be managed with a great deal of forethought and knowledge. Failure to understand a protocol or the consequences of allowing it through your firewall could have disastrous consequences.

However, we do understand that time is short, and sometimes you have to fix the problem and come back to it and understand what effect it has later. Nevertheless, with that said, along with our deepest empathy for all the overworked systems engineers out there, it is very easy to make changes to security models, firewalls, and other security technologies that can have profound and dangerous implications on the security posture of your network if you do not understand what those changes do. This book is not meant to be a replacement for competent technical security advice. There is much to be said for understanding how the products you support work, and firewalls are all the more important to fully grasp. If you're having trouble understanding the guts of your firewall, you could be in for trouble. When in doubt, you can never know too much, so avail yourself of all the information you can get your hands on about information security principles, risk management, and specifically firewall fundamentals. Given the propensity of organizations to rely solely on their firewalls for the lion's share of their security needs, it's critical that the firewall be configured in the most secure manner possible—it could be all that stands between your network's continued normalcy and high-pressure down time.

GOALS OF THIS BOOK

It is our sincere hope that that we accomplish three goals with this book:

1. To teach you, the reader, that security is not really the goal of computer security. As strange as that might sound, the only truly realizable goal of computer security is to manage risk. It's essentially impossible to avoid all the risks out there, so you need to learn to manage the consequences of those risks, while applying reasonable and effective countermeasures to help mitigate those risks. Sometimes, all you will ever be able to do is recover from a risk. The point is to change your mindset and to look at the problem through a different lens: risk management.

2. To teach you how to approach and solve problems in a scientific and methodical manner, using a well-known and widely used problem-solving methodology.
3. Finally and simply, to provide you with a book that contains specific information about how to solve your Linux firewall problem.

We can't cover every possible problem in this book, so as we've already alluded, we will have to show you how to troubleshoot and solve unforeseen problems on your own. To help with this process and to provide access to your peers, we have included numerous references to other websites, mailing lists, and forums where you might be able to seek help from the Linux community, and we've also set up our own website (`www.gotroot.com`) to further assist with the process of documenting new problems and to provide a forum for the community to discuss them and share information.

THE METHODICAL APPROACH AND THE NEED FOR A METHODOLOGY

Oh no you say—not more management speak! Please, I get enough of that already! Fear not; we promise that we won't waste your time with YAUM (Yet Another Useless Methodology). We want you to find your problem and fix it quickly. So you can call this a process, a method, a way, or if you like, call it a methodology—whatever works for you. What we don't want to do is fill your head with some useless babble. This methodology is hard won from years of solving problems.

It all started many years ago. Painted on the wall of our parents' garage was a slogan: "1st law of wrenchin': always check the spark plugs!" Scott painted this on the wall as a reminder of a valuable lesson we learned as teenagers. You need a plan for fixing things, and you need some "laws" to make sure you don't waste a lot of time figuring out what your problem is when it's staring you in the face.

As it was, we had cars, as many young men do in high school, which were the primary focus of our budding engineer minds. They were complex, they were fun, they took us places to have fun, and they were essential parts of teenagers' social lives. Without a car, life was a little less fun, and things were much farther away. However, our cars were used and needed work—sometimes a lot of work—to keep running. So we learned to take care of our cars, to fix them when they broke down, and to modify them to make them better. After all, what kind of engineer is happy with something as it is out of the box? Everything could be a little better, couldn't it? All of this meant that we had to figure out what was wrong with them while not wasting too much time doing so. We paid for those muscle cars, and we wanted to drive them! Enter the first law of wrenchin'.

The first law of wrenchin' was the result of a marathon session of trying to figure out what was wrong with one of our cars (Mike had a 1972 Pontiac GTO; Scott still has a 1970 Oldsmobile 442). For days we struggled to figure out why the motor wasn't performing as it should. We changed the timing, retuned the carburetor, drained the gas tank, uncorked the headers, and so on. We did everything we could think of…except check the spark plugs. That was too easy! It couldn't be the spark plugs, we told ourselves. Five minutes after we stopped convincing ourselves it wasn't the spark plugs, we pulled them out. They were covered with soot. The result of all our other twiddling had made already fouled plugs an absolute mess. In short, the problem was absolute—there was no way we could have gotten that GTO to run correctly, and the only solution was possibly the simplest and easiest to execute. After we had that figured out and had replaced the spark plugs, a five-minute job, we had that 1972 GTO running perfectly. What was really galling about this experience was that we knew better! Of course it could be the spark plugs; we just didn't check them, even though all the signs that this was the root cause were staring us in the face, as it were. The idea that the spark plugs were the cause of the problem just seemed too simple. We thought, wrongly, that it had to be something more complex.

Back then, our problem was caused by the lack of a clear system for problem solving or a mechanism for working through the problem to rule things out before leaping to what we might have thought was the root cause. We needed to work our way up the motor, ruling things out before moving on to the next component, and we had to learn to start with the simple things first. From that moment on, we decided that we wanted to spend more time using our cars as opposed to fixing them. So the journey started toward putting a plan in place to help prevent those forays into endlessly searching for the root cause of a problem. Over the years, we learned to work through the problem carefully, and we discovered that other people had come up with well thought out means of troubleshooting as well. It is that hard won experience and knowledge that we present here to help you to troubleshoot more effectively.

With Linux firewalls, this need for specific steps and a clearly defined approach is no different. There are so many variables in a modern firewall that it's easy to overlook the real source of any problem. For instance, we once had a big high-end SMP DEC Alpha Linux box that we used as a high-end packet switch and firewall. It worked perfectly in our lab in Virginia. We shipped it to our ISP, and all the network cards simply stopped working. We were puzzled, so we got on the box from a console and found that the network cards were up, that they had IPs, but that they simply could not talk to anything.

The NIC driver module was loaded, and the machine had worked without any problem for weeks on our lab's network. The co-location facility's network should have been no different, except for one small change: The ISP's network was Ethernet, and our lab

was FastEthernet. We didn't expect that this would create any problems. After all, the network cards were designed to support both types of network, Ethernet and FastEthernet, and we had worked with these drivers before on both mediums. Our network was faster, and all we did was put the box on a slower network. Stepping down, as it were, shouldn't have made any difference. We were flummoxed. Why would that seemingly insignificant difference cause the cards to stop working?

The simple reason was that we were running the wrong driver. This might seem obvious now, but the cards we were using were DEC cards, and the wrong drivers would load and work with the wrong card, and they would work on a FastEthernet network! In our case, the system thought the card was Tulip FastEthernet card, when the cards were really DE500s. To make matters worse, they even look the same.

So back in our lab, the Tulip driver worked fine at FastEthernet speeds and never once complained about the fact that the card was not a Tulip. Couple this with the fact that we only stocked DEC NIC cards, Tulips, and DE500s, and you can see how this mistake was made. They look the same, and even the OS thought they were the same when it detected them. Fortunately, we had a simple process that we will illustrate here to work through this problem quickly, and we had the system up in short order. Total time to debug this problem and fix it was under five minutes.

The point here is that the best and quickest way to arrive at the root cause of your firewall problem is through the application of a well thought out process of elimination. It's very tempting sometimes to leap to a conclusion about what's causing the problem with your firewall, but unless you're lucky, you can actually make the problem worse. Take our word for it…relax, sit down, and work through the problem step by step. Stick with the approach we have outlined in this book, and you'll soon be tackling the thorniest firewall problems like a guru!

FIREWALLS, SECURITY, AND RISK MANAGEMENT

We believe it's important to discuss the topics of security and risk management, as they relate to firewalls, before we can jump into the process of troubleshooting. After all, your firewall problem might be something fundamental, as opposed to just a misconfiguration that's affecting your security in a very negative way. We strongly believe that you cannot manage a firewall without first thoroughly understanding risk management and computer security.

To begin, it's important to define what security is. Security is defined in some places as "freedom from risk." Some people would perceive this to mean "absence of risk," but we prefer to see it as freedom from its negative effects. The risk might still exist, but you are

free to act (or not act) in spite of that risk. In some cases, you can even eliminate specific risks; in others you cannot but must still act with that risk looming, if you will, over your head. The bottom line is that you properly manage the risks around you so that you can reach a state of being "secure enough." That's what security really is. You can't make all the risks go away—that's impossible—but it doesn't stop people from trying to do it or simply telling themselves those risks don't exist. This is where the problem begins for too many organizations, but given that you are reading this book, you're already on your way to doing better. Learning to accept that risk is a part of life. You have to learn to manage the risks in your life to be "secure enough."

With that said, we are principally concerned with what is referred to as *effective security*—that is, the effort undertaken to manage risk in a calculated, economically feasible and tolerable manner to achieve some goal or set of goals. In a more practical sense, it's useful to think of security as those actions you take to protect yourself from risks that are preventing you from taking certain other actions. To be more succinct, security is everything you do to make it possible to do the things you do.

For example, if you wanted to go for a ride in a car but were concerned with safety, you might wear your seatbelt. You might also want to know how to drive that car and that the weather conditions outside were good enough to make your trip as safe as you wanted it to be. Or you might be a risk taker and be perfectly happy jumping out of an airplane with nothing but altitude and a parachute between you and certain death. In both cases, you might feel totally "secure." For each person, the decisions you arrive at that tell you have achieved that state of "security" are different. They are each dependent on how much risk you are willing to accept and how much you can afford at that particular moment given the information at your disposal.

The bottom line with security is this: Security is not a set of products, technologies, patches, or anything else technical, written, or dictated. It's a process. Security doesn't exist in nature—only the process of becoming "secure enough." Security is about managing your risks and taking actions that allow you to move ahead without getting eaten. When designing your firewall, keep this in mind. You can't stop everything, but you can keep the risks within your range of acceptance.

HOW TO THINK ABOUT RISK MANAGEMENT

There are some core concepts that we should cover first. The first concept to consider is threats. A *threat* is defined as an attacker or any person, trusted or mistrusted, who wants to break into, steal, cause damage, manipulate, or deny access to your information assets.

In Information Warfare parlance, there are three general classes of threats. They are unstructured, structured, and highly structured threats. Briefly, these are defined as

- **Unstructured:** A non-funded adversary with no specific organization or long term planning. *Example: disgruntled employee or a lone "cracker."*
- **Structured:** A funded attacker or group of attackers with some organization and funding with generally longer term planning. *Examples: organized crime or well-funded "cracker" organizations.*
- **Highly Structured:** A highly funded attacker or group of attackers, usually with national state assets. *Examples: national intelligence agencies or terrorist organizations.*

These are general classes of threats, and there are certainly overlaps between them, so don't think of them as hard and fast rules. These are generalizations of all the "bad guys" out there to help you think about who might want to cause harm to your assets, to "steal" them so they can use them for their own purposes or even just to deny you access to your assets. It's folly to try and divine the motivations of a threat. You'll never really know what your attacker is thinking, so don't get caught up in trying to convince yourself that some attacker won't be motivated to do something. People do things for all sorts of irrational reasons, and computer crackers are no exception.

The point here is that understanding the threat is only part of the process of managing risk. You should consider the threats against your assets, but you will also need to consider the second concept at the same time: value. It is not only important to define how valuable your assets are *to you*, but also to understand who might be willing to spend some of their own assets to steal or "borrow" yours and what those assets might be worth *to them*. The secretary's computer might seem unimportant in the grand scheme of things, but maybe your attacker just wants to use her drive space to store something on, or perhaps the attacker needs the secretary's computer to break into the network at large and then break into something far more important. Keep this in mind when analyzing threats and value in your organization.

In economic terms, it's also useful to remember that assets are not only monetary. An asset's value could simply be the time needed to break into your system, or perhaps something more abstract, the opportunity cost of breaking into your system versus some other organization's systems. The bottom line is that value goes beyond just putting a price on an asset; an attacker might not have money in mind at all when breaking into a network.

Consider this example: An unstructured threat, a bored student, decides to try and break into your network. The student might have no funding at all to attack your assets but is simply curious and motivated to see if he can break into a highly secure facility.

Time, for that threat, might be also be a low value asset because the student is on summer break and has free time in extreme abundance. Not all threats are so pedestrian; there are also highly funded and highly motivated organizations that have all the resources necessary to penetrate even the most well guarded assets. The point here is not to underestimate the resources at the disposal of a particular class of threats you might be attempting to defend against. Even though you will never truly be able to read the mind of someone who wants to beak into your computers, it can help to try to think like your opponent. Try to imagine what your enemy might do and, if possible, attempt to break into your own networks from the perspective of those various threats you are attempting to defend against. Just try not to underestimate your threats. A lack of imagination has been the undoing of more than one organization.

With those two critical concepts out of the way, the next core concept to cover is protection or to simply answer the question: What is it exactly that you want to protect? And within that question lies another: What is it about that asset you want to protect? One useful way to answer that question is by considering the following acronym: CIA. It stands for confidentiality, integrity, and availability. This defines what it is about the asset that is important.

For instance, if the asset has to be kept secret, then confidentiality is the goal. Is it critical that the asset not be altered or tampered with? If so, then integrity is a protection goal. And finally, must the asset be available, to whom, and under what circumstances? None of these goals are mutually exclusive; they are simply another means of determining what is important in a security model. You can have an asset that needs all three of these protection goals accomplished—and another with only one.

It's critical that you consider what it is, out of the CIA acronym, that you want to protect within each asset. If your goal is to make sure your website stays up and all the information on your site is public, then IA is important for that system. For e-mail perhaps you are only concerned with confidentiality, or perhaps your company is a health care provider and has a legal requirement to maintain the confidentiality of its customers' information. For each asset, the goals are different, and the reasons for those goals are going to be different.

The next concept to consider is means. How will you accomplish your risk management goals? With any risk management problem, you have three options, which also have their own acronym—AMR: avoid, mitigate, recover.

Specifically with avoidance you can try to prevent or avoid the worst case, with mitigation you attempt to mitigate or manage your risk, and with recovery, you simply plan how you will recover from that risk. As with CIA, none of these are inherently mutually exclusive. You must consider which of these options is acceptable to you before proceeding with your risk management plan.

Above all else, it's important to keep these elements in mind when designing a security model. One example of a "simple" attack on an electronic infrastructure would be to use explosives to destroy the systems. This concept, while far-fetched for our unstructured threat, is not outside the realm of possibility for the structured threat and is part and parcel for the highly structured threat. If you consider how inexpensive explosives are and that they could be placed inside a server, and then that server could be placed inside the co-location facility your systems are located in, the capacity for causing tremendous damage becomes shockingly clear. It's a far simpler solution than a complicated electronic attack when the goal is simply to cause destruction. That's why it's critical to really consider all of these elements when thinking about the risks your systems might face. Your system could simply be in the wrong place at the wrong time. It's a complicated world, and it takes some careful thought to put together a good plan.

Just remember, risk management is a process, not a silver bullet. You cannot "engineer" your way into being secure. Security can only be accomplished by practicing risk management, and that's what this material is all about. With all of this said, this book is not intended to be a comprehensive risk management tome, but merely a guide to the subject to help you think about how to build secure firewalls as part of the process of troubleshooting them. We recommend that if you find these risk management concepts to be new, you read up on risk management before embarking on a new or improved security plan.

COMPUTER SECURITY PRINCIPLES

Computer security, a subset of risk management, is about facilitating activity while minimizing unnecessary exposures that activity creates—in a way that makes it possible for users to still get work done. The bad news is that it's a pretty darn subjective process, and what makes one person feel secure might make another cringe. You can't please everyone. But you can construct a very solid security model by sticking with this simple and powerful premise: Unless we allow the system or network to do something, we will always deny it. The default state of all systems is to deny. Again, the rule is: unless allowed, deny. This is the golden rule for all well-designed firewalls.

This runs counter to the way some people choose to go about creating computing systems and networks. They build a system to do something, and if security is a criteria, they come back to the system after it's built with a set of rules defining what they do not want someone to do. That is the wrong way to go about it, and it also can be a painfully tedious process, even if you do it before you start to build a system. If you want to build a truly secure system, do what you do best—design your system and define what you want your users to do. From there, you construct a security model that opens those

capabilities in your firewall. What you don't want to do is add a bunch of rules to your firewall, defining (exclusively) behavior that your users cannot do. You're bound to miss something that the highly imaginative human mind will come up with. The only thing you want to concern yourself with is what the users need to be able to do. Because your firewall is configured by default to deny everything, you don't run the risk of missing something. You're only allowing things through your firewall that you know about.

Let's explore this further. Think of a firewall as a totally closed system, in essence a "cut" in the wire between two networks through which no traffic at all flows. There are no holes in real firewalls, that is, the physical kind that protect buildings, and we want to keep it that way unless we can prove that there is a need for a hole or service to be opened and that we know we can do it in a way that doesn't introduce fire into the rest of the building. There is a reason firewalls, the networking kind, got their name. A long time ago, that's how most of them were configured—with no holes at all. They were just application proxies that moved data from one domain to the next, and in classified environments there were even firewalls that could only move data one way through a firewall where literally no traffic could flow back in the opposite direction. The point is, firewalls need to be built in as paranoid a manner as you can because with today's demanding networks and customer needs, we have to open a lot of holes in firewalls, and this is where we need to be especially careful. A firewall is not a silver bullet. Allowing traffic through a firewall, sometimes referred to as opening a hole in the firewall, is the same as putting that system out on the Internet with no firewall in front of it!

Remember, the firewall isn't going to magically strip away all the attacks that someone might launch through your firewall. Use the firewall as one of many mechanisms to protect the system or systems behind it. You need to look at things like Intrusion Detection, Protocol analyzers, Intrusion Prevention Systems, classic system hardening, the venerable method of keeping your system patched, cryptography, authentication, software security, and above all else real risk management before you can even begin to say that your system is "secure." Firewalls are only one slice of the security spectrum. You need all the elements of the visible light spectrum to get white light, so it is with security; you need the whole spectrum of methods, procedures, technologies, and management practices to make something "secure."

With that said, here's the bad news…there is no such thing as a "secure" system, which is why we put this word in quotes. The term, secure, is subjective and means different things to different people at different times. For example, your system might be patched up today, so you tell yourself it's "secure," but when you head off for a much earned vacation this weekend, a new vulnerability is published, and you can't get to your system to patch it. And along comes an attacker that uses that vulnerability to break into your, now, insecure systems.

It's also important to keep in mind that with all these measures taken, it is still possible that your system might be broken into or that some other calamity might destroy or make the system unavailable. A flood for instance, an earthquake, or a just plain old hardware failure could be far more damaging than a hypothetical attack. That's why it's important to recognize that there is more to risk management than just securing a system from attacks.

That's the real business that every security professional is really in—managing risk. With risk, as we stated before, you can do three things to manage it: you can avoid it, mitigate it, or recover from it. Sometimes you can do several of theses things; other times you can do only one; and sometimes you can do none of the above. It's important to understand this. This point is critical, and it bears repeating. All security is really about is managing risk. Sometimes you can do something because the risk is acceptable, and sometimes you cannot. This point causes much grief for many security professionals, their customers, users, and bosses when security is confused with some mythical state of being absolutely secure. All you can ever do is manage your risk in a manner that meets your criteria for acceptable risk and loss. You're happy jumping out of the airplane with a parachute; your buddy is not.

Also, to review, threats are a critical competent of risk analysis. To determine if your efforts are necessary or if they are enough, you must consider what threats are going to try to compromise the system(s) you are protecting. For instance, if you are running a simple website with nothing of interest or value to terrorists or organized crime, you might not need to consider those threats. If you're the CIA, you need to consider those threats.

One other thing to keep in mind with threat analysis is that the system you are running might not be the attacker's real target. Your system might be broken into by a highly structured threat to attack the real target or perhaps another "zombie" along the way, which in turn might be used to break into the real target, or your system could just be broken into by a worm, virus, or bored cracker.

The bottom line is that there are some very odd people out there who want to break into systems for all sorts of reasons. It's critical to understand that there are many reasons your system might be broken into, some of which might never have dawned on you. Many a poor system owner has found their "unimportant" system broken into by persons that could care less about what that system had been used for. The attackers now "own it," and they're using it for their own ends. Sometimes, all an attacker really wants is the virtual real estate your systems occupy.

FIREWALL RECOMMENDATIONS AND DEFINITIONS

A firewall is a device that provides secure separation of one or more networks from each other. In the ideal case, a firewall should provide physical separation between all networks. Virtual networks have become very popular, and there is a tendency to create "separation" between networks by using virtual abstractions. This is not a recommended configuration. One error in the configuration, a flaw in the software or hardware, or some other problem can cause your security to fail dramatically and completely with purely virtual network separation.

The best rule of thumb, especially given the availability of free firewalling technologies such as **iptables** and **ipchains**, is to always separate networks in a manner that you can physically verify. Color-coded wires running to networks on physically separated, not virtually or VLAN separated, switches and hubs is a wise investment. Nothing beats physical separation.

We also recommend that your routing scheme be organized in such a manner that if your firewall is misconfigured or fails for some reason, that traffic would not normally move from one domain of your network to the other. This might sound like common sense, but some commercial firewalls have been known to fail "open" and forward all traffic between networks.

For example, we were asked to look at a customer's network that was experiencing some "strange" behavior. After quickly looking at the company over the Internet, it was determined that their entire corporate network, every server and workstation, was completely exposed to the Internet. We knew that they had a firewall, and it was a reliable commercial grade product that should not have allowed this to occur. However, we also knew that they had been issued an Internet routable network by their ISP and were using that IP space for their internal network as well. When we arrived, we were amazed at what havoc a simple mistake could make. Our customer had installed their firewall on a switch and had plugged BOTH their internal and external interfaces into the same unmanageable switch! From their perspective, everything was working perfectly. They could get out to the Internet, mail worked, and everything was as it should be. You see, it turns out that they never bothered to check to see if anyone could get in! Sadly, this turns out to be the case with many organizations, which is why we recommend that your networks be configured in such a way as to make sure this is unlikely to occur if your firewall fails. One simple way to do this is to never, under any circumstances, use Internet routable IPs behind your firewalls. Thankfully, there is an entire IETF RFC dedicated to this, RFC 1918, and there are network blocks set aside for everyone to use for their internal use.

Briefly, they are:

```
10.0.0.0        -   10.255.255.255  (10/8 prefix)
172.16.0.0      -   172.31.255.255  (172.16/12 prefix)
192.168.0.0     -   192.168.255.255 (192.168/16 prefix)
```

If you use these addresses for your internal networks, they will not easily route over the Internet if you start to experience problems with your firewall.

You should also test your firewall—not to see if you can get out, but rather, start by testing to see if you can keep the bad guys out. We can't stress the importance of this enough. It's very easy to misconfigure a firewall and never know about it until it's too late.

Equally, this example serves to illustrate the value of using a private network for your internal and, where possible, DMZ networks. This is not a fool-proof solution, but it adds another layer of security to your risk management program.

WHY DO I NEED A FIREWALL?

If it's not already clear, there are many reasons, but the two really big ones are that there are bad people in the world who want to get in to your networks and that your network depends on millions of lines of buggy, imperfect software, usually filled with really big security holes that can compromise your assets, which no one seems to know about until it's too late.

If this doesn't sound like a problem for you, then you don't need a firewall. In practical terms, it's the very rare case when a firewall would not be helpful, especially with Linux, where we have access to many powerful and free, as in beer and speech, firewall tools. Given the tremendously powerful firewall tools included with Linux, it would be foolish not to use them. And because these tools are free, we strongly recommend that you not only build a network firewall, but that you also use **iptables** or **ipchains** on all your workstations, laptops, servers, devices, and anything else running Linux.

Thankfully, many Linux vendors have made the process of configuring desktop and server firewalls much easier by including firewalling tools that auto-magically configure **iptables** or **ipchains** on install, or any time you wish to change or install those firewall rules. And if your vendor doesn't include tools to do that, there are dozens of tools that make configuring a Linux firewall a snap. Some of them are available at our website, `www.gotroot.com`.

Just remember, like everything else we've discussed so far, firewalls are not a silver bullet. Just because you installed one doesn't mean you're safe. A firewall is a compartmentalization tool with holes in it. If you let someone through to your web service port, it's not going to protect you from web attacks; if your browser has a flaw in it, it's not going to protect from spyware that leverages that hole to attack your computer. It takes more than a firewall to manage risk these days.

DO I NEED MORE THAN A FIREWALL?

Yes, always, without a doubt, you need more than a firewall. You will need tools to help you harden your system, to check the integrity of your files, binaries and drivers, intrusion detection and prevention tools, patch management technologies, and penetration testing tools. Sometimes you might even need military grade cryptography, security policies to describe what users can and cannot do, training for those users, and above all else, a certification program and even a risk management plan. A firewall is never enough by itself.

WHAT KINDS OF FIREWALLS ARE THERE?

There are four types of firewalls, which are all available on Linux platforms. These are, in order of complexity and features, packet filtering, application proxies, stateful inspection, and hybrid. There are, as we will explain, a few very specialized firewalls for extremely high security environments, trusted guards, and one way filters. Those types of firewalls are beyond the scope of this book, but if you are interested in learning more about them, we have a section on those types of devices at our website (`www.gotroot.com`).

FIREWALL TYPES

- **Packet Filtering:** These are the first generation of firewalls, generally what you see on modern routers these days. While useful, they are generally trivial to circumvent an attacker using a number of common attack methods.
- **Application Proxies:** This is the second generation of firewall technology, although it could be said this is actually the first in some ways. An application layer firewall is a proxy server, like the HTTP proxy server, Squid, for example. They also provide a layer of granularity into security policy that you won't find in stateful inspection or packet filtering firewalls.

Basically, an application proxy is an application that runs on your firewall or gateway that relays traffic between you and your destination. The added advantage here is that the traffic is being sent/received between both endpoints by a third-party application, meaning you can enforce very specific guidelines on the way the traffic is crafted between both points.

- **Stateful Inspection:** This would be the third generation of firewall technology, this is related to the packet filtering method, but it extends the capabilities of firewalling by continuing to inspect the packets as they pass through the firewall. Netfilter/**iptables** is a stateful inspection type firewall.
  Netfilter/**iptables**' main features are
  - stateful packet filtering (connection tracking)
  - all kinds of network address translation
  - flexible and extensible infrastructure
  - large number of additional features as patches
- **Hybrids:** Hybrids are the fourth generation. They are a combination of the previous three, giving the users more control of the methods they intend to employ to carry out their firewall policy.

As stated previously, there are two other types of firewalls that we are not going to cover in this book. Their use is very specialized, and it's not practical to construct them with **iptables** or **ipchains**. The other types fall into two categories, trusted guards and one-way firewalls. Trusted guards basically prevent data, in theory, from moving from one domain to another domain. Basically, they are designed to prevent "Top Secret" data, for instance, from moving into a domain that is only rated "Secret," and they accomplish this with specialized hardware and Multi Level Security (MLS) data labeling techniques. With commodity hardware, such as off-the-shelf PCs, it's not really possible to accomplish true compartmentalized security with trusted guards. You need some specialized hardware to protect the memory on the PC and even in some cases, specialized NIC cards. You can accomplish some less trusted attempts at MLS with commodity hardware if you want to tinker, but with that type of hardware the system will never really be "trusted" in the classical computer science sense of the term. Hence, we will not focus on these security models in this book. Regardless, MLS and its use with what are referred to as Compartmentalized WorkStations (CWS) is a really neat and useful concept. It has its uses, but these technologies are rarely used in the commercial world. And, as we alluded to earlier, the capacity to label the data in a manner that would make trusted guards possible without the use of commercial products has only recently entered the standard

Linux kernels, and the hardware to support it is not in high supply for the general public. As it is right now, there is no easy way to set up a trusted guard with Linux, but this is likely to change as demand increases. We are fascinated by the improvements in the Linux kernel in this area and are keeping a very close eye on developments. As before, if this topic interests you, feel free to visit our website (`www.gotroot.com`) and join in the forums there on this topic.

Aside from trusted guards, the other type of firewall we will not cover in this book is known as a "one way" firewall. This kind of firewall involves the use of specialized hardware that literally will only transmit data in one direction. This too is a highly specialized form of a firewall because it is designed to prevent a very unique form of attack through all modern firewalls called a "covert channel attack." Briefly, a "covert channel attack" is where someone behind a firewall is able to send data, or possibly to even construct a full data channel, through a firewall in spite of the firewall's policy. It's beyond the scope of this book to go into more detail on this type of attack, but it's a fascinating issue to consider as it may be relevant to your organization's risk management plan. Again, you can always check `www.gotroot.com` for information on this topic or to ask on our forums for more information.

Both of these types of firewalls are in response to risks that classic firewalls, all four of the types discussed at the first part of this section, cannot adequately protect against. We think it's good to point this out because your firewall can only protect against a finite range of risks. It alone cannot protect against all of the current known threats that networks face.

THE MYTH OF "TRUSTWORTHY" OR "SECURE" SOFTWARE

The goal of building secure and more reliable software can never be under-appreciated or over-emphasized in our opinions. We want to see vendors produce better software, but we do recognize that wanting something is very different from having something, and too many promise what they can't deliver. As it is, software is going to have flaws because it's very hard to write bug free and infallible software, and software is written by imperfect people. That's not to just pick on software; hardware is equally plagued with these issues. Computer hardware and software are complex systems, ever more complex everyday, and market forces sometimes dictate whether or not security will be the product's primary focus, or even if security will be get any attention at all. After all, as we already said, "secure" is a subjective concept that differs from party to party. What the

builder envisions as "secure" might not be the ultimate case for every user. With that said, if you want to manage the security of your systems, you need to start with the cardinal rule:

> *People are not perfect. People make the things we use. Therefore those things we use will not be perfect.*

To be more succinct, your software and hardware probably have bugs in them. They probably have a lot of bugs. Prepare for that, no matter how great your software or hardware vendor is or what they might say about the security of their products or their track record.

It's best to look at your systems as the flawed, imperfect and insecure things they probably are. If you start with this assumption, the task at hand can seem daunting. But don't fear! You've made it this far in life, and real life is no more certain, perfect, or secure than the computing world is. In nature, there is no such thing as total security. That concept does not exist. There is only risk management. Sometimes you crawl out of your hole to seek food, and something else eats you; other times you stay in your hole and starve; and other times, most of the time, you leave your hole, you find food, purpose, enjoyment, and everything turns out fine. It's balancing those risks, threats, and rewards that defines life.

With computers it's no different. Sometimes you have to take greater risks with your Information Technology systems to reach a new customer, to link with an important vendor, or to simply operate your business. If you have systems attached to the Internet, then you are taking a risk.

KNOW YOUR VULNERABILITIES

Continuing this thread, it's important to understand what your weaknesses and vulnerabilities are before you try to solve any security problem through improvement. After all, if you don't know what's broken, how can you fix it? As we mentioned before, the best way to learn what your adversaries can do and where you need to focus your efforts, is to look at your network and its assets from the perspective of an attacker in a brutally honest manner. You will want to attack your network or engage someone who can do it for you. The intent is to enumerate every known vulnerability in your organization so that you can make an informed assessment to manage the risks created by these vulnerabilities. We will explore this topic in more depth in later chapters. Just remember, it's not enough to secure your network. After you do that, you will want to try and break in. You can't know if your efforts are worthwhile without testing them.

CREATING SECURITY POLICIES

In addition to the technical methods used to secure networks and computers, via electronic rules, it's important to create rules for the people that will be using those computers and networks. Many times, you cannot engineer away the cause of the problem: the user. Having a clearly defined security policy is critical to the success of any risk management plan.

TRAINING

After you have your security model and policies in place, users will need to be trained on them. This can be something as simple as, for a home firewall, explaining to your spouse, roommates, or whomever that you now have a firewall in place and how it is configured.

For larger organizations, you will need to go a little further than this. Training your users can be the difference between a security plan that works and one that fails on the first day. For instance, your plan might be thwarted via something as simple as a social engineering attack. An employee or user is convinced to give someone else access to your systems or building.

DEFENSE IN DEPTH

We can't stress enough the need for Defense in Depth (DID). Computers, software, firewalls, and all the creations of man have one fundamental flaw: People created them, and people are not perfect; people make mistakes. You simply cannot rely on one thing to protect your network and computers. Firewalls are wonderful and powerful tools for protecting systems, but they are neither perfect nor complete solutions for the incredibly wide range of vulnerabilities and threats that modern networks and computers face.

SUMMARY

Even though this is a book specifically about firewalls, we would be remiss if we did not point out that security requires more than just deploying a firewall. A firewall will not protect your systems from many of the electronic threats they may face. Remember, a firewall is just one security tool; it's not a silver bullet! We'll discuss this in more detail in the next chapter, but suffice it to say that firewalls are over emphasized in many security plans. At the very least, plan for what you will do if your firewall fails you miserably. Create a plan to recover and build in additional layers of defense to protect your assets.

As to the purpose of this book and its goals, troubleshooting your firewall will require you to develop an approach to solve problems. In later chapters we describe the methodology we use, which also happens to be used by the U.S. Army, emergency medicine, and other high-pressure fields to quickly arrive at the root cause of a problem, to implement a solution, and to test the effectiveness of the implemented solution. You will find that this approach will not only save you time, but it also will save you the trouble of potentially making the problem worse.