

1

Introduction

1.1 Why Security?

What is “computer security”? Broadly speaking, security is keeping anyone from doing things you do not want them to do to, with, on, or from your computers or any peripheral devices. This definition is, of course, much too broad. Nevertheless, it does lead us to some very important questions that must be answered by anyone who wishes to deploy an effective security mechanism.

The first such question is “What resources are we trying to protect?” The answers are not always obvious. Is it the CPU cycles? At one time, that made a great deal of sense; computer time was very expensive. That is no longer true in most situations, supercomputers being a notable exception. More seriously, a CPU—or rather, a CPU running certain software with certain configuration files—has a name, an identity, that lets it access other, more critical resources. These are often more sensitive than CPU time. A hacker who compromises or impersonates a host will usually have access to all of its resources: files, storage devices, phone lines, etc. From a practical perspective, some hackers are most interested in abusing the identity of the host, not so much to reach its dedicated resources, but to launder further outgoing connections to other, possibly more interesting, targets. Others might actually be interested in the data on your machine, whether it is sensitive company material or government secrets.

The answer to this first question will, in general, dictate the host-specific measures that are needed. Machines with sensitive files may require extra levels of passwords or even (in rare cases) file encryption. Similarly, if the target of interest is the outgoing connectivity available, the administrator may choose to require certain privileges for access to the network. Possibly, all such access should be done through a daemon that will perform extra logging.

Often, of course, one wants to protect all such resources, in which case the obvious answer is to stop the attackers at the front door, i.e., not let them into the computer system in the first place. Such an approach is always a useful start, although it tacitly assumes that one’s security problems originate from the outside.

This leads us to our second major question: “Against whom must the computer systems be defended?” Techniques that suffice against a teenager with a modem are quite useless against

a major intelligence agency. For the former, enhanced password security might do the trick, whereas the latter can and will resort to wiretapping and cryptanalysis, monitoring spurious electronic emissions from your computers and wires, and even “black-bag jobs” aimed at your machine room. Computer security is not a goal, it is a means toward a goal: information security. When necessary and appropriate, other means should be used as well. The strength of one’s computer security defenses should be proportional to the threat from that arena; other defenses, though beyond the scope of this book, are generally needed as well.

Figure 1.1 shows two measures of the growth of the Internet. The top shows a count of hosts detected by automated sweeps of the Internet. The counts for recent years are certainly on the low side of the actual number: there is no reliable technology available to count all the computers connected to a large internet. The lower plot shows the number of networks registered on NSFnet over the past few years. Please note: the vertical scale on both charts is logarithmic. These growths are exponential. If there are two million hosts registered, how many people have access to those computers? How many would like to try their hand at hacking, perhaps even as a career?

The third question one must answer before deploying a security mechanism represents the opposite side of the coin: how much security can you afford? Part of the cost of security is direct financial expenditures, such as the extra routers and computers to build a firewall gateway. Often the administrative costs of setting up and running the gateway are overlooked. But there is a more subtle cost, a cost in convenience and productivity, and even morale. Too much security can hurt as surely as too little can. Finding the proper balance is tricky, but utterly necessary—and it can only be done if you have properly assessed the risk to your organization from either extreme.

One more point is worth mentioning. Even if you do not believe you have valuable assets, it is still worth keeping hackers out of your machines. You may have a relaxed attitude, but that may not be evident to the attackers. There are far too many cases on record of systems being trashed by hackers who thought they had been detected. (Someone even tried it to us; see Chapter 10.)

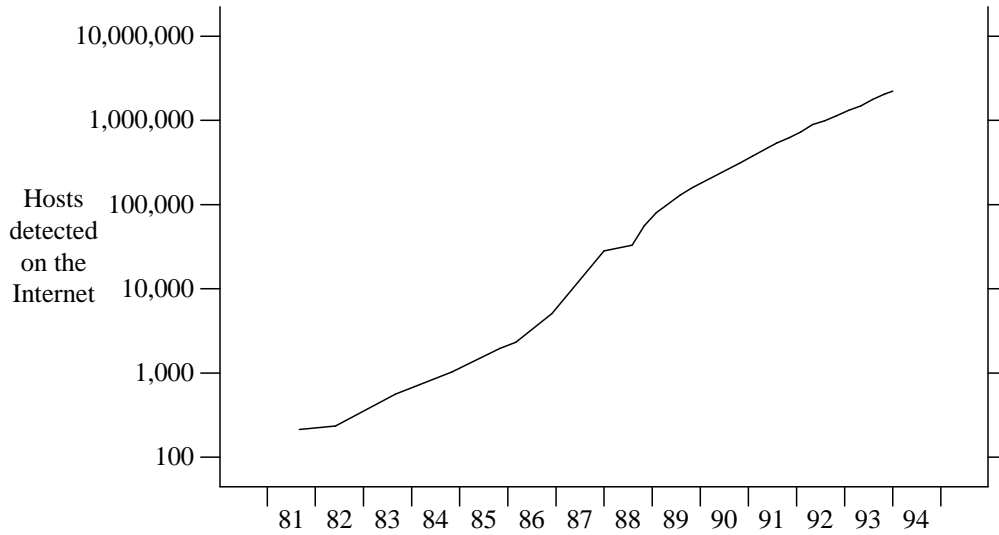
1.2 Picking a Security Policy

Even paranoids have enemies.

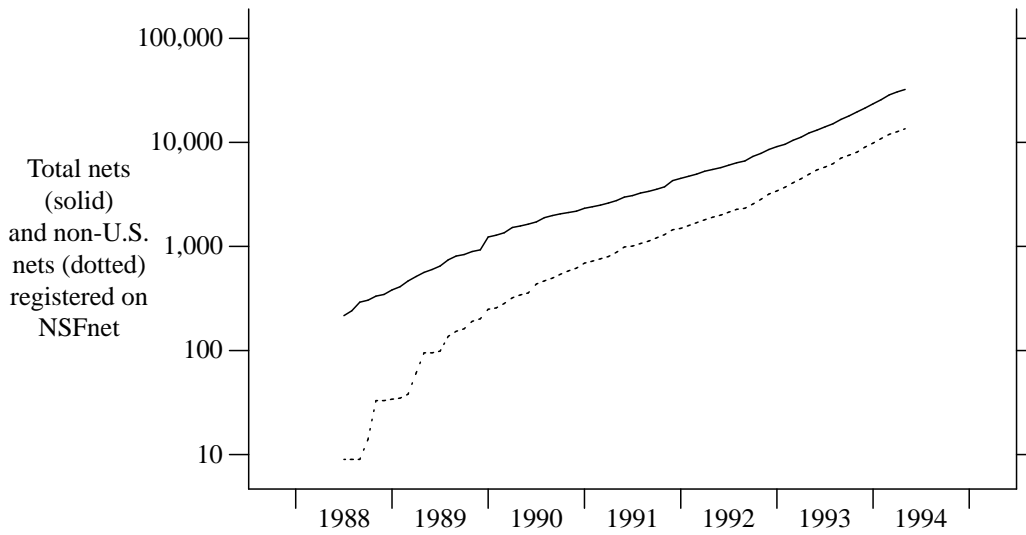
—ANONYMOUS

A *security policy* is the set of decisions that, collectively, determines an organization’s posture toward security. More precisely, a security policy determines the limits of acceptable behavior, and what the response to violations should be. Naturally, security policies will differ from organization to organization. An academic department in a university has different needs than a corporate product development organization, which, in turn, differs from a military site. But every organization should have one, if only to let it take action when unacceptable events occur.

In this book, we are not much concerned with how to respond to incidents; that is covered quite well in other works, such as [Holbrook and Reynolds, 1991]. But defining the limits of acceptable behavior is fundamental to the operation of a firewall.



Source: nic.merit.edu/nsfnet/statistics/history.hosts



Source: nic.merit.edu/nsfnet/statistics/history.netcount

Figure 1.1: Internet growth.

The first step, then, is to decide what is and is not permitted. To some extent, this process is driven by the business or structural needs of the organization; thus, there might be an edict that bars personal use of corporate computers. Some companies wish to restrict outgoing traffic, to guard against employees exporting valuable data. Other aspects may be driven by technological considerations: a specific protocol, though undeniably useful, may not be used, because it cannot be administered securely. Still others are concerned about employees importing software without proper permission: the company doesn't want to be sued for infringing on someone else's rights. Making such decisions is clearly an iterative process, and one's answers should never be carved in stone or etched into silicon.

1.2.1 Stance

The moral of this story is, anything you don't understand is dangerous until you do understand it.

Beowulf Schaefer in Flatlander
—LARRY NIVEN

A key decision in the policy is the *stance* of the firewall design. The stance is the attitude of the designers. It is determined by the cost of the failure of the firewall and the designers' estimate of that likelihood. It is also based on the designers' opinions of their own abilities. At one end of the scale is a philosophy that says, "we'll run it unless you can show me that it's broken." People at the other end say, "show me that it's both safe and necessary; otherwise, we won't run it." Those who are completely off the scale prefer to pull the plug on the network, rather than take any risks at all. Such a move is too extreme, but understandable. Why would a company risk losing its secrets for the benefits of network connection? One can best appreciate just how little confidence the U.S. military has in computer security techniques by realizing that connecting machines containing classified data to unsecured networks is forbidden.

In general, we lean toward the paranoid end of the scale (for our corporate environment, we should stress). We've tried to give our firewalls a fail-safe design: if we have overlooked a security hole or installed a broken program, we believe our firewalls are still safe. Compare this approach to a simple packet filter. If the filtering tables are deleted or installed improperly, or if there are bugs in the router software, the gateway may be penetrated. This nonfail-safe design is an inexpensive and acceptable solution if your stance allows a somewhat looser approach to gateway security.

We do not advocate disconnection for most sites. Our philosophy is simple: there are no absolutes. (And we believe that absolutely. . . .) One cannot have complete safety; to pursue that chimera is to ignore the costs of the pursuit. Networks and internetworks have advantages; to disconnect from a network is to deny oneself those advantages. When all is said and done, disconnection may be the right choice, but it is a decision that can only be made by weighing the risks against the benefits.

We advocate caution, not hysteria. For reasons that are spelled out below, we feel that firewalls are an important tool that can minimize the danger, while providing most—but not necessarily all—of the benefits of a network connection. But a paranoid stance is necessary for many sites when setting one up, and we can prove it.

Axiom 1 (Murphy) *All programs are buggy.*

Theorem 1 (Law of Large Programs) *Large programs are even buggier than their size would indicate.*

Proof: By inspection. ■

Corollary 1.1 *A security-relevant program has security bugs.*

Theorem 2 *If you do not run a program, it does not matter whether or not it is buggy.*

Proof: As in all logical systems, (**false** \Rightarrow **true**) = **true**. ■

Corollary 2.1 *If you do not run a program, it does not matter if it has security holes.*

Theorem 3 *Exposed machines should run as few programs as possible; the ones that are run should be as small as possible.*

Proof: Follows directly from Corollaries 1.1 and 2.1. ■

Corollary 3.1 (Fundamental Theorem of Firewalls) *Most hosts cannot meet our requirements: they run too many programs that are too large. Therefore, the only solution is to isolate them behind a firewall if you wish to run any programs at all.*

Our math, though obviously not meant to be taken seriously, does lead to sound conclusions. Firewalls must be configured as minimally as possible, to minimize risks. And if risks do not exist, why run a firewall? We forbear to label it an axiom, but it is nevertheless true that some paranoids have real enemies.

We can draw other conclusions as well. For one thing, we feel that any program, no matter how innocuous it seems, can harbor security holes. (Who would have guessed that on some machines, integer divide exceptions¹ could lead to system penetrations?) We thus have a firm belief that everything is guilty until proven innocent. Consequently, we configure our firewalls to reject everything, unless we have explicitly made the choice—and accepted the risk—to permit it. Taking the opposite tack, of blocking only known offenders, strikes us as extremely dangerous.

Furthermore, whether or not a security policy is formally spelled out, one always exists. If nothing else is said or implemented, the default policy is “anything goes.” Needless to say, this stance is rarely acceptable in a security-conscious environment. *If you do not make explicit decisions, you have made the default decision to allow almost anything.*

It is not for us to decree what services are or are not acceptable. As stated earlier, such decisions are necessarily context-dependent. But the rules we have given are universal.

¹See CERT Advisory CA-92:15, July 21, 1992. Information on obtaining CERT advisories is given in Appendix A.

1.3 Strategies for a Secure Network

1.3.1 Host Security

To some people, the very notion of a firewall is anathema. In most situations, the network is not the resource at risk; rather, it is the endpoints of the network that are threatened. By analogy, con artists rarely steal phone service *per se*; instead, they use the phone system as a tool to reach their real victims. So it is, in a sense, with network security. Given that the target of the attackers is the hosts on the network, should they not be suitably configured and armored to resist attack?

The answer is that they should be, but probably cannot. Theorem 3 shows that such attempts are probably futile. There *will* be bugs, either in the network programs or in the administration of the system. It is this way with computer security: the attacker only has to win once. It does not matter how thick are your walls, nor how lofty your battlements; if an attacker finds one weakness—say, a postern gate (backdoor), to extend our metaphor—your system *will* be penetrated. Unfortunately, that is not the end of your woes.

By definition, networked machines are not isolated. Typically, other machines will trust them in some fashion. It might be the almost-blind faith of *rlogin*, or it might be the sophisticated cryptographic verification used by the Kerberos authentication system [Bryant, 1988; Kohl and Neuman, 1993; Miller *et al.*, 1987; Steiner *et al.*, 1988], in which case a particular user will be trusted. It doesn't matter—if the intruder can compromise the system, he or she will be able to attack other systems, by taking over either *root*, and hence the system's identity, or some user account.

It might seem that we are unduly pessimistic about the state of computer security. This is half-true: we are pessimistic, but not, we think, unduly so. Nothing in the recent history of either network security or software engineering gives us any reason to believe otherwise. Nor are we alone in feeling this way.

Consider, for example, the famous *Orange Book* [DoD, 1985a]. The lists of features for each security level—auditing, access controls, trusted path, and the like—get all the attention, but the higher levels also have much more stringent assurance requirements. That is, there must be more reason to believe that the system actually functions as designed. Despite those requirements, even the most trusted system, with an **A1** evaluation, is not trusted with the most sensitive information if uncleared users have access to the system [DoD, 1985b]. Few systems on the Internet meet even the **C2** requirements; their security is not adequate.

Another challenge exists that is totally unrelated to the difficulty of creating secure systems: administering them. No matter how well written the code and how clean the design, later human error can negate all of the protections. Consider the following sequence of events:

1. A gateway machine malfunctioned on a holiday weekend, when none of the usual system administrators was available.
2. The backup expert could not diagnose the problem over the phone and needed a guest account created.
3. The operator added the account *guest*, with no password.

4. The expert neglected to add a password.
5. The operator forgot to delete the account.
6. Some university students found the account within a day and told their friends.

Unlikely? Perhaps, but it happened to one of our gateways. The penetration was discovered only when the unwanted guests happened to trigger an alarm while probing our other gateway machine.

Our firewall machines are, relatively speaking, simple to administer. They run minimal configurations, which in and of itself eliminates the need to worry about certain things. Off-the-shelf machines have lots of knobs, buttons, and switches with which to fiddle, and many of the settings are insecure. Worse yet, many are shipped that way by the vendor; given that higher security generally makes a system less convenient to use and administer, some manufacturers choose to position their products for the “easy-to-use” market. Our internal network has many machines that are professionally administered. But it also has many departmental machines that are unpacked, plugged in, and turned on, and thereafter all but ignored. They run old releases of the operating system, with bugs fixed if and only if they directly affect the user population. If the system works, why change it? A reasonable attitude, much of the time, but a risky one, given the intertwined patterns of transitive network trust.

1.3.2 Gateways and Firewalls

*'Tis a gift to be simple,
'Tis a gift to be free,
'Tis a gift to come down where we ought to be,
And when we find ourselves in the place just right,
It will be in the valley of love and delight.
When true simplicity is gained,
to bow and to bend, we will not be ashamed
To turn, turn, will be our delight,
'Til by turning, turning, we come round right.*

—SHAKER HYMN

By this point, it should be no surprise that we recommend using firewalls to protect networks. We define a *firewall* as a collection of components placed between two networks that collectively have the following properties:

- All traffic from inside to outside, and vice-versa, must pass through the firewall.
- Only authorized traffic, as defined by the local security policy, will be allowed to pass.
- The firewall itself is immune to penetration.

Boom!



Not all security holes are merely bad. Some go all the way to truly horrendous. We use a “bomb” symbol to indicate a particularly serious risk. That doesn’t mean you can be sanguine about the others—the intruders don’t care much how they get in—but it does give some rough guidance about priorities.

We should note that these are design goals; a failure in one aspect does not mean that the collection is not a firewall, simply that it is not a very good one.

That firewalls are desirable follows directly from our earlier statements. Many hosts—and more likely, most hosts—*cannot* protect themselves against a determined attack. Firewalls have several distinct advantages.

The biggest single reason that a firewall is likely to be more secure is simply that it is not a general-purpose host. Thus, features that are of doubtful security but add greatly to user convenience—NIS, *rlogin*, etc.—are not necessary. For that matter, many features of unknown security can be omitted if they are irrelevant to the firewall’s functionality.

A second benefit comes from having professional administration of the firewall machines. We do not claim that firewall administrators are necessarily more competent than your average system administrator, but they may be more security conscious. However, they are almost certainly better than nonadministrators who must nevertheless tend to their own machines. This category would include physical scientists, professors, and the like, who (rightly) prefer to worry about their own areas of responsibility. It may or may not be reasonable to demand more security consciousness from them; nevertheless, it is obviously not their top priority.

Fewer normal users is a help as well. Poorly chosen passwords are a serious risk; if users and their attendant passwords do not exist, this isn’t a problem. Similarly, one can make more or less arbitrary changes to various program interfaces if that would help security, without annoying a population that is accustomed to a different way of doing things. One example would be the use of hand-held authenticators for logging in (Chapter 5). Many people resent them, or they may be too expensive to be furnished to an entire organization; a gateway machine, however, should have a restricted-enough user community that these concerns are negligible.

More subtly, gateway machines need not, and should not, be trusted by *any* other machines. Thus, even if the gateway machine has been compromised, no others will fall automatically. On the other hand, the gateway machine can, if you wish (and if you decide against using hand-held authenticators), trust other machines, thereby eliminating the need for most passwords on the few accounts it should have. Again, something that is not there cannot be compromised. (Other components of the firewall can shield vulnerable services on the gateway machine; see Chapter 3.)

Gateway machines have other, nonsecurity advantages as well. They are a central point for mail and FTP administration, for example. Only one machine need be monitored for delayed mail, proper header syntax, return-address rewriting (i.e., to *Firstname.Lastname@ORG.DOMAIN* format), etc. Outsiders have a single point of contact for mail problems and a single location to search for files being exported.

Our main focus, though, is security. And for all that we have said about the benefits of a firewall, it should be stressed that we neither advocate nor condone sloppy attitudes towards host security. Even if a firewall were impermeable, and even if the administrators and operators never made any mistakes, the Internet is not the only source of danger. Apart from the risk of insider attacks—and in some environments, that is a serious risk—an outsider can gain access by other means. In at least one case, a hacker came in through a modem pool, and attacked the firewall from the *inside* [Hafner and Markoff, 1991]. Strong host security policies are a necessity, not a luxury. For that matter, internal firewalls are a good idea, to protect very sensitive portions of organizational networks. AT&T uses them; we leave to your imagination exactly what is being protected.

1.3.3 Protecting Passwords

ԲԱՐՐԿ ԵՂՇՇԱՌ Ը ԵՂՈՒԿ
(Speak, friend, and enter.)

“What does it mean by *speak, friend, and enter*?” asked Merry.


“That is plain enough,” said Gimli. “If you are a friend, speak the password, and the doors will open, and you can enter.”

...

“But do not *you* know the word, Gandalf?” asked Boromir in surprise.

“No!” said the wizard. . . . “I do not know the word—yet. But we shall soon see.”

Lord of the Rings
—J.R.R. TOLKIEN

 System bugs are the exciting way to crack a system, but they are not the most common attack. That honor is reserved for a rather mundane feature: user passwords. A high percentage of system penetrations occur because of the failure of the entire password system.

We write “password system” because there are several causes of failure. However, the most common problem is that people tend to pick very bad passwords. Repeated studies have shown that password-guessing is likely to succeed; see, for example, [Klein, 1990] or [Morris and Thompson, 1979]. We are not saying that *everyone* will pick a poor password; however, enough people will that password-guessing remains a high-probability approach for an attacker.

Password-guessing attacks take two basic forms. The first involves attempts to log in using known or assumed user names and likely guesses at passwords. This succeeds amazingly often;

```

root:DZo0RWR.7DJuU:0:2:0000-Admin(0000):/:
daemon*:1:1:0000-Admin(0000):/:
bin*:2:2:0000-Admin(0000):/bin:
sys*:3:3:0000-Admin(0000):/usr/v9/src:
adm*:4:4:0000-Admin(0000):/usr/adm:
uucp*:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp*:10:10:0000-uucp(0000):/usr/spool/uucppublic:/usr/lib/uucp/uucico
ftp:anonymous:71:14:file transfer:/:no soap
research:nologin:150:10:ftp distribution account:/forget:/it/baby
ches:La9Cr9ld9qTQY:200:1:me:/u/ches:/bin/sh
dmr:laHheQ.H9iy6I:202:1:Dennis:/u/dmr:/bin/sh
rtm:5bHD/k5k2mTTs:203:1:Rob:/u/rtm:/bin/sh
adb:dcScD6gKF./Z6:205:1:Alan:/u/adb:/bin/sh
td:deJCw4bQcNT3Y:206:1:Tom:/u/td:/bin/sh

```

Figure 1.2: The bogus `/etc/passwd` file in our anonymous FTP area.

sites often have account-password pairs such as *field-service*, *guest-guest*, etc. These pairs often come out of system manuals! The first try may not succeed, nor even the tenth, but all too often, one will work—and once the attacker is in, your major line of defense is gone. Regrettably, few operating systems can resist attacks from the inside.

This approach should not be possible! Users should not be allowed an infinite number of login attempts with bad passwords, failures should be logged, users should be notified of failed login attempts on their accounts, etc. None of this is new technology, but these things are seldom done, and even more seldom done correctly. Many common mistakes are pointed out in [Grampp and Morris, 1984], but few developers have heeded their advice. Worse yet, much of the existing logging on UNIX systems is in *login* and *su*; other programs that use passwords—*ftpd*, *rexecd*, various screen-locking programs, etc.—do not log failures on most systems.

The second way hackers go after passwords is by matching guesses against stolen password files (`/etc/passwd` on UNIX systems). These may be stolen from a system that is already cracked, in which case the attackers will try the cracked passwords on other machines (users tend to reuse passwords), or they may be obtained from a system not yet penetrated. These are called *dictionary attacks*, and they are usually very successful. Make no mistake about it: if your password file falls into enemy hands, there is a very high probability that your machine *will* be compromised. Klein [1990] reports cracking about 25% of the passwords; if that figure is accurate for your machine, and you have just 16 user accounts, there is a 99% chance that at least one of those passwords will be weak.

A third approach is to tap a legitimate terminal session and log the password used. With this approach, it doesn't matter how good a password you have chosen; your account, and probably your system, is compromised.

We can draw several conclusions from this. The first, of course, is that user education in how to choose good passwords is vital. Sadly, although almost 15 years have passed since Morris and Thompson's paper [Morris and Thompson, 1979] on the subject, user habits have not improved much. Nor have tightened system restrictions on allowable passwords helped that much, although there have been a number of attempts, e.g., [Spafford, 1992b; Bishop, 1992]. Others have tried

How Long Should a Password Be?

It is generally agreed that the eight-character limit that UNIX systems impose is inadequate [Feldmeier and Karn, 1990; Leong and Tham, 1991]. But how long should a password be?

Part of the problem with the UNIX system's password-hashing algorithm is that it uses the seven significant bits of each typed character directly as an encryption key. Since the algorithm used (DES; see Chapter 13) permits only 56 bit keys, the limit of eight is derived, not selected. But that begs the question.

The 128 possible combinations of seven bits are not equally probable. Not only do most people avoid using control characters in their passwords, most do not even use characters other than letters. Most folks, in fact, tend to pick passwords composed solely of lowercase letters.

We can characterize the true value of passwords value as keys by using *information theory* [Shannon, 1949]. For ordinary English text of 8 letters, the information content is about 2.3 bits per letter, perhaps less [Shannon, 1948, 1951]. We thus have an effective key length of about 19 bits, not 56 bits, for passwords composed of English words.

Some people pick names (their own, their spouse's, their children's, etc.) for passwords. That gives even worse results, because of just how common certain names are. Experiments performed using the AT&T online phone book show that a first name has only about 7.8 bits of information in the whole name. These are very bad choices indeed.

Longer English phrases have a lower information content per letter, on the order of 1.2 to 1.5 bits. Thus, a password of 16 bytes is not as strong as one might guess if words from English phrases are used; there are only about 2^{19} to 2^{24} bits of information there. The situation is improved if the user picks independent words, to about 2^{38} bits. But if users fill up those bytes with combinations of names, we have not helped the situation much.

to enforce password security through retroactive checking [Muffett, 1992]. But perversity always tends toward a maximum, and the hackers only have to win once.

If you cannot keep people from choosing bad passwords, it is vital that the password file itself be kept out of enemy hands. This means that one should

- carefully configure the security features for services such as Sun's NIS,
- restrict files available from *tftpd*, and
- avoid putting a genuine `/etc/passwd` file in the anonymous FTP area.

(There is room for fun, of course. Our *ftpd* will happily deliver `/etc/passwd` file to you (Figure 1.2), complete with passwords crackable by trying words from a dictionary [Klein, 1990]. They come to “why are you wasting your time”. The first of these, nominally for *root*, has shown up on a hacker bulletin board, which says something about hacker quality control.)

Some UNIX systems provide you with the ability to conceal the hashed passwords from even legitimate users. If your system has this feature (sometimes called a “shadow” or “adjunct” password file), we strongly urge you to take advantage of it. Many other operating systems wisely hash and hide their password files.

1.3.4 Encryption

Encryption is often touted as the ultimate weapon in the computer security wars. It is not. It is certainly a valuable tool (see Chapter 13), but it, like everything else, is a tool toward an ultimate goal. Indeed, if encryption is used improperly, it can hurt the real goals of the organization.

Some aspects of improper use are obvious. One must pick a strong enough cryptosystem for the situation, or an enemy might cryptanalyze it. Similarly, the key distribution center must be safeguarded, or all of your secrets will be exposed.

Other dangers exist as well. For one thing, encryption is best used to safeguard file transmission, rather than file storage, especially if the encryption key is generated from a typed password. Few people bequeath knowledge of their passwords in their wills; more have been known to walk in front of trucks. There are schemes to deal with such situations (e.g., [Shamir, 1979; Gifford, 1982; Blaze, 1994]), but these are rarely used in practice. Admittedly, you may not be concerned with the contents of your files after your untimely demise, but your organization—in some sense the real owner of the information you produce at work—might feel differently.

Even without such melodrama, if the machine you use to encrypt and decrypt the files is not physically secure, a determined enemy can simply replace the cryptographic commands with variants that squirrel away a copy of the key. Have you checked the integrity of such commands on your disk recently? Did someone corrupt your integrity-checker?

Finally, the biggest risk of all may be your own memory. Do you remember what password you used a year ago? (You do change your password regularly, do you not?) You used that password every day; how often would you use a file encryption key?

If a machine is physically and logically secure enough that you can trust the encryption process, encryption is most likely not needed. If the machine is not that secure, encryption may not help.

There is one exception to our general rule: backup tapes. Such tapes rarely receive sufficient protection, and there is never any help from the operating system. One can make a very good case for encrypting the entire tape during the dump process—if there is some key storage mechanism guaranteed to permit you to read the year-old backup tape when you realize that you are missing a critical file. It is the *information* that is valuable; if you have lost the contents of a file, it matters little if the cause was a hacker, a bad backup tape, a lost password, or an errant *rm* command.

1.4 The Ethics of Computer Security

Sed quis custodiet ipsos custodes? (But who will guard the guards themselves?)

Satires, VI, line 347
—JUVENAL, C. 100 C.E.

At first blush, it seems odd to ask if computer security is ethical. We are, in fact, comfortable with what we are doing, but that is because we have asked the question of ourselves, and then answered it to our own satisfaction.

There are several different aspects to the question. The first, of course, is whether or not computer security is a proper goal. We think so; if you disagree with us about that, there is probably a deep philosophical chasm between you and us, one that we may not be able to bridge. We will therefore settle for listing our reasons, without any attempt to challenge yours.

First, in a technological era, computer security is fundamental to individual privacy. A great deal of very personal information is stored on computers. If these computers are not safe from prying eyes, neither is the data they hold. Worse yet, some of the most sensitive data—credit histories, bank balances, and the like—lives on machines attached to very large networks. We hope that our work will in some measure contribute to the protection of these machines.

Second, and more important, computer security is a matter of good manners. If people want to be left alone, they should be, whether or not you think their attitude makes sense. Our employer demonstrably wants its computer systems to be left in peace. That alone should suffice, absent an exceedingly compelling reason for feeling otherwise.

Third, more and more of modern society depends on computers, and on the integrity of the programs and data they contain. These range from the obvious (the financial industry comes to mind) to the ubiquitous (the entire telephone system is controlled by a vast network of computers) to the life-critical (computerized medical devices and medical information systems). The problems caused by bugs in such systems are legion; the mind boggles at the harm that could be caused—intentionally or not!—by unauthorized changes to any such systems. Computer security is as important in the information age as were walled cities a millennium ago.

A computer intrusion has even been blamed for loss of life. According to Scotland Yard, an attack on a weather computer stopped forecasts for the English Channel, and that led to the loss of a ship at sea [Markoff, 1993b].

That the hackers behave badly is no excuse for us doing the same. We can and must do better.

Consider the question of “counterintelligence,” the activities we undertake to learn who has been pounding on our door. Clearly, it is possible to go too far in that direction. We do not, and will not, attempt to break into the malefactor’s system in order to learn more about the attacks. Similarly, when we found that our machine was being used as a repository for pirated software, we resisted the temptation to replace those programs with virus-infected versions. (But we did joke about it.)

On the other hand, we do engage in activities that ring alarms if someone does the same thing to us. For example, given that we log *finger* attempts, and trace back *rusers* calls, are we justified in using those protocols ourselves? We also use *telnet* to connect to various services in an attempt to learn a usable machine name. Is this an unethical probe of someone else’s system? On occasion, we have had mail to a site administrator bounce; we have had to resort to things like hand-entered *VERFY* commands on the SMTP port to determine where the mail should be sent. Is that proper?

Lures are somewhat more problematic. For example, our *finger* daemon will inform the curious that *guest* is logged in, which is not the case; we have no real *guest* login. But the dummy message tends to generate lots of attempts to use this nonexistent account, which in turn generates lots of noise in our log files. Are we entrapping folks? This is a borderline case (though we should note that real *guest* accounts are extremely rare these days); we are careful *not* to send our usual warning notes in response to isolated attempts. Even repeated attempts are more likely to generate a “please stop it; the log messages are bothering us” than a complaint to the site’s system administrator.

The ethical issues go even further. Some people have suggested that in the event of a successful attack in progress, we might be justified in penetrating the attacker’s computers under the doctrine of self-defense. That is, it may be permissible to stage our own counterattack in order to stop an immediate and present danger to our own property. The legal status of such an action is quite murky, although analogous precedents do exist. Regardless, we have not carried out any such action, and we would be extremely reluctant to. If nothing else, we would prefer to adhere to a higher moral standard than might be strictly required by law.

Overall, we are satisfied with what we are doing. Within the bounds set by legal restrictions (see Chapter 12), we do not regard it as wrong to monitor our own machine. It is, after all, *ours*; we have the right to control how it is used, and by whom. (More precisely, it is a company-owned machine, but we have been given the right and the responsibility to ensure that it is used in accordance with company guidelines.) Most other sites on the Internet feel the same way. We are not impressed by the argument that idle machine cycles are being wasted. They are our cycles: we will use them as we wish. Most individuals’ needs for computing power can be met at a remarkably modest cost. Finally, given the current abysmal state of host security, we know of no other way to ensure that our firewall itself is not compromised.

Equally important, the reaction from system administrators whom we have contacted has generally been quite positive. In most cases, we have been told that either the probe was innocent, in which case nothing is done, or that the attacker was in fact a known troublemaker. In that case, the very concept of entrapment does not apply, since by definition entrapment is an inducement to commit a violation that the victim would not otherwise have been inclined to commit. In a few cases, a system administrator has learned, through our messages, that his or her system was itself

compromised. Our peers—the electronic community of which we are a part—do not feel that we have abused their trust.

1.5 WARNING

In the past, some people have interpreted our descriptions of our security mechanisms as an invitation to poke at us, just to see if we would notice. We are sure, of course, that their hearts were pure. Conceivably, some of you might entertain similar misconceptions. We therefore humbly beseech you, our *gentle readers*:

PLEASE DON'T.

We have quite enough other things to do; it is a waste of your time and ours, and we don't really need the extra amusement. Besides, AT&T Corporate Security has no sense of humor in such matters.