If you are new to security, I recommend that you start with Chapter 2, "Quick Fixes for Common Problems." At first glance, this chapter sounds simple and sweet. However, it covers the security problems that are so severe they are likely to result in a rather unpleasant and expensive break-in. These problems affect all unhardened Linux systems regardless of their use. This chapter is not full of fluff, theoretical ideas, or impractical suggestions. Instead, it is a hundred pages of no-nonsense, thoroughly explained, practical advice that has proven successful for many systems over the course of time.

Chapter 2’s section entitled "The Seven Most Deadly Sins" explains exactly these. Indulge any of these seven sins and almost certainly your systems will be broken into in one to twelve months. I find that the difference between Linux and Windows is that with knowledge and a little effort, Linux can be made very secure. I provide that knowledge in Real World Linux Security, Second Edition, along with some humor and real world examples of successes and failures. Following the easy advice in the pages of this section will boost the security of an "out of the box" Linux system by an order of magnitude. Shall we begin?

Bob Toxen

2.2 The Seven Most Deadly Sins

These are the seven common problems most likely to allow major damage to occur to your system or bank account. If any of these are currently a problem on your systems, you should take care of them immediately.¹

2.2.1 Weak and Default Passwords (#1)

As a system administrator, you are aware of the system breaches possible on your Linux machine. You have taken the time and effort to devise a difficult-to-guess password that uses at least eight characters, uses both letters and numbers, upper- and lowercase letters, and possibly some punctuation. Your root password is awesome—no one could guess it in a hundred years. (OK, some obsessive with a decrypt package could destroy it in a few days except that you use shadow passwords, but that is another story.)

¹ This section was generously written for this book by Larry Gee, an extremely experienced Linux System Administrator and client/server C and Perl programmer.
How are your users doing? Choke, cough, gag, hack. Every account is a possible entry point. Have your users followed your advice, company policy, or threats to devise a good password? Are they being as careful as you are? Probably not. Now it is your turn to don the black hat and think like your enemy.

Using programs like `crack` (which cracks passwords), can you break into your users’ accounts? You definitely will need to get written management approval to conduct this level of security audit. There are notable cases of unauthorized audits landing people in jail or at least on the unemployment rolls. Randal Schwartz is one.

You might even install a module in the `passwd` program that automatically tries to break a user’s proposed new password. Though the standard `passwd` program makes very simple tests, there are more sophisticated routines that include much of `crack`’s capability. One way to do this is to make use of the `cracklib` capability in the Pluggable Authentication Modules (PAM) enhancements to the `passwd` program. The `cracklib` library analyzes passwords to determine whether they are easily crackable. PAM offers additional security for Linux systems and other operating systems too. Edit the `/etc/pam.d/passwd` file to include:

```
passwd password requisite /usr/lib/security/pam_cracklib.so retry=3
passwd password required /usr/lib/security/pam_pwdb.so use_authtok
```

This will cause the PAM-enabled `passwd` program to load these dynamically loadable program libraries. PAM now is standard with Red Hat. On some systems these are in `/lib` instead of `/usr/lib`.

There is more documentation on configuring PAM-enabled utilities at:

```
www.kernel.org/pub/linux/libs/pam/
```

Another good source for PAM information is:

```
www.sun.com/software/solaris/pam/
```

On Slackware, this capability will be enabled if the following line is present in `/etc/login.defs` (and the dictionary is installed):

```
CRACKLIB_DICTPATH /var/cache/cracklib/cracklib_dict
```

See also “Restricting Login Location and Times” on page 315.
2.2 The Seven Most Deadly Sins

2.2.2 Open Network Ports (#2)

Just as every account on your system is a potential path for a cracker, every network service is a road to it. Most Linux distributions install “tons” of software and services by default. They deliberately prefer “easy” over “secure.” Many of these are not necessary or wanted. Take the time to remove software and services you do not need. Better still—do not install them to begin with.

To find out what services are being run, use the `netstat -atuv` command or use the `ports` program discussed in “Turn Off Unneeded Services” on page 86. Either will list all open ports on your system. Even a home system can have dozens or hundreds of ports open. A large Web server could have many more.

If there are services listed that you do not want to be provided by this box, disable them. Many distributions offer a Control panel to do this easily, including Red Hat and Mandrake. You might want to remove the binaries from the disk or `chmod` them to 0, especially any that are set-UID or set-GID.

NFS, finger, the shell, exec, and `login r*` services (rsh, rexec, and rlogin), FTP, `telnet`, `sendmail`, DNS, and `linuxconf` are some of the more popular services that get installed by default on many Linux distributions; at least some of these should not be enabled for most systems. Most of these are controlled by the inet daemon, `inetd`; these can be disabled by editing the `/etc/inetd.conf` file.

You do not need the FTP or `telnet` daemons to use the respective clients to connect into other systems. You do not need the `sendmail` daemon listening on port 25 to send mail out or to send mail to local users or to download mail via POP or IMAP. (You do need to invoke `sendmail` periodically to de-spool delayed outgoing mail. The techniques are explained in “Hardening for Very High Security” on page 306.) You only need DNS (`named`, the name daemon) if other systems will be querying yours for this data. Most programs running on
your own system will be very happy to read /etc/resolv.conf and query your ISP’s or organization’s main DNS server instead of contacting a named process running on your system. Coincidentally named’s ports are some of the most popular ports that crackers use to break into systems. If you do need to run named, use the recently added facilities that allow it to chroot itself and switch to a nonroot user.

All these services, except the normal installation of NFS,2 DNS, and sendmail, are started on demand by inetd. They may be turned off by commenting out their entries in /etc/inetd.conf. Many distributions offer a Control panel or Linuxconf to do this easily, including Red Hat and Mandrake. The standalone services are turned off by altering their entries under /etc/rc.d.

On Red Hat–based systems, issue the following commands to shut down portmap and prevent it from being restarted on reboot. Even as late as Red Hat 7.3 on a standard non-server install, the evil portmap is invoked.

```
/etc/rc.d/init.d/nfs stop
/etc/rc.d/init.d/nfslock stop
/etc/rc.d/init.d/portmap stop
chkconfig --del nfs
chkconfig --del nfslock
chkconfig --del portmap
```

An alternative tool is the ASCII menu-based ntsysv program. Like chkconfig, ntsysv only manipulates the symbolic links under /etc/rc.d/rc[0-6].d so you also will need to explicitly shut down the service. To do both of these issue the commands:

```
/etc/rc.d/init.d/portmap stop
ntsysv
```

On other distributions that use the System V-style of startup scripts (/etc/rc.d/rc[0-6].d directories for Red Hat derivations and /etc/rc.[0-b].d for Debian), rename the appropriate script under rcX.d (X usually is 3) that starts with a capital-S and has the service name in it. For example,

```
cd /etc/rc.d/rc3.d
mv S11portmap K11portmap
```

Just as only scripts starting with “S” are invoked when entering the respective run level, scripts starting with “K” are invoked when exiting that run level. This is to turn off daemons that should run only in that run level. For example, this mechanism will turn off sshd, the

---

2. NFS consists of these daemons: rpc.nfsd, mountd, portmap, rpc.lockd, rpc.statd, rpc.rquotad, and automounter, scattered among a number of startup scripts. Technically, portmap is not part of NFS. It is a part of the underlying Remote Procedure Call (RPC) layer but on many systems it is used only for NFS. When an RPC server, such as NFS, is started, that server will tell portmap both its portmap service number and its TCP or UDP port number. A client then will ask portmap, “What TCP or UDP port should I use for portmap service X?” A cracker process can lie to portmap and masquerade as a legitimate server. NFS has had plenty of security bugs in the past and its design prevents its being made secure in many configurations.
2.2 The Seven Most Deadly Sins

SSH daemon, when switching from run level “3” (multiuser with networking) to run level “s” (single-user mode). Just as a selected $something script can be disabled by renaming to $something, one of these latter scripts can be renamed from $something to k$something to disable it.

On Slackware and similar systems, simply comment out the lines that start them in /etc/rc.d/*. The grep program may be used to find these. Be sure to terminate any of these services that currently are running on your system after altering the configuration files.

If you do not want to bother with kill, a simple reboot will do this and verify that the configuration files were correctly altered. (A set of available rescue disks before this reboot would be a fine idea.)

The most careful SysAdmins will reboot their systems several times after making changes to startup scripts to ensure correct and reliable startup and operation.

To remove these services from your system, you can use your distribution’s package manager to delete them. Red Hat–based installations (Red Hat, Mandrake, Caldera, Yellow Dog, TurboLinux) use RPM. Debian-based distributions (Debian and Corel) use dpkg. SuSE uses YAST and Slackware uses pkgtool.

Linux is like the Swiss Army knife of networking—it has one or two tools of mass destruction that get used all the time, others that are used less often, and some that are never used. Unlike the Swiss Army knife, you can slim down Linux to just the services you need, and discard those you do not. I will never use the awl or corkscrew on my knife just like I will never use rsh or finger. Decide which ports you want to have open (such as www and ftp) and close the rest. Closing unnecessary ports makes your system more secure and perform better.

2.2.3 Old Software Versions (#3)

Linux is not perfect. There are new vulnerabilities being found monthly. Do not despair, though. The speed with which problems are found and fixed in Linux is the fastest on the planet. Your challenge as an administrator is to keep up with the changes.

Each distribution has a mailing list that you can (and should) subscribe to, where security bulletins are issued, and an FTP or Web site at which the fix will be available. Also, there are the independent security mailing lists, such as Bugtraq and X-Force’s Alert, that are excellent. Get yourself tapped into those mailing lists.

---

3. Most recent Linux vulnerabilities are not exploitable remotely on most systems. This means that most Linux systems are not at risk for remote attack via the Internet. Many of the vulnerabilities may be taken advantage of only by someone with a regular account on the particular Linux system. Other vulnerabilities are in programs that most people do not use. In my opinion, this is different from most Windows vulnerabilities where almost every client system or Web server using that major version of Windows is vulnerable to remote attack via the Internet and thus to complete control by crackers.
Other good sources of Linux security information are:

www.lwn.net/
http://linuxtoday.com/

They are distribution-neutral and carry all the major distributions’ security advisories. There is much more information on Web resources in Appendix A.

One of the beauties of Linux is that when a fix is issued, it is very quick to install, and unless it is in the kernel, your downtime for that service is on the order of seconds or minutes. Rarely, if ever, is a reboot necessary.

2.2.4 Insecure and Badly Configured Programs (#4)

The number of security bugs and their severity in commonly used Linux programs have been reduced so dramatically that I have dropped the subject of poor physical security from the most deadly sins. In its place is the use of insecure programs (such as rsh, NFS, portmap, and FTP) in other-than-carefully-controlled situations and the failure to properly configure other programs. These “other programs” are capable of good security only if properly configured. Therefore, the Seven Most Deadly Sins has been updated for the second edition to reflect this.

This pushes poor physical security off the list and into the new “must read” section, “Law of the Jungle—Physical Security” on page 121. Additionally, Deadly Sin #5, Insecure CGIs, has been merged with this one. This is because, while locally written or adapted insecure CGI programs continue to be a problem, a substantial percentage of problems are due to SysAdmins making use of known insecure features of programs or failing to take advantage of security features.

Most SysAdmins know that POP and IMAP (unless wrapped in SSL), telnet, and FTP\(^4\) send passwords in the clear. They know that NFS and portmap have a history of security problems as well as design defects in their authentication. Many use them anyway and then are surprised when they get broken into. Do not do that! Instead, use spop, simap, SSH, and

\(^4\) If you are doing anonymous FTP, your password is normally your e-mail address. Unless you are a government researcher at Groom Lake (Area 51) and you do not want to acknowledge the existence of such a facility, then generally you have nothing to worry about.
2.2 The Seven Most Deadly Sins

SSH’s scp or sftp. See “Replace These Weak Doors with Brick” on page 94, “New Lamps for Old” on page 103, and “United We Fall, Divided We Stand” on page 115.

Many programs are secure only if properly configured, and it is common for SysAdmins to configure them improperly. Sometimes, it is a lack of training and understanding of the risks, while other times use of an insecure feature is deliberate, because “I just gotta have it.” A recent case in point is Apache’s PHP capability. It has had a recent history of security problems. PHP’s security problems have been well publicized, and still some people cannot seem to use it securely or find an alternative. Security and convenience often are contradictory; frequently, a choice must be made. Chapter 4, “Common Hacking by Subsystem,” on page 171 and Chapter 6, “Advanced Security Issues,” on page 261 may be helpful.

Before deciding to deploy a service (i.e., changing what capabilities will be used or how the service will be deployed), research its security. Check its security history and understand how it may be deployed securely. If it cannot be deployed securely, what are the secure alternatives? I still encounter people using FTP who don’t realize that sftp is an excellent alternative. Putting an insecure service, in this case NFS, behind a firewall was a good solution for one client. For another, putting its insecure Windows networks behind firewalls, with its different offices linked via a VPN between these same Linux firewalls, offered excellent security. Another client had me configure a firewall with separate subnets for its students, its financial administration and human resources, and the rest of its employees, along with internal and external DMZs. The rest of this section will address CGI problems specifically.

A CGI program will allow anyone to access your Web site, good intentions or not. Although other “accepted” servers, such as sendmail and named, will also talk with anyone, the scope of what a client may request is far smaller. While these latter servers have had their share of serious security bugs, those that keep up-to-date (as discussed in “Old Software Versions (#3)” on page 31) have minimal risk. Here are a few hard, fast rules that will help you make your Web site secure. (See “Special Techniques for Web Servers” on page 284 for several ways to increase Web server security.)

A. **Never, ever, run your Web server as a privileged user (such as root).**

Even some documentation for various products tries to seduce you into running Apache as root. You will get scalped if you do.

B. **Know your data (that is supplied by Web clients).**

- **Establish maximums and minimums for data entry values and lengths of fields.**
- **Decide what characters are acceptable in each field.** Expect the malicious to send you control characters and non-ASCII bytes. Expect that crackers will use the “%” encoding to generate these evil characters. (“%” encoding is a “%” followed by two hexadecimal characters that will be mapped into the equivalent ASCII character.) Unless you stop them, they will use this method to send your CGI programs arbitrary binary bytes. This makes it easy to overflow input buffers and drop machine code (instructions) directly into CGI programs.
Thus, you need to check for illegal characters both before and after “%” conversion. I have seen this latter attempt used against our sunset CGI program, fortunately without success.

- **Double-check each entered value.** A surprising number of shopping cart packages put the price of items in the form and believe the price in the filled-out form sent by the user. All a user needs to do is to alter this form and give himself a discount. Very little skill is required for a user to use this exploit. Many sites never detect the loss. It has been reported that Cart32 versions 2.5a, 2.6, and 3.0 have this “price in the form” bug and that even though this bug was widely known for four months, the vendor has chosen not to repair this problem.\(^5\)

- **If possible, enumerate the allowed values instead of using ranges.**

- **Understand, too, that an evil Web client can send any bytes back to your server.** The cracker might copy and alter your Web form so that, even if your form pops up a list of the member European Union countries, she could supply

  ```
  crash_with_a_long_name_having_control_characters
  ```

- **Use a secure language.** Client-supplied data should never be handed directly to a shell script; there are too many opportunities for a cracker to get herself a shell or to exploit a buffer overflow vulnerability. Do you know how bash or tcsh react to buffer overflows? Neither do I, so you must not trust them in situations where a buffer overflow is possible.

  For many, that secure language will be C, Perl, or Python. If that language offers checking for tainted data, use it. One language does not fit all. Use what is best for each CGI, consistent with programmer skills. Perl has a number of features to enable safer CGI programs.\(^6\) These include the “tainted data” feature, the `-w` flag to warn you about things that you are creating but not using, and the strict capability that is discussed in

  ```
  ```

  Even more security can be achieved with perlsec, discussed in

  ```
  ```

C. **Analyze CGIs for vulnerabilities.**

When writing CGI programs, look at them the way a cracker would and try to break them. Check for buffer overflows by using good programming techniques. For example, when using C, make use of the `fgets()` routine which allows the programmer to specify how large a buffer has been allocated and will not overflow it. An easy way to determine if the

---

6. Most of the information on Perl presented here is from section 8 of Kurt Seifried’s writings.
2.2 The Seven Most Deadly Sins

line is larger than the buffer is to see that it does not end with a newline character, as this example illustrates.

```c
#include <stdio.h>
#include <string.h>

int c;
char buf[200];

if (!fgets(buf, sizeof buf, stdin))
    error();
else if (!strchr(buf, '\n')) {
    /* Read rest of long line. */
    while ((c = getchar()) != EOF
        && c != '\n')
        ;
    overflow();
}
```

Do not use the `gets()` routine because it does not do any checking for buffer overflows; use `fgets()` instead. Many of the other popular C string functions have similar weaknesses. The `strcpy()` function, for example, “lets” you copy a large buffer into a small buffer, overwriting unrelated memory. The `strncpy()` function is a good replacement for it.

---

A safe way to copy strings is:

```c
strncpy(dest_buf, source_buf, sizeof dest_buf);
dest_buf[sizeof dest_buf - 1] = '\0';
```

To detect a problem, one possibility is:

```c
if (strlen(source_buf) >= sizeof dest_buf)
    error();
else
    strcpy(dest_buf, source_buf);
```

Check for escape sequences, the possibility of a client issuing Linux commands (by inserting spaces, quotes, or semicolons), binary data, calls to other programs, etc. Often it is safer to have a list of allowed characters rather than trying to determine each unsafe character.

The following C code may be used to process a field that the client should supply her name in. In this example, the calling process supplies a NUL-terminated string and this routine returns 0 if the string is a legal name and –1 otherwise. The second argument specifies
the maximum legal string allowed, including the terminating NUL byte. Note that the calling routine must be careful to ensure that its buffer does not overflow. I chose clear code over slightly more efficient code.

#include <string.h>

char    okname[] = " .'abcdefghijklmnopqrstuvwxyz"
         "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
/* Return 0 on legal names, -1 otherwise. */
legal(char *name, int maxlen)
{
    if (!name || !*name
         || strlen(name) >= maxlen)
        return -1;
    while (*name)
        if (!strchr(okname, *name++))
            return -1;
    return 0;
}

Many of the system break-ins relating to Linux Web servers happen via insecure CGIs. All it takes is one buggy CGI and most systems will break.

D. Implement “Rings of Security” in CGIs.

Try to design your application so that even if a CGI vulnerability is found, the system is protected from major damage. One solution is to have CGIs be just front ends for a solidly written server running on a different machine. The more hurdles a cracker must jump to reach the goal, the more likely it is that he will stumble.

Consider using the SubDomain kernel module from WireX. It allows implementing kernel-based access control lists on a per-program basis to add additional limits to what any given program may do. It can prevent many of the recent security breaches, such as the named bug and the Dansie “fiasco,” both discussed elsewhere in the book. Its object is only about 24 KB and is easy to install; WireX is generous with its allowance for free use. Its developer previously created StackGuard. It is available at:

www.wirex.com/subdomain.html

E. Watch for bug reports in third-party CGIs and inspect their code.

If using third-party-supplied CGI scripts (such as shopping carts), subscribe to their mailing lists and watch for security bulletins. If possible, get the source code and review it. If you do not know the language, get people who do and have them review it.

In the spring of 2000 Joe Harris made a posting to Bugtraq, one of the security mailing lists, claiming that a popular Perl-based shopping cart program, Dansie, has a back door. Harris claimed that this back door lets anyone who knows the secret form name (URL on the server’s system) to execute any arbitrary Linux command as the user running the shop-
2.2 The Seven Most Deadly Sins

Pinging cart software. Dansie denied the existence of the back door but someone else also claimed to have seen this problem and it was mentioned in *InternetNews* on April 13, 2000.

No customers have disputed the claim of a back door and Dansie’s phone was disconnected shortly after this matter became public. I am not convinced that Dansie’s denial is true because a second alleged user confirmed the problem and because there were no reports of Dansie denying the back door; Dansie gave a vague response to my e-mail sent to the company following the news reports: **Inspect source code.** A source quoted in an article on the Dansie problem said that this destroyed his confidence in the product.

But Dansie ain’t got *(sic)* nothin’ on Red Hat. On April 25, 2000 it was reported that Red Hat left an undocumented back door in its Piranha product that allows anyone who knows it to execute arbitrary commands on the server as whatever user is running the product. This unintentional bug was in version 0.4.12 of piranha-gui. Patches are available.

Many shopping cart packages and CGIs, both commercial and open source, have severe security holes that are well-known to the cracker community. We see our sites probed for these vulnerabilities periodically and we do not even handle money online.

**F. Avoid creating and using set-UID and set-GID programs to the maximum extent possible (and try real hard).**

Many of these programs run as root. Where root is found on a system, a cracker is not far behind, probing for weaknesses. Many programs that run set-UID to root (or which are invoked by root) do not need to be.

Frequently all these programs need to be set-UID for is to run as *some* user to gain access to data that should not be world-accessible. The solution is to create a new user that has no other purpose or access. Then make such programs set-UID to that new user. Different programs might need to be set-UID to different users to protect them from each other.

**G. Do not keep clients’ confidential data on the Web server.**

Avoid storing users’ privileged data (credit card numbers, financial details, mailing addresses, and phone numbers, etc.) on the same machine as the Web server. This separation will force a cracker to have to crack two systems instead of just one to get this data. (See “One-Way Credit Card Data Path for Top Security” on page 302 for an innovative solution to this problem.)

**H. Do not include users’ confidential data (credit card numbers, financial details, mailing addresses, phone numbers, and so forth) in a URL or cookie.**

Frequently this is done as arguments to a CGI program, for example:

```plaintext
http://www.abroker.com/cgi-bin/address_change?account=666
?passwd=secret&addr=1+Maple+St.&phone=555-1212
```

---

7. Fidelity Investments, which manages $900 billion of its customers’ money, did not follow this advice. In May 2002, it was reported that by changing the digits in the URL of the page displaying his statement—a three-digit number—a client saw other clients’ statements.
The problem is that this information will show as the “referring URL” if the user then clicks on a link shown on the results page. Thus, this privileged information will be given to complete strangers (the Web site that the link points to). Several well-known companies were scandalized by this in early 2000 and some were investigated by the authorities. Some browsers might store this URL (containing confidential data) in a history file. If someone is browsing from a public terminal, such as a school or library, you could be liable for careless handling of that person’s data. Similar issues are present for cookies.

**I. Be very sure that the privileged data that a user supplies on a form does not show up as the default data for the next person to “pull down” that form and see.**

Yes, this actually has happened.

**J. Always protect the user who types in his password.**

Take him to a secured area prior to this information being entered. This will decrease the likelihood that the password will be stolen. Commonly this protected area will be https (SSL-wrapped http) to encrypt his password to guard against network sniffing.

### 2.2.5 Insufficient Resources and Misplaced Priorities (#5)

In many organizations, management simply will not approve sufficient resources to allow SysAdmins to provide good security. Good security does not happen by accident. It takes many elements to achieve a truly comprehensive security solution. Education, design, proper implementation, user training, maintenance, and continual vigilance all are required for an organization’s system to be secure. When security is not supported (i.e., funded) by an organization, it is frequently limited to what a SysAdmin is willing to do on his own time. Yet, if he is unwilling to spend the time, he certainly will be blamed for any violations. This deadly sin puts the SysAdmin in the middle of problems that are not his direct responsibility. In other words, management will not allow him to make the changes necessary for good security and good business.

This may not be a “technical” problem, but I have found it to be the cause of break-ins at numerous organizations. Sadly, most of my clients come to me only after they have suffered a major, expensive break-in. I would much rather help people maintain good security than to do a painful and expensive repair.

A lack of resources commonly is due to misplaced priorities. “We have not been broken into and the media exaggerates every danger well beyond the true risk.” This is a common belief of those whose organizations have not been broken into. I have deliberately peppered this book with accounts of break-ins in the hope that they may make an impression on management somewhere. Consider making a present to your boss of Bruce Schneier’s excellent book, *Secrets and Lies: Digital Security in a Networked World*. *Secrets and Lies* is aimed at management and makes a good companion to this book. Some of the examples are the same as in this book, as both books were written at about the same time and they show common fallacies.

On a number of occasions, I have warned clients about major security problems only to have them decide that security was not as important as getting that next release out or
2.2 The Seven Most Deadly Sins

making nonsecurity computer improvements. Later, they learn the sad reality—recovering from a security breach commonly costs ten times as much as having implemented security before the break-in would have cost—and they then must spend the money to implement the security. This “ten times the cost” figure represents only the direct cost. It does not account for lost market opportunities, for delayed products, for loss of customers who heard about the security breach and went elsewhere, and for costs to customers and employees who simply could not access your site and e-mail during recovery. It does not account for lost investors or for other consequences of bad publicity, and it most certainly does not account for the damage done to an IT professional’s career.

What can be done to resolve insufficient resources and misplaced priorities? Spend an hour or two a week working on security as a skunk works project. Demonstrate a Linux firewall, Web server, or VPN. Show how easy it is to update Linux software when patches come in, to use SSH and GPG, to crack most passwords, or attack a Wi-Fi wireless network. Do scans of your network from your home system (using nmap with the -O flag) to show how open your network is. Install Snort and Portsentry outside of your firewall (if any) to show how much your network is attacked.

Talk with your colleagues to find accounts of problems and relay them to your management. Have a good consultant, or other trusted outside source, do a security audit of your company and recommend improvements. Never give up. Never surrender. Giving up leads to procrastination and results in compromised systems. That is the dark side of The Force. Finger pointing is a popular game after a major problem.

Misplaced priorities also can mean using Microsoft because “We are a Microsoft shop,” disregarding that it may not have sufficient security for servers accessible from the Internet.

---

A Microsoft Corp. Vice President recently told a U.S. federal court that sharing information on vulnerabilities with competitors could damage national security and even threaten the U.S. war effort in Afghanistan. He later acknowledged that some Microsoft code was so flawed it could not be disclosed safely.

The amazing statements and candid admissions were some of Jim Allchin’s testimony during his two days in court before Judge Kollar-Kotelly, who is hearing the case in which nine states and the District of Columbia are seeking stricter penalties for Microsoft’s illegal antitrust behavior.

---

8. A skunk works project is one completed in secret without management approval or knowledge. Sometimes it is done as a pilot project to demonstrate the feasibility of something that the engineers think is very important but which management will not fund. After demonstrating a working prototype, often funding can be obtained. I have launched a number of important projects this way.

The term “skunk works” comes from Lockheed’s elite design group, who in complete secrecy designed the U2, the SR-71, and other highly classified aircraft. This elite design group was headed for many years by the legendary Kelley Johnson.

9. Thanks “Galaxy Quest.”
2.2.6 Stale and Unnecessary Accounts (#6)

As discussed before, each account is a possible entry point into the system. Imagine, for a moment, that you realize your system has been compromised and you have to send a message to everyone to change their passwords immediately.

When a user no longer will be using the system, be sure to remove her account from the system.
2.2 The Seven Most Deadly Sins

A stale account’s password will not be changed, thereby leaving a hole. If they have data that needs to be reassigned, disable their account by putting a “*” or “!!” in the ex-user’s password field in the /etc/passwd file. This disables logging in via that account because no password encrypts into either of these values and shadow password-enabled code understands these sequences. Get things cleaned up as soon as possible. Make sure that no set-UID or set-GID programs or publicly readable or writable files containing confidential data remain owned by that account.

Issuing

```bash
chmod 0 /home/someone
find / -user someone -ls
```

is a good start. Note that the user may have a mailbox, entries on mailing lists, files in the print spool directory, accounts in various applications, etc. that will need to be attended to.

Note that some of the services you removed (while correcting an earlier sin) have accounts in the /etc/passwd file. When you remove that service, make sure that the /etc/passwd account is also removed or disabled. Some of the notables are FTP, NFS, uucp, mail, gopher, and news. If you do not need them, get rid of them.

2.2.7 Procrastination (#7)

In many reports of intrusions the SysAdmin says, “I meant to install...TCP Wrappers...a newer version of...a firewall...meant to turn off NFS...” Clearly they knew, at least vaguely, what had to be done but delayed until it was too late. Sure, you have more responsibilities than time, but consider setting aside an hour twice a week to upgrade security.

This book is organized so that it may be used as a workbook, with the most urgent and fundamental problems discussed first. This allows you to pop a book marker in it and advance the marker every few days.