Aspects of Computer Security: A Primer

Steve Langer and Brent Stewart

As health care organizations continue on the path toward total digital operations, a topic often raised but not clearly understood is that of computer security. The reason for this is simply the vastness of the topic. Computers and networks are complex, and each service offered is a potential security hole. This article describes for the lay person the fundamental points of computer operation, how these can be points attacked, and how these attacks can be foiled—or at least detected. In addition, a taxonomy that should aid system administrators to evaluate and strengthen their systems is described.
Copyright © 1999 by W.B. Saunders Company

KEY WORDS: computer security, encryption, RSA, public key, Satan, Tripwire, Virtual Private Networks.

AS HEALTH CARE continues to embrace digital imaging, archiving, and telemedicine, a topic that is often raised but not fully understood is the security of patient-related information that is sent between referring physicians and specialists at large medical centers. The need for security will only increase, however, especially as tertiary care facilities continue to build large referring networks of clinics around themselves. With increased use of telemedicine consultation, secure communications across the Internet will become essential. Fortunately, computer security issues already have been studied by others (ie, the military, financial markets, and Internet-based retailers) so that a host of solutions are available. Thus, the issue becomes not whether it is possible to build secure computer networks, but rather how to build them from the multitude of possibilities. Numerous articles in the medical literature address parts of this question, but a full study of the field encompasses computer engineering, system administration, cryptography, and communication journals.[1,2]

When people think about computer security, the risks that probably come to mind are either some damaging agent (ie, a virus) or third-party eavesdropping on digital conversations. These are certainly valid concerns. However, to computer scientists there are several others. In fact, there are at least five distinct areas of concern.[3] These are:

Authentication: Are the agents (including user and programs) really who they claim to be?

Authorization: Does the agent have rights to the specific computer, file, or service?

Integrity: Are data sent by an agent intact or were they corrupted by a "man in the middle?"

Privacy: Are the data sent from agent to agent unreadable to third parties?

Nonrepudiation: Can an agent send data and then later deny having sent it?

The issues raised by these concerns cover hardware, networks, computer file systems, operating systems, human behavior, and advanced mathematics. To cover them all in detail would require months of research, but the basic goals and methods can be sketched in a reasonable survey. The goals of this paper are to:

- Examine the basics of local and network data representations
- Examine how operating, file, and network systems can be compromised, and rank the risks
- Look at the toois and methods available to a potential computer "cracker"
- Survey the defensive measures that are available in response
- Reexamine a system's vulnerability with the defenses in place
- Provide a useful reference list for further study (a glossary appears at the end of this article)

We start with the basics of computers, their operating, file, and networking systems—and how users interact with them.

COMPUTER BASICS AND VULNERABILITIES

One of the most basic goals of computer security is to protect the files contained on a mass storage device within the computer, or that data in transmission. These files can be simple data or working programs. The user interacts with these files via the computer's operating system (OS), and in turn the OS interacts with the files via the file system (FS). The most common desktop OSs in use today are the Mac OS (Apple, Cupertino, CA) and Windows

From the Department of Radiology, University of Washington Medical Center, Seattle, WA.

Address reprint requests to Steve Langer, PhD, Department of Radiology, University of Washington Medical Center, Box 357115, 1959 NE Pacific St, Seattle, WA 98195.

0897-1889/99/1203-0003$10.00/0

95/98 (Microsoft, Redmond, WA). These OSs are relatively primitive in that they are not multi-user nor do their FSs offer much in terms of default security (there are exceptions, but few users ever exercise them). As an example, consider the file listing on a Windows 95 machine.

Listing 1

```
My-file.doc    32 KB    MS-Word    1/11/1999 11:23 PM
```

A Windows 95 file listing showing file name, size, file type, and creation/modify date. Notice that this listing does not show an owner, or owner-based access restrictions.

Note that there is no way from the data shown in Listing 1 to determine who owns the file. In contrast, more sophisticated OSs offer this information. For example, the two most common advanced OSs are Microsoft Windows NT (currently at version 4.0) and UNIX. There are many variants of UNIX, but they share much in common, such as true multi-user, true multi-tasking, strict FS permissions, and a multitude of network services. Windows NT 4.0 is not truly multi-user out of the box (third-party products can alter this), but it does require user authentication and has a secure FS. Contrary to the FS of a primitive OS, UNIX and NT files carry much more information. Of primary importance is the concept of the file's owner, as well as the group to which the file belongs. A specific UNIX example is seen below:

Listing 2

```
-rwxr-xr-x    1 sgl user    360448 Mar 28    1997 pgp
```

A UNIX file entry. The first pattern (starting with a hyphen) shows that the file owner has rwx (read-write-execute) permissions, whereas the group and all other users have only r-x permissions. The "1" is a piece of data specific to this UNIX. The file owner is "sgl," who belongs to group "user." The file size is 360,448 bytes. It was created on March 28, 1997, and its name is "pgp." NT files carry even more information in the form of delete permissions and allow multiple groups assigned to the file.

Thus far, we have considered only data files within the computer. However, in the modern computing environment much work is only possible across the network. For example, e-mail, PACS applications, file servers, and electronic medical records all require the user's client computer to access a remote server. Furthermore, these remote servers usually require the user to login via the client application. How this is done can have a massive impact on system security. For instance, UNIX systems have a program called "telnet," which permits users to login to remote UNIX machines that run a telnet server (called "telnetd"). On older systems, the telnet client-server package sent login names and passwords in clear text. This means that anyone with a network sniffer could learn all the accounts and passwords on the server. Thankfully, newer implementations send the password encrypted.

While there are exceptions, Macs and Windows 95/98 machines usually have only client software on them such as web browsers, mail clients, and disk-sharing tools. In recent years, both Apple and Microsoft have learned the value of transmitting passwords in an encrypted form. Yet, the bulk transmitted data is in clear text. Meanwhile, NT/UNIX servers, in addition to the client software seen on the primitive OSs, usually offer a variety of services including web servers, e-mail, database servers, file sharing, and perhaps DICOM image archiving. Because of their server role, these machines and their server software must always be available.

To understand how a single computer can sort out requests for different services (and to understand the defensive strategies that will be described later), it is helpful to have a brief look at how the TCP/IP (Transmission Control Protocol/Internet Protocol) network standard is implemented. The standard provides the backbone services on which the Internet is built, and is analogous to an apartment building with an address and numerous apartments. The mail is first delivered to the building, then it is sorted to the various rental units based on apartment number. Similarly, a computer has an IP address of the form aaa.bbb.ccc.ddd, and within the computer there are thousands of possible "ports," each of whom may be assigned to a server application. The most common services are mapped to standardized ports so authors of client software can connect to them easily. The following listing shows the mapping of the most popular services.

Listing 3

```
# Network services, Internet style
#
echo        7/tcp
daytime     13/tcp
netstat     15/tcp
qotd        17/tcp      quote
msp         18/tcp      # message send
                          protocol
```

Listing 3 (Cont'd)

```
ftp              21/tcp
# 22 - unassigned
telnet           23/tcp
# 24 - private
smtp             25/tcp        mail
bootps           67/tcp        # BOOTP server
gopher           70/tcp        # Internet Gopher
finger           79/tcp
www              80/tcp        http # World-
                                 WideWeb HTTP
link             87/tcp        ttylink
kerberos         88/tcp        krb5 # Kerberos v5
ntp              123/udp       # Network Time
                                 Protocol
netbios-ns       137/tcp       # NETBIOS Name
                                 Service
netbios-dgm      138/tcp       # NETBIOS Datagram
                                 Service
netbios-ssn      139/tcp       # NETBIOS session
                                 service
imap2            143/tcp       # Interim Mail Access
                                 Proto v2
snmp             161/udp       # Simple Net Mgmt
                                 Proto
```

A subset of the standard port mappings for some common services. Obviously not all systems implement all these services, but all are possible. This is the /etc/services file from a UNIX system; the first entry (eg, echo) is the service name, and the second entry (7/tcp) defines the port and protocol used.

To see which services actually are implemented on a UNIX system, one can look at the "/etc/inetd.conf" file as in Listing 4. The first entry (ie, "time") shows what the service is named. The terms "stream," "tcp," "nowait," and "root" describe the type of TCP socket the service uses, how the process is launched, and who owns the process. The final group of words (ie, "internal") shows what program actually is associated with the service. For example, an "internal" service is part of the core OS while the string "/usr/sbin/tcpd in.telnetd" shows that the telnet service is started by the TCP super-server (tcpd), which in turn calls the program "in.telnetd." The lines that begin with "#" are commented out, and the service is not loaded. (Similar services are available on NT servers, but the port mappings are encoded in the NT registry, and the list of active services are found via the Control_Panel/Services tool.)

Listing 4

```
# See "man 8 inetd" for more information.
#
# ⟨service_name⟩ ⟨sock_type⟩ ⟨proto⟩ ⟨flags⟩ ⟨user⟩ ⟨server-
  _path⟩ ⟨args⟩
#
time   stream tcp   nowait root   internal
ftp   stream tcp   nowait root   /usr/sbin/tcpd wu.ftpd
telnet   stream tcp   nowait root   /usr/sbin/tcpd in.telnetd
shell   stream tcp   nowait root   /usr/sbin/tcpd in.rshd -L
login   stream tcp   nowait root   /usr/sbin/tcpd in.rlogind
# exec   stream tcp   nowait root   /usr/sbin/tcpd in.rexecd
# talk   dgram   udp   wait   root   /usr/sbin/tcpd in.talkd
#ntalk   dgram   udp   wait   root   /usr/sbin/tcpd in.talkd
#
# Kerberos authenticated services
# klogin   stream tcp   nowait root   /usr/sbin/tcpd rlogind -k
#
#finger stream tcp   nowait nobody   /usr/sbin/tcpd in.
  fingerd -w
#systat stream tcp   nowait nobody   /usr/sbin/tcpd /bin/ps
  -auwwx
#netstat   stream tcp   nowait root   /usr/sbin/tcpd /bin/netstat
  -a
#
# Ident service is used for net authentication
auth   stream tcp   nowait root   /usr/sbin/in.identd   in.identd
#
# These are to start Samba, an smb server that can export file-
  systems to
# Pathworks, Lanmanager for DOS, Windows for Workgroups,
  Windows95, Lanmanager
#
#netbios-ssn   stream tcp   nowait root   /usr/local/samba/bin/
  smbd smbd
#netbios-ns   dgram   udp   wait   root   /usr/local/samba/bin/
  nmbd nmbd
# End of inetd.conf.
```

A partial listing of services on a UNIX system as displayed in the /etc/inetd.conf file. The uncommented lines (those that do not start with "#") actually start the service at boot time.

Having described the basics of file representation and network services, we can consider two modes of operation for a given computer: standalone console mode (no network connection) and networked modes. (Obviously, the networked mode inherits all the vulnerabilities of the console mode, but has the added risks associated with transmitting data across the network.) We assume that a "cracker" is attempting to break into the system. We shall also rank each security component for each scenario on a scale of 1 to 5, with 1 being the poorest and 5 the best. The scale is subjective, but scaled relative to the poorest and best methods that are available. For instance, an OS that does not require any user authentication would score a 1, whereas a non-networked computer in a guarded room that can be entered only by providing a name, password, and a fresh DNA sample would tend to score near a 5 on the authentication scale.

Console Operation (Not Networked)

We shall use the term *console* to imply that the cracker has sat down at a keyboard and monitor that are attached directly to the computer (Fig 1). No network connection is postulated at this point. In this scenario, the only way for a cracker to compromise the files on the computer is to gain access to the console and log in. Physically locking the console behind a closed door would offer maximum security—at the expense of making the computer difficult to use. Short of this, the options are explored below.

Primitive OS (Mac and Windows 95/98)

- Default Behavior

  For these systems there is no user login, so anyone can get on the system. (Windows will query for a network login, but this will not prevent the cracker from accessing local files.)

- Vulnerabilities

  Because there is no login (by default), authentication is nonexistent. Furthermore, once on the console, the cracker has access to all files. He/she may plant viruses, Trojan horses (malicious programs masquerading as something else), or simply delete/modify anything. Because there is no telling who was on the computer, it is also possible that a cracker could submit e-mail in the rightful user's name (assuming that a nonsecure mail client is used).

Fig 1. In this instance, the "cracker" has direct physical access to the console that is attached to the computer.

- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
1	1	1	1	1

Advanced OS (NT/UNIX)

- Default Behavior

  All users must login by providing both an account name and a password. Accounts are created by the computer administrator, hence only trusted accounts should be on the computer. Once logged on, a user will have full access to his/her files, but no access to others unless the owners of those files have set suitable group and world permissions.

- Vulnerabilities

  If crackers can somehow guess the password of user X, they will have access to any files that X does. In addition, they may be able to secure a copy of the password file and submit it to a password cracker. If the cracker succeeds in getting the administrative password, the entire system lies open.

  Another method that the NT/UNIX cracker may use is to boot the computer into single-user mode, in which case the cracker becomes the administrator. If this occurs, the cracker can simply redefine the administrative password without ever knowing what the original one was.

  Given basic authentication, as well as the concept of file owners and rights, the ranking for the Advanced OS group comes up a bit from the Primitive group. The fact that file owners can choose to hide their files from others also boosts the privacy ranking. Integrity is improved because a user must have write privileges on a given directory/file in order to add to or modify them.

- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
3	3	2	2	2

Networked Mode

In this instance, the cracker is on a remote keyboard and terminal (perhaps another computer) and is attempting to somehow molest the target machine across the network (Fig 2). As one would expect, because primitive OSs offer fewer services, they have fewer openings for this kind of attack. For instance, Macs and Windows 95/98 usually are equipped to allow only disk and printer sharing (they can be configured to offer Worldwide Web and other services, but this is relatively rare). In contrast, NT/UNIX systems offer many services: disk sharing, web servers, database servers, telnet, e-mail, etc.

Primitive OS (Mac and Windows 95/98)

- Default Behavior
  Some Mac/Windows client programs may send clear text login passwords to the server, and virtually all other data are transmitted in clear text unless the application itself encrypts it (as do many web browsers). In contrast to the lack of a local login, both Macs and Windows 95/98 machines can be made to require a login and password before allowing network agents to gain access to their local files. The passwords for this procedure are transmitted in encrypted form.

- Vulnerabilities
  The cracker may sniff the network for clear text passwords and data. He also may use "man in the middle" attacks to substitute Trojan horses (ie, a virus) in place of a valid file that the user requested. Finally, the cracker may assault the computer with various Denial of Service Attack (DSA) attacks. In addition, even though the passwords for file sharing are sent across the network in an encrypted format, a wily cracker could still copy it and write a modified login program that would just send the encrypted password to the server. In sum, the situation is arguably worse than the console scenario.

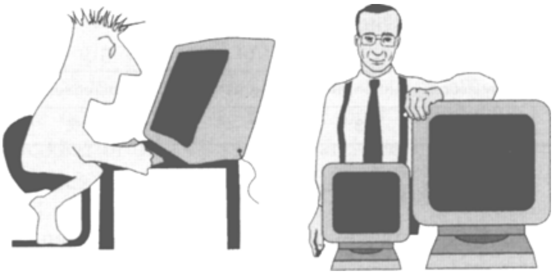

Fig 2. The cracker attempts to harass the server remotely, via the network.

- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
1	1	1	1	1

Advanced OS (NT/UNIX)

- Default Behavior
  Older UNIX systems send passwords as clear text, and currently both NT and UNIX send the bulk of data as clear text. Furthermore, NT and UNIX often are shipped with many services enabled that are unnecessary and offer an avenue of attack (eg, the finger service, which can be used to identify users on UNIX systems).

- Vulnerabilities
  Basically, the cracker will try the same gambits as for the primitive case: sniffing, "man in the middle" substitutions, and DSA attacks. However, the attacks will likely be more persistent because the cracker will get a bigger thrill by taking down a server rather than a simple desktop computer. As in the primitive OS case, a cracker could sniff encrypted passwords and with a modified login program still gain access to the server.

  Because clear text passwords and data could be sniffed from the network, the authentication and privacy rankings would be diminished from the console example. With authentication weakened, nonrepudiation is weakened as well.

- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
2	3	2	1	1

CRACKER STRATEGIES

In the previous section we described the default behavior of basic OSs and how a cracker might attempt to compromise those systems. In this section we provide the details behind the previously mentioned attacks. Such strategies can be ranked in three basic categories based on intent: gaining illicit access to files or machine resources, substituting programs or data with malicious intent (eg, Trojan horses and other viruses), or causing a computer to fail by exploiting bugs in the target machine's software (the so-called DSA attack). (Although crackers are of both genders, for simplicity the following discussion assumes a female.)

• Gaining Illicit Access

Sniffing

Historically, TCP networks have been shared. That is to say that, on a given network subnet (eg, one with 2 or 200 computers), one cable carried the same data to all computers. On a trusted network, this is not a problem because a given computer will only "listen" for TCP packets that are addressed to it, and then route those packets to the proper port. However, many networking cards can be put into what is known as "promiscuous" mode and will then listen for all packets on the subnet. If a cracker sifts through all that data (or a program does it automatically), it is trivial to learn the login names and passwords for the machines on that subnet. One of the more common such programs is called "sniffit."

Cracking

On both NT and UNIX systems, there are files that contain the names and passwords of all user accounts (NT's "registry" and "/etc/passed" on UNIX). If a cracker can get access to this file, she can use any number of "dictionary attack" programs to decrypt the password file on her own computer. Given enough time, simple passwords composed from ordinary words and names, and a large enough dictionary to check against, such gambits ultimately pay off. Two of the more popular programs are "Crack" for UNIX and "NTCrack" for NT. Typically, a cracker will acquire the password file by entering the system via an insecure login (NT's guest account or a poorly protected "anonymous ftp" account in UNIX).

Single-User Reboot

Both NT and UNIX permit a computer to be booted into a single-user "maintenance" mode. Typically, one must be at the system console to achieve this state, but once there a cracker could redefine the administrative password to be anything, thus taking over the computer. In a strict sense, the computer has not been "cracked" because the original passwords were never learned, but that may be small comfort.

• Substituting Bogus Data

The goal of the next two strategies is to somehow place a malicious file on the system without the users of that system expecting it. Generally, all such files are known as Trojan horses and they may be as simple as a substitution for the telnetd program (which could allow the cracker to get into the system through a back door), or they could be the most destructive virus imaginable. Possibly worse from a radiologist's perspective, the cracker could simply substitute a false image for a patient's x-ray examination.

Direct File System Attack

In this scenario, the cracker has somehow achieved direct access to the files on disk and replaced them with her own. These "Trojans" may be on a specific client computer or on a server from which many clients will be "infected."

IP Spoofing

IP spoofing is the name given to the strategy wherein a cracker impersonates a "real" server and generates TCP/IP packets that have false originating IP addresses. This permits several ploys, including the ability of the cracker to "place" her machine between a client and the intended server. By acting as the fake server, the cracker becomes the "man in the middle" (Fig 3). This allows her to receive requests from the client and study them. She may decide to pass the data through unmolested to the real server and see what the response will be, then alter it on the return leg, or she may be satisfied to simply observe the data.

• Denial of Service (DSA) Attacks

In systems as complex as NT/UNIX servers, there are always bugs. If they happen to be in a network service, it is entirely possible for a remote

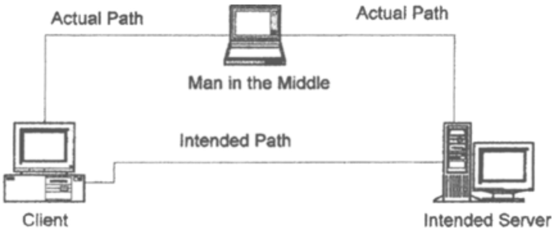

Fig 3. The "man in the middle" attack. The middle computer assumes the IP address of the real server to observe (and possibly alter) data flowing between the client and the real server.

cracker to instigate the bug into shutting down the targeted service or (rarely) even the entire computer. The goal of the cracker here is not to steal data or even to break into the system, but rather to simply bring the computer down. Following are some well known ploys.

Ping of Death

This attack is the result of an interesting union of two different bugs wherein the first was not widely discovered until the second occurred. First, the majority of machines that run TCP services inherited the underlying network software (called Berkeley Sockets) from programs that were written by the University of California Berkeley. Not just computers, but routers, switches, printers, firewalls—nearly all TCP devices contain software that can be traced back to that origin. One of these programs (the server side for the diagnostic tool "ping") did not check that incoming packets would "fit" into the storage space allocated for them. If an incoming packet overran the storage space, the error could bring down the TCP service and, on primitive OSs, the entire device. However, this bug was never triggered because the authors of the ping client knew what the maximum packet length was and never exceeded it.

Then came Microsoft Windows 95. Its version of ping had a bug that miscalculated the length of its packet, so all a cracker had to do was type "ping-1 65527 victim.org" and the machine "victim.org" would crash.[4]

WinNuke

There are several versions of this program, whose purpose is to send a specific string of characters to Windows 95 and NT 4.0 machines. The target on the victim is port 139 (one of the NetBIOS ports from Listing 3). Once hit, the Windows network driver fails completely and the machine goes off of the network. One can imagine that this would be devastating to a business providing Internet services using NT servers.[4]

Smurf Attack

Named for the prodigious reproductive capability of the cartoon characters, a smurf attack creates an amplified avalanche of data packets against the target computer.[5] The program "smurf.c" sets the originating IP address of a ping packet to be the same as the victim computer's address. It then sends the ping packet to the broadcast address of other computers on the target's subnet. When the other computers reply, the victim machine experiences an avalanche—and the entire subnet becomes saturated (particularly if it is a shared subnet). (The cure for this attack is to disable network broadcasting on the subnet.)

Syn DSA Attack

"Syn" is short for "synchronous" and is related to the way this attack brings down TCP/IP network services. Normally when a data packet reaches a networking card, it issues a request and the receiver responds to the transmitter (based on the packet's originating IP address) that it is ready for more data. The receiver then awaits additional data from the transmitter. If the transmitter does not reply, the buffer on the receiver becomes "stuck." If enough packets arrive, all from seemingly different IP origins, the receiving computer uses up all its networking buffers and ceases to function.

The theory for this attack was first described by Robert Morris in 1985, and the example code was later published in the magazine 2600.[6] A key requirement of the attack is the ability of the attacker to construct bogus packets that contain false originating IP addresses (as in the IP Spoofing attack). In September 1996, Public Access Networks Corporation of New York City lost its web and mail service for several days to this ploy.

DEFENSIVE STRATEGIES

Not surprisingly, defensive strategies are aimed at thwarting the attacks described in the previous section, although some additional benefits (such as nonrepudiation) come in the bargain. It should be mentioned that most of the discussion here focuses on protecting UNIX and NT servers because most crackers aim their attacks at the server.

The first category of "restricting access" is an attempt to keep intruders out of a network subnet that is believed secure (and it is assumed that only trusted computers are on it). Techniques detailed in the "Encryption" section, particularly the Virtual Private Network (VPN) methods, are useful when communication must go across unsecured networks, such as the Internet. Finally, "Logging Tools" are useful diagnostics to assure that a specific computer has not been compromised.

• Restricting Access

Switched Networks

Recall the sniffing strategy listed under Cracker Strategies. That attack worked because the network data was shared among all the computers in a subnet. In faster local area networks (LANs) such as 100 megabyte/s (as opposed to the older 10 Mega Bit/s), it has been found expedient to use a switching topology that actually creates brief direct connections between any two communicating computers. A side benefit of this is that a sniffer can only see data that was actually intended for the local computer's IP address. This benefits all machines on the subnet.

Firewall

The purpose of a firewall (Fig 4) is to put a virtual wall in front of the local network and isolate it from the "outside" Internet. This is done by putting two network cards into the same computer: one connected to the outside world, the other to the internal network. A set of rules determines what traffic will be allowed to cross the bridge and in what direction.

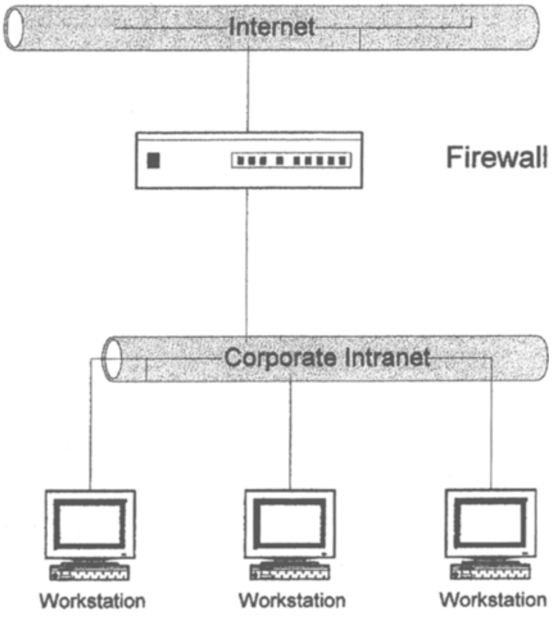

Fig 4. A firewall. One network card is connected to the outside Internet, the other to the internal network. Rules determine what traffic is passed.

Restrictions can be made as tight as desired by configuring the firewall to pass or halt traffic based on the originating IP address of the sender, and the type of service requested. For example, the administrator of a network may wish to forbid anyone outside the local subnet from telnetting into the departmental server. Since telnetd is on port 23 (Listing 3), the administrator could simply put the following line in the firewall's configuration tables[4]:

Listing 5

```
# by default we'll accept most traffic
/sbin/ipfwadm -I -p accept
#but we'll block outside telnets
/sbin/ipfwadm _I -a deny -P -S0.0.0.0 -Dmy_server/32 23 -o
```

Setup for a firewall to forbid outside telnet requests to "my_server" while passing all others.

In a similar manner, the WinNuke attack discussed in the last section could be prevented by blocking traffic to port 139. Since the firewall can be any type of computer and protects an entire network, a firewall benefits all computers behind it.

TCP Wrappers

TCP_Wrappers function exactly like a firewall except that, instead of a separate computer, the required software is installed directly on the computer to be protected and "wraps" the service to be protected.[7] As noted in Listing 4, the TCP superserver tcpd is responsible for actually launching most of the TCP services on UNIX computers. By installing TCP_Wrappers, an additional check is placed between tcpd and the associated service (eg, in.telnetd). When tcpd is about to launch an instance of in.telnetd, a rules file is checked in the same manner that it was for the firewall. Only if the service is allowed will the request be honored. The disadvantage of TCP_Wrappers is that the package must be installed on every computer to be protected.

SecureID

One of the more advanced hardware-based user authentication methods is the SecureID card (Security Dynamics Technologies, Bedford, MA). Every valid user on the host computer carries a unique SecureID card that generates a new six-digit random number every 60 seconds. The user must login with an account name, password, and the random number within the 60 seconds that the number is valid. Only the card, user, and host know what the number is and, once used, the number cannot be reused (in case a fast-sniffing cracker tried it within the time limit).

Other Physical Authenticators

More elaborate physical methods of access authentication are available including biometric methods such as fingerprint or retinal scanning, and voice recognition. In the near future, these methods may be combined with smart cards (credit card–sized devices with embedded electronics). Such smart cards may contain a compressed image of the authorized user's thumbprint. The user would then slide the smart card through a computer-based reader, enter his name, password, and present his thumb for scanning. Only after all those tests are passed would the smart card authenticate the user to the computer network.[8]

ENCRYPTION THEORY

The goal of encryption is to perform some mathematical transformation to a "clear text" and produce a "cipher" text that cannot be read. The transformation may be irreversible (as in the case of hashing operations performed on passwords) or reversible (as when one wishes to have a decipherable message). One may ask what the use is of a message that cannot be decoded. Consider the passwords for user accounts on a computer. One would not wish to store them in a form that could be read by crackers, so they are stored in an encrypted format. Now when a user logs in and enters her password, the transformation is applied to it and the result compared with the stored version. If the two match, the user is authenticated. This scenario works as long as the transformation produces a unique output for every given input, and in fact a great deal of research has been done to develop such algorithms.[9]

In contrast to the irreversible hashing function just described, the methods listed below make use of reversible algorithms that can return the original clear text. The methods described all rely on advanced mathematics and algorithms that have been derived over years of research. Some excellent review texts are available that go into greater detail than we shall do here.[10,11] However, one point that will become clear is that encryption (and "public key" in particular) aids all security efforts with the exception of authorization.

In the language of cryptography, a "key" is a string of letters or numbers used to transform the clear text. The "strength" of encryption is partly related to the number of bits used to encode the keys, and a variety of algorithms are available that are more or less resistant to attack. However, because of the different underlying algorithms, one cannot simply claim that a 56-bit key from one method is weaker than a 64-bit key from another. It is only correct to say that, within a given algorithm, the more bits a key has, the stronger the encryption. In fact, some encryption routines are so good that the United States government discourages their domestic use and makes it illegal to export them.

Private Key Methods

In private key encryption, a single private number or string of characters is used to both encrypt and decrypt the message, as in Fig 5. Therefore, the key must be shared among all those who wish to communicate with each other. The Data Encryption Standard (DES) is a widely used method that was judged so difficult to break by the US government that it was restricted for export to other countries. Because DES applies a 56-bit key to each 64-bit block of data, there are more than 72 quadrillion encryption keys that can be used. For each message, the key is chosen at random from among all possible keys.

DES originated at IBM in 1977 and was adopted by the US Department of Defense (IBM, Westchester, NY). It is specified in the American National Standards Institute X3.92 and X3.106 standards. However, while the US government has made it illegal to export the software from the United States, the algorithm has been reimplemented and free versions are widely available on web sites outside the country. Formerly, concern by law enforcement that the algorithm was too secure cast doubt as to whether DES would be recertified by the National Institute of Standards and Technology. However, by 1997, a challenge put forth by a commercial firm resulted in a DES cracking machine (RSA Data Security, San Mateo, CA).

The International Data Encryption Algorithm (IDEA) was developed by Xuejia Lai and James Massey of the Swiss Federal Institute of Technology to efficiently encrypt/decrypt large amounts of

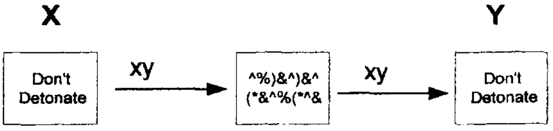

Fig 5. A private key encryption-decryption cycle. X encodes the message with the private key xy and transmits it to Y. Y then decodes the message with the same key.

data quickly. IDEA uses a block size of 64 bits with a 128-bit key that is generally considered sufficiently strong against cryptoanalysis. The use of a cipher feedback mode of operation strengthens the algorithm further. In this mode, cipher text is used as input into the encryption algorithm.[12] Statistical analysis of the characters in the cipher text gives no clues to the contents of the plain text because IDEA spreads out a single plain-text bit over many cipher text bits.[13]

Public Key Methods

A more sophisticated approach, public key methods rely on an asymmetric application of two different keys, as shown in Fig 6. For example, agent Y produces two keys (y and y′). The private key (y) is known only to Y and is never transmitted across a network. In contrast, the public key (y′) is freely sent to all of Y's associates—perhaps even posted on Y's Worldwide Web page. When X wishes to communicate privately with Y, he/she encrypts the message with Y's *public* key and then sends the message across an insecure network with the knowledge that only Y has the private decoding key.

There exists the possibility that an impostor (M) may post a public key, claiming that it belongs to X. The only way for Y to know for certain whether the x′ is valid is if X personally gives it to Y or a trusted third party is available to certify the authenticity of the key.

In addition to privacy, public key systems can be used to authenticate each other by a method called "digital signing" (more on that below). Two of the more well-known implementations are RSA and PGP, detailed below:

- RSA
  In 1977, Ron Rivest, Adi Shamir, and Leonard Adleman developed RSA (from the authors' last initials). The RSA algorithm is the most common embedded encryption and authentication algorithm and is licensed by most web browsers including those from Netscape (Netscape Communications Corp, Mountain View, CA) and Microsoft. The algorithm patent is owned by RSA Data Security (RSA Data Security, San Mateo, CA). The RSA algorithm multiplies two large prime numbers and through additional operations derives a public key and a private key. Once the keys have been developed, the original prime numbers are no longer important and can be discarded. Several public key algorithms are patented by RSA, including RC4 and RC5.

  The strength of the encryption varies according to the length of the keys. Currently, RSA Laboratories recommends key sizes of 768 bits for personal use, 1,024 bits for corporate use, and 2,048 bits for extremely valuable keys like the root-key pair used by a certifying authority (RSA documentation).

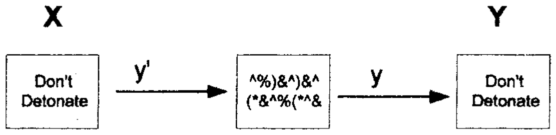

Fig 6. A public key encryption-decryption cycle. X encodes the message with Y's public key y′. The message is then sent to Y, who decodes the message with her private key y. Not even X can decode the message once it has been encoded with y′.

- PGP
  RSA is very strong, but has a few drawbacks. Once X and Y have authenticated each other, they could continue to converse using RSA encryption. The problem is that RSA is extremely slow compared with other algorithms that are nearly as secure. A more efficient way to continue the conversation is to switch to another encryption scheme once authentication has occurred. Pretty Good Privacy (PGP) is a hybrid cryptosystem that uses several different encryption standards. When ultimate security is needed, PGP uses the public/private key. For medium security with greater performance, PGP switches to IDEA because IDEA is very efficient for large amounts of data.[14]

Digital Signatures

As alluded to previously, public key algorithms permit not only secure communication but also authentication among users. This is done by digitally signing a message. As in the example above, X may encrypt a message for Y by encoding it with y′ (Y's public key). However, X also could sign the document with x (her private key). Upon receiving the note, Y would then check its authenticity by using x′ to mate with x. Because only X has x, Y would know the note was genuine.

Digital signing has the ability to completely defeat "man in the middle" attacks. The steps in doing so are as follows (for simplicity, we did not encrypt the actual message in this example, although that certainly would be possible):

- X prepares a document.
- X performs a mathematical function on the document that creates a unique value called the "message digest" (MD).
- X digitally signs the MD with her private key, x, and sends both to Y.
- Y receives the document and computes the MD.
- If the MD calculated by Y does not match the MD signed by X, Y knows that the document has been altered.

All these steps must be executed to fully authenticate the document. Omission of any one of them opens the door to sabotage. Consider Fig 7.

Finally, one should consider how digital signing meets the goal of nonrepudiation. Since only X possesses x, any message whose digital signature matches x′ can be assumed to have come from X. There are only two ways that X can deny having sent a message: if there is no certifying authority to authenticate the public key and an imposter distributes a false one, or if her account has been compromised—an avenue that will be addressed below.

ENCRYPTION IN PRACTICE

In the discussion above, we considered the various methods by which encryption can be implemented, but we have not seen any practical demonstrations. The following discussions consider how

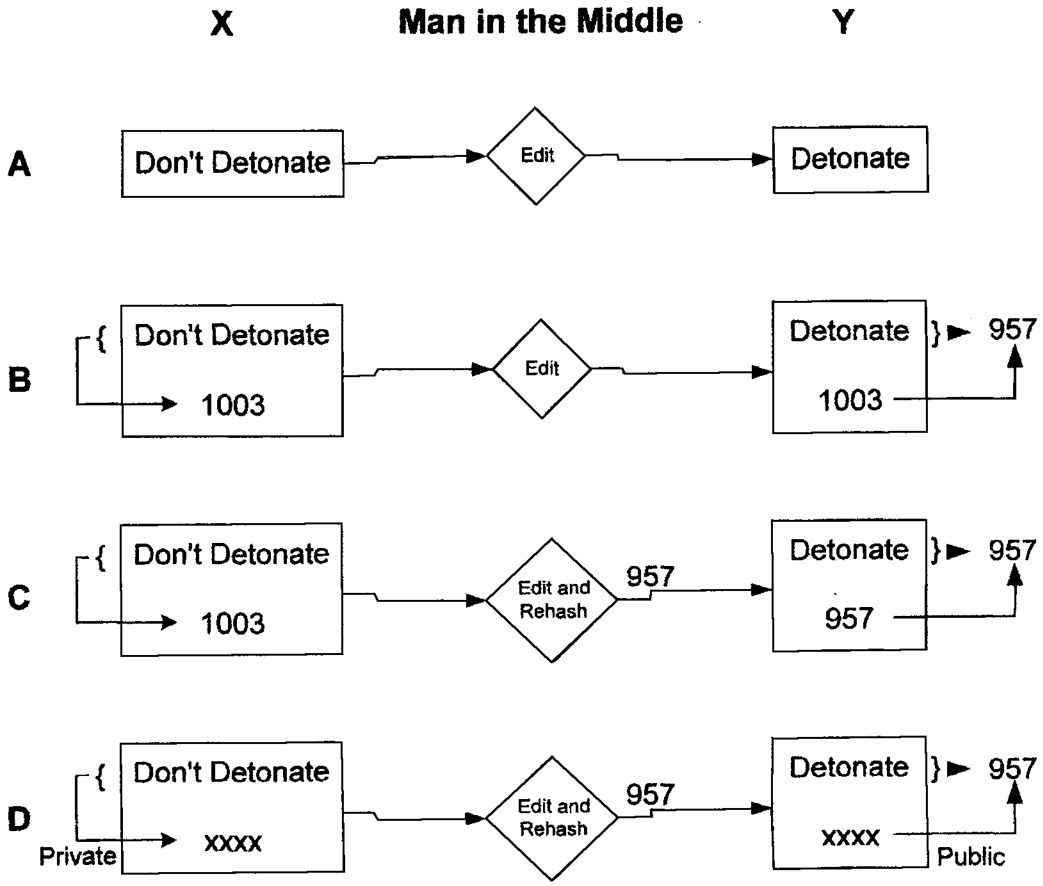

Fig 7. (A) X sends a message, and it is altered by M. Y cannot tell. (B) X computes the document's message digest (MD) and sends its value (1003) with the document. M alters the message. Y recomputes the MD of the message and detects an alteration because the MDs do not match (ie, 957 is not equal to 1,003). (C) This time, M recomputes the MD and sends it with the altered message. When Y recomputes the message MD and checks against the sent MD, the two match and Y is fooled. (D) X digitally signs the MD, and M cannot reproduce X's signature without X's private key. Nevertheless, M alters the message. When Y decodes the MD signed by X and compares it to the recomputed MD of the altered message, Y detects the substitution (ie, 957 is not equal to 1,003).

encryption may be applied to single files, entire disks, or networks.

Disk Encryption

If a cracker sits down at the keyboard of a primitive OS, the entire contents of the local disks lie open to them. Similarly, if the cracker succeeds in logging onto a NT/UNIX machine, he has access to all the files accessible to that user. If the administrative account is compromised, every file is at risk. To mitigate this risk, it is possible to encrypt the entire contents of a disk, thereby making the disk unreadable without a pass phrase. Replacement file systems are available for most UNIX versions that can implement encryption. For instance, a group at the University of California–Berkeley distributes a Cryptographic File System (CFS) for standard UNIX file systems that implements both DES and IDEA.[15] Without supplying a pass phrase (which can be up to 120 characters), the encrypted disk cannot be read, and if it is the system's boot disk, the entire computer becomes unbootable. Similar capabilities exist for Windows and Mac platforms. Those options will be discussed more fully in the next section.

With disk-wide encryption, systems may only be booted by people entrusted with the pass phrase. However, once the disk is unlocked and the computer in operation, any user on the system can see all files for which he or she has permission, which, in the case of a primitive OS, again means all files. The next section deals with this issue.

File-by-File Encryption

On a primitive OS's clear disk, or a decrypted cipher disk, any user can see any other user's files. On NT/UNIX, users can read any files for which they are the owner or group member (or if the file is marked *world readable*). If a cracker compromises the administrative account, he or she can read any file. The last line of defense is file-by-file encryption. One tool we have seen already can deal with this. PGP is not just an encryption algorithm—is available in the public domain for file encryption on UNIX, Windows, and Mac computers (Network Associates Inc, Santa Clara, CA). The PGP program maintains both a public key ring and a private key ring. To send secure e-mail to a colleague, one would encrypt the file with his or her public key and digitally sign with the private key. However, to protect files for personal use, one would encrypt the file using one's own public key—and decrypt later with one's private key. The challenge is to keep one's private keys in a secure place (such as a floppy disk) that would never be susceptible to mischief.

Although PGP can be pressed into service for personal file protection, it can be time-consuming to implement. A more user-transparent form of file encryption (for UNIX) is provided by the Transparent Cryptographic File System (TCFS). Written by the Informatics Department at the University of Italy–Salerno, the TCFS application is not just useful for local disks.[16] In NT/UNIX systems, it is common for one computer to mount the disks of another via Sun's Network File System (NFS) protocol. By default, a client may mount NFS disk partitions if it has a trusted IP address or Internet name. If a cracker impersonates a trusted IP address (recall IP spoofing), it is possible to access the files on the NFS server. In addition, the file data are sent across the network in clear text.

TCFS is installed under the standard UNIX file system and uses DES encryption for both file contents and file system data. During initial login, or when accessing new local or networked disks, the user must supply his TCFS login name (which can be the same or different as his system login). If the login matches his private key, file access is granted and the files are decrypted. Furthermore, all data (including passwords) are sent across the network in encrypted form. Although it is still possible for a cracker to mount a disk on a TCFS file server (via IP spoofing), without the pass phrase the cracker will never be able to read the files.

For Windows and Mac users, there is "For Your Eyes Only" (Symantec Corp, Cupertino, CA). It can use RC4, DES, and triple DES to provide both public and private key encryption. Furthermore, encryption can be performed on files, on directories, or for the entire disk. Different directories may be encrypted by different users with different algorithms and passwords. Microsoft has announced that the FS that will accompany NT version 5 will include disk-based public key encryption, but files will still be transmitted in clear text across networks.[17]

Virtual Private Networks

Virtual Private Networks (VPNs) use cyptographic data transmission over insecure networks to achieve a degree of secure communication. One of the most widespread methods, the Secure Socket Layer (SSL), is a Netscape modification to the Berkeley Sockets Library (recall the DSA discussion). The standard socket functions allow TCP programs to communicate with each other, but the communication is in clear text. With SSL, the standard Berkeley code is replaced by functions that establish an RSA (or alternate) public key link between the client and the server. After the original "handshake" in which public keys are exchanged, additional client/server communication occurs over the encrypted link—including any needed passwords. Both Netscape's and Microsoft's browsers have licensed RSA's RC4 algorithm (using 40-bit keys) to support SSL. In general, SSL can be used to provide secure communications across all TCP services. However, when used by just a web browser, the method is sometimes referred to as HTTPS (the Secure Hyper Text Transfer Protocol) in contrast to standard HTTP (the Hyper Text Transfer Protocol), which is used during clear-text web browsing.

A modification of the UNIX telnet-like remote shell, the Secure Shell ("ssh," also available for Windows 95/98/NT and Mac), applies encryption to the client/server network link. Normally, telnet utilizes TCP port 23 for its own data use. However, the authors of ssh gave it the ability to reroute traffic from other ports through its encrypted channel, making it possible to protect ftp, web, X Window displays, or any other desired traffic.[18] The exact encryption used is negotiated when the ssh client connects to the server, but can include public or private key methods.

Microsoft has developed the Point-to-Point Tunneling Protocol (PPTP), a secure variant of the standard TCP dial-up tool Point-To-Point Protocol (PPP), for users who must network across insecure phone lines. Unlike ssh, which can selectively encrypt TCP services, PPTP encrypts everything that flows through it (which can create bottlenecks). The RSA RC4 standard is used to create a 40-bit session key based on the client password, although in the United States and Canada a 128-bit version is available (based on a Microsoft white paper). This key is used to encrypt all data that are passed over the Internet, keeping the remote connection private and secure.

Secure Applications

Although a uniformly applied method of encryption probably provides the most robust security solution (ie, SSL or PPTP), various applications attempt to fulfill the need within their own environment. For example, we already have seen how browsers implement SSL within the HTTPS framework. In a similar manner, other specialized applications have come forward. For example, e-mail clients by numerous vendors license 40-bit RSA encryption algorithms for secure e-mail.

Kerberos is a widely used authentication and authorization service for UNIX, and Microsoft claims that NT 5 and Windows 95/98 ultimately will support it. In its most basic implementation, it uses private key encryption to authenticate users, acting as a firewall guarding the computers and other resources behind it. Developed by the Massachusetts Institute of Technology, a user must satisfy the Kerberos server before any further access is granted.[19] In fact, some implementations require the user to login to individual computers behind the Kerberos server, but most automatically forward the Kerberos authentication to avoid inconvenience. Similar to SSL, Kerberos establishes a secure link before any passwords are transmitted, but in addition Kerberos can maintain a detailed authorization database for users and computers—regulating which people or resources can communicate. Unlike SSL, however, applications have to be written to use Kerberos services, whereas SSL is a nearly transparent modification to established TCP standards.

The Challenge-Handshake Authentication Protocol (CHAP) and Password Authentication Protocol (PAP) methods are both used for authentication on PPP dial-up connections. Like PPTP, both systems avoid sending a clear-text password across insecure links (by sending only the hashed password equivalents). Unlike PPTP, though, after authentication neither system provides an encrypted channel.

Some versions of UNIX now come with a software framework called Pluggable Authentication Modules (PAM). PAM itself does not provide any encryption services, but rather acts as a switching yard for changing security schemes at will. For example, if one has RSA, DES, and IDEA algorithms installed on the system and a user attempts to enter the system using ssh, PAM automatically will negotiate the strongest possible encryption that the ssh client can support. Further, PAM's configuration files permit defining rule sets for users including when and from which machines they may connect.[20]

LOGGING TOOLS

Assuming that an astute administrator has implemented all the solutions described above, how can he be convinced that the effort has indeed enhanced the overall system's security? For that task, one needs to use tools similar to (and sometimes the same as) the ones a cracker would use in probing for weaknesses. Several are described below.

Testing TCP/IP Ports—Satan

The System Administrator's Tool for Analyzing Networks (SATAN V1.1.1) looks at a computer the way a cracker might, searching for vulnerabilities in the network services. Satan will poor over the network service ports and report on any glaring openings, but it's up to the administrator to correct the problems found.[21]

Monitoring Port Access—Courtney

Satan is at least as popular among crackers as it is among system administrators. To learn if a system is being probed by a Satan-like program, one should install and use a port monitor. Courtney is one such program. It logs port activity, and can be configured to mail the administrator if certain events occur.[22]

Tripwire

Tripwire V1.2.2 takes a very simple concept and uses it to provide a nearly bullet-proof means of establishing the integrity of any desired file set. Generally, the administrator sets up a database of system configuration files and key programs. Tripwire is then run, and it computes the MD of each file (recall the message digest concept from the Encryption section). The MD can be computed via several different hashing algorithms, or a combination of them. Three are MD5, MD4, and MD2 (RSA Data Security Inc, San Mateo, CA). Another is the Xerox Snefru Secure Hash Function (Xerox Corp, Stamford, CT). The last two are the Secure Hash Algorithm and the Havel code.[23] To prevent the Tripwire database itself from being corrupted, it is best run on a floppy disk that is then write-locked. From then on, an administrator can run periodic checks of the system files and compare them to the Tripwire floppy. Any deviance denotes that a system file has been tampered with.

Auditing Tools

Both NT and UNIX supply various auditing services that can be enabled and configured to report errors to log files if certain events occur. In UNIX, the logs may be e-mailed to the administrator, but more often are sent to files in the /var/adm/logs directory. The event levels required to generate a log entry are set in the configuration files for each particular service. Under NT, all errors generated by NT services are written to files that can be read with NT's Event Viewer. Basic events to be audited are failed logins, all administrative logins, disk usage, and any user-level programs that attempt to run with an administrative privilege. For firewall administrators, it is also prudent to log all incoming traffic from the "outside." One can study the outbound traffic as well (to assure that users are not violating company use policies).

RESULTS

Considering the preceding sections outlining cracker strategies and the security measures to combat them, we now come full circle to reexamine the console and networked modes of operation. Now we shall include advanced techniques for both cracking and defending the target systems. There are two important points that the reader should be aware of by now. First, networked systems always will be less secure than a console system for the simple reason that they inherit all console weaknesses plus those associated with the network. Even if a cracker can only accomplish an obscure DSA attack (where no data are actually compromised), the loss of functionality itself may be a security breach. This brings us to the second point. Consider the consequences of a cracker being able to cripple surveillance systems, auto-pilot navigation systems, or safety interlocks in a petroleum refinery. Clearly, reliability and uptime should also be considered in the security equation.

In the Console and Networked discussions below, we first examine the additional methods each OS provides for enhanced security, but which are not enabled by default. We then look at how those stock OS enhancements can be defeated and examine the third-party alternatives.

Console Operation (Not Networked)

Primitive OS (Mac and Windows 95/98)

- OS Resident Enhancements and Counter-Attacks

  To simulate the login process that enhances authentication on advanced OSs, Mac users can invoke "At Ease," whereas Windows 95/98 users may enable restrictive user policies. These steps will require the cracker to log into the system as if it were a more advanced OS. Furthermore, users on either platform can engage file-level passwords (ie, Microsoft Word password protection) to encrypt individual files.

  However, At Ease can be circumvented if the cracker knows how to stop extensions from loading (eg, holding down the Apple key during boot). The same is true for password-protected Mac screen savers. Windows 95/98 policies are tricky to implement and thus not often done by most users. Typical application-supplied security passwords (such as those used in Microsoft Word) can be cracked easily by third-party programs (eg, CRAK Software, Phoenix, AZ).

- Third-Party Enhancements and Counter-Attacks

  "Disk Lock" and the improved "For Your Eyes Only" for both Macs and Windows 95/98 can encrypt the entire disk or individual files (Symantec Corp, Cupertino, CA). A 360- to 2,048-bit public key algorithm is used and should be relatively secure.

  Assuming that a user has implemented login passwords and disk and file encryption, the following security estimate is reasonable.

- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
4	4	3	4	3

Advanced OS (NT/UNIX)

- OS Resident Enhancements and Counter-Attacks

  Because a cracker may try brute-force guessing of passwords, it is wise to set a maximum number of login attempts before an account is locked. Unfortunately, this cannot be done to administrative accounts (doing so would mean that no one would be able to recover the system if that account were locked). Also, passwords should be chosen that are not easily guessed (by mixing uppercase and lowercase, using punctuation, etc) to be immune from a dictionary attack–based password cracker. Replacements for the standard password-setting programs for UNIX make this easy to do. Similar abilities are available for NT.

  Auditing should be enabled, at least to identify failed logins, administrative logins, and any programs that attempt to change from a user mode to an administrative mode on the system. Additional auditing may be required (such as disk space quotas) to alert an administrator to unexpected large changes in disk usage.

  However, the system is still vulnerable to a cracker who can reboot it into single-user mode.

- Third-Party Enhancements and Counter-Attacks

  To prevent a UNIX cracker from taking over the system via the single-user boot attack, one can encrypt the entire file system so that the hard drive will not even mount without a password. Furthermore, one can employ file-based encryption to provide file privacy among users on the same computer. Finally, to ensure that no Trojan horses have been planted, the administrator of these systems could run Tripwire to maintain file integrity.

  Finally, one could implement a SecureID system in addition to all the above. Being unable to gain access to the file system, and through the use of file logging, this system is extremely secure.

- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
5	5	5	5	5

Networked Mode

Because the networked modes inherit all console weaknesses, it is first assumed that the users have implemented all the defenses described in the Console section above. The points listed below are additional defenses to deal with the network threats.

Primitive OS (Mac and Windows 95/98)

- OS Resident Enhancements and Counter-Attacks
  At present there does not seem to be much beyond the default behavior to enhance the network security of these systems. However, newer versions of the Mac and Windows 98 OSs may include some VPN tools.
- Third-Party Enhancements and Counter-Attacks
  Users should attempt to use browsers, mail, and other network applications that implement SSL (or similarly encrypted) communication channels.
- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
3	3	2	3	2

Advanced OS (NT/UNIX)

- OS Resident Enhancements and Counter-Attacks
  First, all nonessential network services should be disabled. Second, one should enable auditing on telnet and remote login ports, particularly for administrative accounts. In fact, nonconsole administrative login should be disabled to prevent a remote cracker from directly attacking an administrative account.
- Third-Party Enhancements and Counter-Attacks
  The administrator should add a port monitor (such as Courtney) and probe their system with Satan. Furthermore, a firewall or the equivalent of TCP_Wrappers should be installed to regulate who may access specific services, and from what locations. Ultimately, these services should be combined in a Kerberos-like environment running on SSL (or similarly encrypted) communication channels.
  Using all the tips suggested in the console section plus those listed here, one can greatly diminish the security threats inherent in a networked server. Yet clearly, a determined cracker with enough resources and time might still find a way to compromise the networked data.
- Security Ranking

Authentication	Authorization	Integrity	Privacy	Nonrepudiation
4	5	5	4	4

DISCUSSION

In the preceding sections we have examined the basic vulnerabilities of computers and some methods used to counteract them, and developed a home-grown scale for evaluating systems. The approach is subjective, with a score of 1 representing an essentially open system and higher scores denoting more sophisticated OSs. The reader should be aware that there is a more formal approach to ranking computer security, which has been defined by the National Security Agency of the United States. Some of the contents of the "orange book" have been reprinted by the Department of Defense. The full definition can be found in the Department of Defense Trusted Computer System Evaluation Criteria, published by the Department of Defense Computer Security Center (DOD 5200.28-STD). An actual listing of various products, grouped by their security ranking (with the definition) can be found at this web site: http://www.radium.NCSC.mil/tpep/epl/epl-by-class.html.

Under that taxonomy, NT running in console mode is ranked C2 (if networked, it is unranked). Many UNIX flavors are ranked as C1 or higher (in the B class). Currently, no OSs are ranked at the A level. A key point to bear in mind is that different UNIX versions will have different rankings, and a given vendor (eg, Sun with its "Trusted Solaris") may offer more secure versions of its standard OS.[24]

The advantages to the approach described in this report are several. First, an administrator can examine each specific security area for a given OS, and rank it independent of the others. This allows for a finely grained comparison of the strengths and weaknesses of competing OSs. Second, the resulting specificity should aid the administrator in identifying individual weaknesses, and thereby help to target relevant solutions.

As alluded to earlier, it is also worthwhile to consider the availability of a given computer system as part of the security equation. Especially for servers, it does little good to have a computer that thwarts crackers and keeps data files secure if it is out of service at unpredictable times or for extended periods. Toward that end, fault tolerant, multiprocessor systems with redundant disk arrays should be considered. Alternatively, tightly clustered networked computers may be used that have the ability to monitor each other and take over the functions of a failed cluster member. Although perhaps not immediately obvious, the concept of "openness" also impacts reliability. *open* In this instance, refers to the independence of a software platform from any particular vendor, and implies a consortium has produced publicly available specifications and documentation (Open Software Foundation, Menlo Park, CA). UNIX, TCP/IP, and the X Window system are examples of "open" software. Because of their common origin, the various flavors of UNIX share many common weaknesses and strengths. Whereas this can lead to a single bug being a threat to many systems, it also indicates that the fix will be rapidly found and distributed. In contrast, NT (owing to its single source) is not as well (publicly) documented, and system administrators have little recourse when attacked except to hope that Microsoft will acknowledge the weakness and provide a fix.

It is also important to note that security depends on the overall computing system architecture, not just encrypting certain parts or programs on a given machine.[25,26] For instance, the proper choice of firewalls and switched networks simplifies and increases the security of all machines. Similarly, restricting physical access to the consoles of servers does much to eliminate single-user "maintenance" attacks.

Finally, one must be prepared to look in nontraditional places for information on computer security. Although texts are available, the rapidity with which a security threat can develop can best be countered in a more immediate forum. Hence, in the Additional Sources section of this report, several Worldwide Web–based sources are provided.

CONCLUSION

In summary, with current technology it is possible to provide extremely secure networked computer environments. Evidence of this is demonstrated every day by (the best) worldwide Web–based retailers. Furthermore, it is incumbent on medical informatics managers to consider the security of their patients' medical records and take steps to protect them.

GLOSSARY

CHAP: Challenge Handshake Authentication Protocol. A method for sending hashed (not clear text) passwords across insecure PPP links. Subsequent data are in clear text.

DES: Data Encryption Standard. A private key algorithm developed by IBM.

DSA: Denial of Service Attack.

Firewall: A device that has two network cards attached to a public network and a private one. Traffic flowing between the two is regulated by type, origin, destination, and other rules.

FS: File system, used by the computer to access local disk files.

HTTP: Hyper Text Transmission Protocol. The definition for sending data between web servers and client browsers.

Kerberos: Essentially a more complex version of a firewall, a Kerberos server authenticates users before they may reach computers behind it. The Kerberos server also may regulate which resources specific users may access based on the user's rights.

MD4, MD5: Hashing algorithms that produce a unique cipher output for any input. These two algorithms are patented by RSA.

NFS: Sun's Network File System. Performs the duties of an FS to disks on remote network computers.

OS: Operating System. The software that a human uses to interact with a computer and which, on the computer, coordinates access to various resources such as the File System.

PAP: Password Authentication Protocol. A method for sending hashed (not clear text) passwords across insecure PPP links. Subsequent data are in clear text.

PGP: Pretty Good Privacy. A hybrid public/private file encrypting program.

PPP: Point-to-Point Protocol. A method of sending TCP/IP packets across serial (phone) links.

PPTP: Point-to-Point Tunneling Protocol. A superset of PPP developed by Microsoft to encrypt all data flowing through the communication channel.

RC4, RC5: Two public key encryption algorithms patented by RSA.

S-HTTP: Secure Hyper Text Transport Protocol. A version of HTTP that implements SSL on the TCP/IP port used by the web server.

SSL: Secure Socket Library. Netscape's modification to UC-Berkeley's sockets library to provide encrypted communication on TCP/IP channels.

TCP/IP: Transmission Control/Internet Protocols. The fundamental definition of how data packets are constructed to move across the Internet.

VPN: Virtual Private Network. Any method that can provide encrypted (secure) communication across an open and insecure channel (ie, the Internet).

REFERENCES

1. Makris L, Argiriou N, Strintzis M: Network and data security design for telemedicine applications. Med Informatics 22:133-142, 1997

2. Willenberg C: Strategy for securing medical documents by electronic signature and encryption. Radiology 37(4):305-312, 1997

3. Summers R: An overview of computer security. IBM Systems J 23:309-325, 1984

4. Wouters P: Designing a safe network using firewalls. Linux J 40:32-38, 1997

5. Thomas RO: Haunted by the ghost of smurfing. Sys-Admin 7:63-64, 1999

6. Stewart D, Maginnis P, Simpson T: Who is at the door: The SYN denial of service. Linux J 38:12-16, 1997

7. Brotzman L: Wrap a security blanket around your computer. Linux J 40:17-23, 1997

8. Corcoran D, Sims D, Hillhouse B: Smart cards and biomentrics: Your key to PKI. Linux J 59:68-71, 1999

9. Garfinkel S: PGP: Pretty Good Privacy. Sebastopol, CA, O'Reilly & Associates, www.oreilly.com

10. Scott C, Wolfe P, Erwin M: Virtual Private Networks (ed 2). Sebastopol, CA, O'Reilly & Associates, www.oreilly.com

11. Garfinkel S, Spafford G: Practical UNIX & Internet Security (ed 2). Sebastopol, CA, O'Reilly & Associates, www.oreilly.com

12. Stallings W: Network and Internetwork Security Principles and Practice. Newark, NJ, Prentice Hall, 1995

13. Schneier B: The IDEA encryption algorithm. Dr Dobb's J 18:50-56, 1993

14. Stallings W: Pretty good privacy. ConnecXions 8:2-11, 1994

15. Giles B: Encrypted file systems. Linux J 51:64-67, 1997

16. Mauriello E: TCFS: Transparent cryptographic file system. Linux J 40:64-68, 1997

17. Richter J, Cabrera L: A file system for the 21st century: Previewing the Windows NT 5.0 file system. Microsoft Systems J November 1998

18. Whalin G: Virtual private networks. Sys-Admin 7:21-26, 1999

19. Stein JG, Neuman C, Schiller JL: Kerberos: An Authentication Service for Open Network Systems. USENIX Conference Proceedings, Dallas, TX, Winter 1998

20. Fenzi K, Wreski D: Linux Security HOWTO, 1998. http://metalab.unc.edu/mdw/HOWTO/Security-HOWTO.html

21. Havelt R: SATAN: Analyzing your network. Linux J 40:77-78, 1997

22. Ali S: Freeware based security. Sys-Admin 8:39-44, 1999

23. Fenzi K: Tripping up intruders with tripwire. Linux J 40:75-76, 1997

24. Russinovich M: NT vs. UNIX: Is one substantially better? Windows NT 4:121-132, 1998

25. Hare C: IT Security coming of age. Sys-Admin 7:57-64, 1998

26. Epstein MA, Pasieka MS, Lord WP, et al: Security for the digital information age of medicine: Issues, applications, and implementation. J Digit Imaging 11:33-44, 1998

ADDITIONAL SOURCES

Articles:

a. Cypherpunks home page is at http://www.csua.berkeley.edu/cypherpunks/ This group from UC-Berkeley has one of the best collections of encryption software and technical articles on cryptography.

b. Computer Emergency Response Team (CERT) is at http://www.cert.org Carnegie Mellon provides e-mailed alerts on new-found security issues to system administrators. They also log fixes to previously found security bugs in system software.

c. Electronic Frontier Foundation is at http://www.eff.org The EFF is a collection of technical articles, legal challenges to encryption users, and progress reports on groups whose goal is to break cryptosystems.

d. Humberto Barberá's security page is at http://www.um.es/~humberto/security.html Contains an excellent collection of documents and reports from the DOE Computer Incident Advisory Capability and CERT articles.

e. Phrack Magazine is available at http://www.phrack.com

Tools:

a. SATAN can be found at http://www.fish.com/~zen/satan/satan.html

b. sniffit is available at http://reptile.rug.ac.be/~coder/sniffit/sniffit.html

c. NTCrack information can be found at http://www.njh.com/latest/9703/970329-01.html