A Short Course on

Information Protection in Personal Computers and Local Area Networks

by Dr. Frederick B. Cohen

ASP Press	FAX:412-422-4135
PO Box 81270	TEL:412-422-4134
Pittsburgh, PA 15217	USA

ISBN#1-878109-14-6

Contents

1 Introduction

The proliferation of Personal Computers (**PC**s) and Local Area Networks (*LAN*s) has brought about a resurgence in computer security problems that were largely solved for mainframes, and a wide variety of new problems that are specific to **PC** and *LAN* technologies.

In most **PC**s, there are no access controls, backups are poorly managed if they are kept at all, the operating system can be accidentally corrupted, and denial of service is commonplace.

In most *LAN*s, the access controls in place can be easily violated, there is no protection against any user examining or modifying most of the network traffic, and existing controls are not adequately used.

Subtle interactions between different machines on *LAN*s can extend security problems, sometimes making even seemingly secure environments open to simplistic attacks.

The purpose of this book is to explain the nature and scope of **PC** and *LAN* protection problems, describe the defenses available today and the tradeoffs in their use, and to discuss available tools for understanding the technology.

About the author:

Dr. Cohen is widely known in computer security circles, where he has developed basic theory, performed ground-breaking experiments, prototyped practical solutions in widespread use, developed commercial security products for over 15 years, and provided security products and services to organizations worldwide.

In 1986, Dr. Cohen received his Ph.D. in Electrical and Computer Engineering from the University of Southern California with a dissertation titled "Computer Viruses", which has become internationally recognized as the seminal work in that field. He was a professor of Electrical Engineering and Computer Science at Lehigh University then the University of Cincinnati from 1985 to 1989, and is currently the President of ASP, a company specializing in information protection.

Dr. Cohen has published over 40 papers in the last 7 years, has been an invited speaker at almost 100 conferences, is the author of several well known books on computer security topics, and has appeared in numerous television shows, radio programs, newspapers, and magazines worldwide. He is a member of the international board of editors of the IFIP Journal "Computers and Security", and the DPMA, IEEE, and ACM "Annual Conference on Computer Viruses and Security".

2 Basic Protection Concerns

2.1 Why Protection?

Most modern computer systems do not protect their users' interests. Problems can be caused by malicious people, hardware faults, typos, and power failures, just to name a few. Organizations tend to catagorize protection issues in terms of damage and look for ways to protect against the sorts of damage they are concerned about. The most common types of damage are:

- Corruption: the inability to maintain standards of accuracy, or suitability. Systems that protect against corruption are said to maintain 'integrity'.
- Leakage: the inability to keep confidential information in confidence. Systems that protect against leakage are said to provide 'secrecy', 'security', or 'privacy'.
- Denial: the inability to provide expected services. Systems that protect against denial are said to provide 'availability'.
- Evasion: the inability to accurately characterize things that take place. Systems that are able to accurately characterize things are said to have 'accountability'.

Integrity, privacy, availability, and accountability do not exist in a vacuum. Without all of them, we cannot currently achieve any of them with any degree of certainty. Because they are interdependent, we must consider them part of an overall protection strategy, and apply them in proper combination in order to achieve our protection goals.

2.2 Integrity

Most people I have talked to are astonished to find out that 50% of the people at my talks who track their telephone bills find at least one error per year. This is doubly puzzling when you realize that these telephone bills are entirely automated; there are no human data entry errors involved. If we could eliminate these errors without substantial cost increases, most people would support this integrity improvement.

Computer viruses are able to spread from program to program, user to user, system to system, and network to network, corrupting programs and data as they spread. Viruses have brought *LAN*s with hundreds of **PC**s to a complete halt, and in some cases have caused total loss of information. There are several hundred well known viruses at this time, and on the average, new viruses are placed in the environment more than one per week. It is common for maintenance people to accidentally carry viruses from one organization to another. Salespeople often provide demonstration disks, and in several cases, these have contained viruses. Software released by the best known software vendors have even contained viruses.

A typical *LAN* based computer fraud involves the modification of a list of payments before they are printed and summarized. Even though the computer security on one machine on the *LAN* protects the file from access, the *LAN* introduces vulnerabilities that the machine on the *LAN* cannot defend against. Another typical fraud involves forging a returned purchase, thus generating a reimbursement when there was no original payment. If the system doesn't have a sufficient set of integrity controls, it cannot tell that no payment was made in the first place.

It should be clear from these examples that integrity is not as simple to maintain as we might imagine. In fact, the protection community cannot even agree on a definition. Since different people have different standards, integrity requirements and capabilities may vary dramatically from system to system. Some people include such areas as authenticity of information and/or its source, the suitability for a particular purpose or utility of information, the accuracy with which the information reflects a worldly state, and control over changes, to describe aspects of integrity. There are clearly many viewpoints on this subject.

In current **DOS** based *LAN* environments, these problems are so widespread and severe that any user can corrupt information throughout a *LAN* in a matter of minutes. Access is often granted by default, so that any *LAN* user can access all *LAN* information. Many *LAN* users have their *LAN* passwords on-line and no access control on their **PC**, so anyone who enters their office can cause widespread corruption. **DOS** is not even able to even maintain the integrity of the operating system itself, much less programs and data stored on-line.

2.3 Privacy

When people in the United States apply for health insurance, they are usually required to provide blood and urine samples that are tested for a number of disorders. These results are kept in a national database which is accessible to hundreds of thousands of people in the insurance industry, government agencies, doctors, and lawyers. Most people find this a violation of their privacy, and many find it appalling that they don't have rights to control information about their own bodies.

It is common practice to make credit checks on people and companies being granted credit in order to assure that they have a history of paying their bills. Since credit is so widely used, there are a very large number of people that can get this information. As an example, hundreds of thousands of people can find out which bills you pay on time, which you pay late and how late your payments are, what type of car you drive and what you paid for it, where you live and what your mortgage payments are, what credit cards you have, how much you have borrowed on each, and what you pay in interest for them. When John Belushi died, the author of the best selling book on his death was able to trace virtually every one of his movements over several weeks purely by tracing commonly available credit transactions.

Cliff Stoll grew famous in computer security by helping chase down a network attacker who accessed sensitive information in government systems throughout the United States. The attacker was in Germany, and without even paying for long distance phone service, he accessed hundreds of government and industry computers (primarily by guessing passwords), gathered substantial quantities of SDI related information, and was only caught because one individual chanced upon a minor accounting discrepancy. It is a virtual certainty that thousands of others are actively pursuing similar activities in global networks today.

It should be clear from these examples that the problems of privacy are both significant and widespread, and are not being adequately addressed in many modern systems and networks. These examples are cases where relatively unsophisticated users can exploit widely known inadequacies to get information they may not have a legitimate need or right to see. Further, these are all examples of attacks on systems with underlying access control, password protection, and a significant number of staff members dedicated to the protection function.

The problem is far worse on **PC**s and **PC** based *LAN*s, because the underlying design of the **PC** makes privacy protection inherently difficult, **DOS** makes no attempt to ameliorate this problem, and *LAN* designers are largely unaware of protection issues or proper solutions to the issues they are aware of. In fact, *LAN*s are designed to provide increased flow of information, and many *LAN* designers simply don't understand that there are conflicting goals of privacy and information flow. On almost every installed *LAN* today, any user can watch user IDs and passwords as they are entered over the *LAN*, and thus gain access to virtually all of the information stored on the *LAN* in under one day.

2.4 Availability

In 1989, 1990 and 1991, tens of thousands of telephone customers lost service for 8 hours or more. The impact of this sort of event is truly staggering. In 1989, a Chicago substation had a fire that caused loss of service to tens of thousands of customers for over two weeks. In the 1990 incident 50% of US long distance calls didn't go through for half of a business day. The losses in 1991 included loss of air traffic control capabilities in the New York city area.

The Internet Virus of 1988 caused nearly complete denial of services to about 6,000 computers, and severe degradation of networking services to 60,000 computers for several days. The IBM Christmas card virus of 1987 resulted in worldwide denial on thousands of mainframes for several hours. The Jerusalem-B virus has caused several **DOS** based *LAN*s all around the world to lose all information and cease all processing for hours to days.

Fire, water, and natural disasters have caused numerous cases of denial for organizations of all sizes and types. The San Francisco earthquake of 1989, the high winds in the UK and Europe in 1990, and the flooding in Australia in 1990, all resulted in widespread denial of services. Man made disasters have had similar results. War, terrorism, and insurrection, all have serious denial implications.

Clearly there is a limit to how well we can provide service continuity under some circumstances and what we are willing to pay for the privilege. Very few organizations can achieve continuity in the event of thermonuclear war, but of course that is not a very likely event. Fire, theft, earth movement, water, and power failure tend to be the most likely physical causes of service denial, while programming errors, operator errors, and computer viruses tend to be the most common non-physical causes of service denial.

In **PC**s and *LAN*s, the denial problem is very different than in mainframes. Since **PC**s are relatively distributed, there is inherent redundancy that makes it is less likely to sustain widespread damage from natural causes. On the other hand, there tend to be many **PC**s with hardware failures in large networks. A **PC** has a typical hardware failure rate of about one failure per 2 years. For a network with 100 **PC**s, this translates into about 1 failed **PC** per week. Backups in most **PC**s are rarely or never performed, and almost never verified or tested before they are needed. In most *LAN*s, backups of the file server are performed fairly regularly, but much of the information resides on the **PC**s which are not adequately backed up.

2.5 Accountability

When a bank that processes electronic funds transfers (EFTs) has a system failure, the implications may be extreme if accurate records are not available. The average bank transfers it's entire assets in EFTs about every two days. If a system crashes and there aren't adequate records accounting for the transactions over the last 48 hours, the bank could lose track of all of it's assets. Even with adequate audits, it is sometimes quite difficult to reconstruct events. In one case that I am aware of, there was a discrepancy of about 10 million dollars picked up by an internal audit (due we believe to a programming error). Reconstruction was required in order to determine what had happened. It turned out that there were hundreds of millions of dollars in transaction errors, and it took several months to reconstruct all of the information.

Large accounting firms that do EDP audit work use audit trails to detect frauds and other sorts of abuse. Without good audit trails, they cannot accurately assure that the transactions accurately reflect what transpired, and thus cannot assure that the records are correct. Even with accurate audit trails, some evasion is hard to detect. The so called 'drip' fraud that operates by taking many small amounts of money is particularly hard to detect because minor discrepancies are common as a result of normal business operations.

In a number of cases I have been involved with, organizations with networks have lost on-line data due to file-system failures or intentional acts. They had backups, but because they didn't have adequate audit trails, after recovery from backups, they couldn't tell what had changed since the restoration. In these cases, they simply lost an unknown amount of information.

Without accountability, proper operation cannot be assured to any reasonable degree. Evasive techniques can be applied by system operators or users, inaccurate entries cannot be tracked down, users can act as other users without risk, and users can evade all responsibility for their actions.

In current **DOS** and *LAN* environments, there is little inherent auditing capability. Even with auditing, under **DOS** it is ineffective without identification and authentication, because there is no way to associate a **DOS** user with an action. Anyone who walks up to a **PC** is normally able to do anything that **PC** can do, and most add-on audit systems are not secured from modification or deletion.

2.6 Assurance Issues

Many people who buy protection products want to get an idea of 'how much' protection they get. There are at least three important dimensions to this issue. One dimension is the breadth of protection, which deals with the types of attacks covered. Another dimension is the difficulty of attack, which deals with how much knowledge and time it takes for an attacker to violate protection with impunity. A third dimension is the degree to which the system assures that protection is kept in place and properly configured.

A key issue in protection decisions is to balance these three aspects of protection to give as high a degree of assurance as possible within the limits of price, performance, space, and features. This balance is achieved by trading off breadth, difficulty of attack, and assurance. Wherever possible, these tradeoffs should be kept flexible by providing configuration parameters, so that the user can choose the most appropriate settings for the application. However, as the number of parameters increases, the difficulty of management also increases, and sophisticated tools for protection management become vital.

The issue of breadth is covered throughout this text, and we only pause to note that in order to decide on the appropriate coverage, you have to understand your needs and how the available techniques fulfill them. The more you understand, the better decisions you will make.

Most protection systems are not very difficult to attack by a knowledgeable expert with physical access to the computer. In order to protect against these attacks, it is necessary to have physically secure devices, hardware based encryption and access control, redundant hardware and software, and a wide variety of other things. By design, you should seek difficulty of attack in each facility appropriate to the exposure being covered and the perceived likelihood of attack. In software, the best you can do is normally to make automated attack not specifically directed against the defense very difficult, make human attack by the average user very difficult, and attack without physical access very difficult.

Management tools are usually critical to assurance. A relatively sophisticated and automatic management tool assures that the protection will be used and properly configured for the application.

One final note on assurance. In almost every environment, the largest hole in protection is user complacency and ignorance. Without some training in and awareness of information protection, it is a virtual certainty that users will walk away from logged in systems, write passwords on the console, forget to do backups, leave doors unlocked, use any floppy disk that comes along, and tell unknown callers from ‘maintenance’ their user ID and password. Protection is something you do, not something you buy! Tools can help you provide effective protection with a minimum of difficulty, and help teach you about some aspects of protection, but they cannot force you to act in a prudent fashion. Protection tools help those who help themselves.

2.7 Financial Issues

The bottom line of protection is the same as the bottom line of the organization. We use information protection because there are potentials for financial loss (called ‘exposures’) due to information system vulnerabilities. Corruption, leakage, denial, and evasion, all have the potential for causing financial loss ranging from minor to catastrophic. In order to make rational protection decisions, we must consider the costs of defenses, the exposures they cover, the likelihood that they will be needed and effective, and the relative import of our protection options given budgetary constraints.

3 Defensive Techniques and PC Insecurities

There are several common techniques used to provide protection in modern information systems and networks. We quickly describe these techniques here.

3.1 Procedures

Procedures performed by system users, operators, and managers, are commonly used to facilitate protection. A typical example of a procedural defense is the policy that no external disks be placed in systems. This policy is commonly used to prevent the entry of viruses. Policies exist for effectively controlling all aspects of protection, even though they tend to be relatively ineffective without adequate education, training, and management of the procedural methods.

In **DOS**, no procedural methods are typically advised for or provided with the system. The one exception is manuals and software to perform system backups, hence providing limited coverage against corruption and denial. In **DOS** based *LAN*s, procedures tend to be designed to prevent effective protection, often including facilities for bypassing the need to enter a password for *LAN* access, making effective backups nearly impossible, and generally providing methods for bypassing any attempt at adding protection.

3.2 Physical

Physical protection such as uninterruptable power supplies, locked doors, bolting computers to desktops, and guards are commonly used to prevent theft, deny access, and prevent all manners of other abuse.

PCs make physical protection harder because they are more widely distributed, light weight, easily hidden and transported, use tiny floppy disks that contain a great deal of information, and are harder to account for than larger mainframes. *LAN*s tend to spread the physical information system over a large area, making for a much more difficult physical protection problem.

3.3 Architecture

Most computer architectures provide separation of the operating system from user programs, restricted instructions for physical access to peripheral devices, paging for efficient use of resource, and other protection related mechanisms. **PC**s were originally designed without any sort of architectural protection, making operating system protection very difficult to attain with any substantial degree of assurance.

3.4 Operating System

Most computer operating systems are designed to separate users and processes from each other, require user identification and authentication, protect stored information, provide mandatory and discretionary access control, automate many procedures, maintain audit trails, protect against object reuse, and provide other protection features.

DOS provides no protection features worthy of consideration, and the operating system is designed to be arbitrarily modified by user programs, so as to make protection very difficult to implement. Most **DOS** based *LAN*s provide only the bare minimum protection of user identification and authentication, and limited discretionary access controls.

3.5 Redundancy

Many computer systems are provided with CRC coded disks with automated error correction, SEC/DED memories which detect and correct memory parity errors, redundant file structures for automating recovery from file structure corruptions, redundant operating system structures to detect stack overflows, bound system call parameters, and recover from minor faults, auditing of system activities, etc.

Most **DOS** based computer systems provide no redundancy to speak of, and thus many users augment the operating system with utilities and other tools in order to recover manually from the sorts of minor faults that other systems recover from automatically. In most **DOS** based systems, a single memory parity error, a single bit error in the file allocation table, the boot block, the operating system, or any other program file, can cause the entire system to collapse.

3.6 Cryptography

Relatively few operating systems apply substantial amounts of cryptography, but many systems provide cryptographic facilities and use cryptographic techniques for such tasks as password protection. In Unix systems, automation is provided for editing and executing encrypted files without explicit decryption. Recent Unix based *LAN*s have been developed with *LAN* encryption built-in, and many networks encrypt the network packets to prevent casual observation.

DOS provides no cryptographic capabilities, and any software based encryption technique is severely limited by the ability of an attacker to observe cryptographic keys as they are entered or stored. Most **DOS** based *LAN*s have no cryptographic protection except coverage of passwords at *LAN* login.

3.7 Applications

Many applications for timesharing systems are designed with their own protection facilities to augment and exploit the underlying protection capabilities of the operating system and architecture.

DOS based applications are rarely designed with any protection considerations, partly because the operating system and architecture provide no protection facilities for them to exploit, partly because their protection is almost always ineffective, and partly because most **DOS** programmers have no background or understanding of protection issues.

3.8 Summary

	C_p	C_d	C_l	L_p	L_d	L_l	D_p	D_d	D_l	E_p	E_d	E_l
Procedures	+	=	+	+	-	=	+	=	=	+	-	-
Physical	+	=	-	+	=	-	+	=	-	+	-	-
Architecture	+	-	-	+	-	-	+	-	+	+	-	-
Oper Sys	+	-	=	+	-	=	+	-	-	+	-	=
Redundancy	+	-	-	-	-	-	+	-	+	+	-	-
Cryptography	+	-	-	+	-	=	-	-	-	+	-	-
Application	+	-	-	+	-	-	+	-	-	+	-	=

+ Feasible/Common, - Infeasible/Uncommon, = Mediocre

	Possible	**DOS**	*LAN*
Corruption	C_p	C_d	C_l
Leakage	L_p	L_d	L_l
Denial	D_p	D_d	D_l
Evasion	E_p	E_d	E_l

4 Protection By Extension

The basis of all protection in modern computer systems is the extension of a basic protection mechanism to cover a broader spectrum of desired facilities. In **PC**s, we have only one basic protection mechanism; the write-protection on the floppy disk. With hardware modification, we can introduce ROMs and special purpose protection hardware. In current *LAN*s, cryptography is the only way to achieve any degree of real protection, and while file servers provide nominal protection against some sorts of problems, a great deal of work is needed to improve the situation.

4.1 The Write-Protection on the Floppy Disk

At **PC** bootstrap, the hardware of most **PC**s loads a system bootstrap routine from the first sector on the floppy disk, or if there is no floppy disk in the floppy disk drive, from the first sector of the hard-disk. From that point forward, the computer system is controlled by the sequence of programmed instructions provided to it. Assuming that the **PC** has not been physically tampered with, the protection provided by the write protection hardware assures only that the contents of the floppy disk will not be modified by the sequence of programmed instructions.

The only thing we can then assure is that the program run at startup cannot be modified by any operations performed thereafter, and thus that the same program will be run at startup each time we restart the computer, as long as we have this disk in the disk drive every time we bootstrap the computer. What then can we do with the programmed instructions to provide further protection?

It turns out that any general purpose computer can perform any computation that any other general purpose computer can perform, as long as there is enough total accessible storage and performance is not an issue. That means that by providing an appropriate bootstrap program, a **PC** could simulate any other computer system (even a mainframe), and provide as good a protection facility as can be provided by the most secure system in the world.

Unfortunately, performance suffers greatly when we simulate instructions, usually by a factor of 10 or more. Furthermore, the reason many people choose **PC**s is that they provide software and hardware facilities that are appropriate to the application. If you want a mainframe, you will probably get a mainframe. How then can we compromise between very good protection and these other forces.

Expanding to the Operating System

The most common compromise is to provide an interface between the **DOS** operating system and the hardware of the **PC**, which simulates the unprotected hardware when an authorized request is made, and simulates hardware failures, non-existent hardware, or different hardware when unauthorized requests are made. This technique is both easily implemented and full of loopholes.

On a **PC**, a major portion of the hardware interface and the operating system interface operate by calling a set of hardware and/or software subroutines whose addresses are stored in a fixed set of memory locations called the 'interrupt vectors'. By replacing the default interrupt routines with a set of routines loaded into memory by the protection system, we can replace the built-in mechanisms with the ones we design to provide added protection. If we do this at system bootstrap, the operating system itself is essentially unaware of the distinction between the hardware routines stored in ROM and the replacement routines stored in RAM.

Expanding to Applications

For applications requesting services of the operating system or simulated portions of the hardware interface, protection is automatically extended. As long as the replacement routines are compatible with the original operating system routines, applications programs can operate unaware of the protection mechanism except where the mechanism prevents them from performing tasks they would be allowed to perform if there were no protection in place. In these cases, if locality of reference and adequate error handling is provided, the applications programs will continue to operate acceptably.

Where the House of Cards Falls Down

Although this looks very nice on paper, if we look deeply enough, this method falls apart in many substantial ways. Before we discuss these problems, however, we should note that software based solutions of this sort are quite effective in most cases, and operate quite well against automated attackers or attackers without deep knowledge of **DOS** and **PC**s. They are also quite inexpensive relative to hardware add-ons, flexible enough to provide customized protection to meet organizational needs, and may be more cost effective depending on the financial aspects of the situation. There are two basic problems with this scheme.

This software scheme is fundamentally unsound in that a knowledgeable attacker can easily trace processor operations, determine the location of the hardware routines, and bypass the protection mechanism. In fact, it is relatively easy to automate this process for simple cases. Even for schemes that encrypt all information on disk, tracing operation will reveal the placement and usage of the decryption algorithm and all data necessary for decryption, thus presenting the opportunity to bypass the protection mechanism. This mechanism is only sound in limited function applications where it is impossible to trace processor operations.

The second problem is that many **DOS** applications assume that they have full hardware control, and may violate protection by bypassing the protection mechanisms, or may no longer work because the protection mechanism is not in fact identical to the underlying hardware (if only because it provides protection). An example is the **DOS** 'chkdsk' utility which examines physical disk locations to compare the file allocation table (FAT) to the directory structure revealed by **DOS** system calls, reporting and repairing any discrepancies. If the protection scheme modifies only the **DOS** calls, 'chkdsk' will report numerous problems where there are none, while placing the protection scheme at a lower level makes it more complex and less portable.

4.2 Protection By Augmented Hardware

Another alternative for protection is extending central processor hardware protection. Before the advent of **PC**s, most computer systems were designed with hardware based protection. These protection mechanisms normally provide a special set of restricted instructions for accessing peripheral devices and separation of memory areas between the operating system and user processes. The user process typically makes a request by executing an illegal instruction, which is trapped by the hardware and presented to the operating system for handling. The operating system process, residing in its own memory area, and with access to more of the processor instructions, then checks the propriety of the request, performs any appropriate operations for the user process, and returns control to the user process with results placed in user accessible memory areas.

Unfortunately, most **PC**s have no such protection mechanisms, so in order for the operating system to be fully compatible, it cannot exploit the protection features of more recent microprocessor advances. As a result, even though a substantial portion of current **PC** processor hardware is capable of providing this sort of protection, **DOS** does not take advantage of this capability. Instead of providing tens or hundreds of different products for different types of **PC**s, most companies with hardware protection products for **PC**s augment **DOS** protection by providing a limited number of hardware devices. The large number of new **PC** hardware systems and the increasing number of laptop and notebook sized **PC** compatible systems makes hardware based solutions both expensive and difficult to provide.

ROM Based Protection

A fairly simple way to augment a **PC** to provide limited protection is the replacement of the **PC** bootstrap read-only-memory (ROM) to provide an alternative bootstrap process. This is essentially the same as the floppy-disk based mechanism described above, except that there is much higher assurance that the mechanism will not be bypassed by failure to follow procedure. ROMs are fairly inexpensive to purchase and program, and a relatively small number of different ROMs can provide protection for the vast majority of current processors.

Special Purpose Protection Hardware

Several vendors provide customized hardware boards for augmenting protection in a **PC**. These boards are typically designed to encrypt all information residing on disk using high speed hardware encryption devices, and to limit access to portions of disk so as to make access virtually impossible without physical attack or proper identification and authentication as a legitimate systems administrator. They usually operate by replacing the default device control boards and making protection decisions based on their internal programming. Since their internal programmed memory is not normally accessible by user programs, there is no obvious way to trace their operation or bypass their controls.

The major advantage of hardware boards for protection is their ability to rapidly encrypt stored and transmitted information. This makes disks unreadable even by hardware maintenance and repair personnel, makes protection of transmitted information feasible, and provides a high degree of assurance that provided protection will not by bypassed. The major disadvantages are high cost, limited portability, and increased power consumption.

Exploiting Existing Hardware

Although not all **PC**s have hardware protection, a substantial portion of existing **PC**s now have '286, '386, and '486 processors which provide a limited hardware protection capability. It is relatively easy to implement a protection monitor using the 'protected mode' of these processors, and extend protection from there to cover the remainder of the processing environment. We still have a relatively insecure bootstrap process to deal with, but once that is secured, we can potentially provide inexpensive protection with a high degree of assurance.

The only problem with this technique in the current environment is that it has already been exploited by memory managers and windowing systems so as to provide multiprocessing and process separation, but these implementations are not designed to be compatible with other protection mechanisms. As a result, a large segment of the user community is investing in mechanisms that are difficult to augment for protection except through their original vendors, most of whom are not interested in or capable of providing these features.

4.3 LAN Protection Extension

In *LAN*s, protection is currently provided in three different ways. Cryptography is used in order to protect transmitted information, file server operating system protection is provided to control access to file server information, and **PC** based protection is used to protect the *LAN* from being exploited by user actions.

Cryptography

Cryptography is the use of information obscuring techniques to provide protection. Most people think of cryptography as a method for obscuring the contents of a message or file, and this is one of the major applications for preventing leakage, but cryptography has several other protection applications. So called 'public-key' cryptography can be used to prevent evasion in financial transactions or agreements being executed by parties at remote sites, for privacy transformations, and for data integrity protection. Numerous 'private-key' cryptographic schemes are used for privacy and integrity transforms, and can be used to protect against evasion under certain circumstances.

File Server Protection

Most *LAN* file servers are delivered with some protection capabilities in place. The most common protection is access control based on a user ID and *password* provided at *login*. In almost every commonly used *LAN*, the file server associates the user with the machine identification sent by the network interface card or provides a session identifier to that computer. Subsequent messages sent with that identifier are then treated as authorized transactions of the identified user. File servers typically process requests by receiving messages requesting services, and exchanging further messages appropriate to the fulfillment of that request, if authorized.

Allowable transactions are dictated by the authorization mechanism of the file server. Most file servers provide read, write, and execute access to each of the owner, group, and world sets of users over every file and directory on the file server. Some servers provide additional facilities such as 'locked' files that can never be modified once locked, 'append only' rights for mail files and other similar applications, 'setuid' or 'setgid' files for granting the executing process the privileges of the file's owner or group during execution, trusted program facilities for commonly used and trusted applications that are granted special access, etc.

File server protection typically resides in the file server's operating system rather than the user's **PC** and therefore tends to provide process separation between user processes and file server processes. Attack against the server must therefore be concentrated on bypassing protection measures or getting a file server process to execute attack code. In most cases, these attacks are fairly easily accomplished by an attacker with technical expertise and motivation, but that comprises a fairly small segment of the user community.

Extending PC Protection

Additional *LAN* protection can be accomplished by extending **PC** based protection. For example, some **PC** based mechanisms extend transparently to *LAN*s because they operate at the file level based on pathnames. Since most *LAN* interfaces treat *LAN* based files as if they were on a separate disk, this type of mechanism can be as effective on a *LAN* as on a single **PC**.

The problem with extending **PC** protection to a *LAN* is primarily one of assurance, and for current implementation of this sort, there is little assurance relative to the assurance on a typical timesharing system.

5 What PC Protection Packages Do

There are several hundred **PC** based protection packages providing a wide variety of different tradeoffs, but there are a relatively small number of basic mechanisms that form the core of **PC** based protection. There are no widely accepted terms for describing some of these mechanisms, so where no other name is available, we may use tradenames in place of generic terms.

5.1 BootLock Protection

On a **PC**, there is almost never physical protection, there is almost always a floppy disk, and the hardware is almost always designed to bootstrap the computer from the floppy disk, if present. In order to prevent casual attackers from exploiting this mechanism to bypass any on-line protection, it is common to modify the default **DOS** hard-disk 'Master Boot Record' (MBR) so as to make it unreadable via standard **DOS** utilities. On bootstrap from the hard-disk, the mechanism intercepts I/O calls to the MBR and returns appropriate values to simulate a legitimate MBR. This forces bootstrapping to proceed from the hard-disk in order to access hard-disk areas, and permits the protection mechanism to perform some form of identification and authentication prior to granting access to the system

If it is implemented purely in software, any such mechanism can be bypassed by bootstrapping from the floppy disk, examining the physical hard-disk, and determining how it performs a bootstrap. An attacker will typically trace through instruction executions simulating system operating until reaching a point where the data on the hard-disk is accessible. In most instances, this takes only a few instructions, since hard-disk MBRs are only 512 bytes long. Thus, although this mechanism is effective against most common attacks, it does not provide a great deal of assurance, particularly against sophisticated attackers.

5.2 Disk Content Protection

The contents of a disk can be examined despite any software based protection put in place, and in the case of a hardware repair-person, even most hardware based mechanisms can be bypassed to the extent that the bits stored on the disk can be read and/or modified, but this does not mean the content can be understood. There are a number of techniques for obscuring the content of the contents of a disk.

The simplest technique involves obscuring the relationship between disk blocks and files, thus making the ordering of blocks difficult to ascertain. This can be done by encrypting the file allocation table (FAT) that associates filenames with sequences of disk blocks, placing the FAT in an obscure location, or otherwise modifying the **DOS** mechanisms for making this association.

A substantially more comprehensive form of protection can be provided by encryption of the contents of files, the blocks on the disk, or other information units within the file structure. There is a very substantial tradeoff involved in encryption, requiring either hardware assistance or substantial performance degradation in order to operate transparently, and creating a substantial problem of key management.

5.3 Operating System Startup Protection

Operating system startup on a **PC** can almost always be modified by bootstrapping from the floppy disk, interrupting the normal startup process, or modifying the stored copy of the operating system, the bootstrap programs, or other system critical startup information. In order to protect this process from undue modification, access control, limited function, and integrity checking are used.

Access control is described in some detail later, and we simply note here that by preventing access to the portions of the **DOS** disk containing this information, we can prevent its alteration. Similarly by using limited function in all system operations, we may be able to protect the operating system from modification.

Integrity checking is commonly used to detect and correct modifications to the **DOS** startup areas. This mechanism normally works by verifying the operating system contents against redundant information, and replacing portions that have become corrupt with backup copies. The same mechanism can be applied to any other system information.

5.4 Integrity Mechanisms

Integrity protection is commonly provided through change control and integrity shells. Change control is most often used in environments with large numbers of centrally controlled **PC**s. Integrity shells are used in all environments.

Change control is the systematic control of changes. This is normally done in **PC** based systems with limited function, because with general purpose function, only fully automatic change control can be effective.

Integrity shells are essentially automated change control mechanisms that detect and correct arbitrary corruption. They provide the most cost effective virus defense and protect against a host of other types of corruption.

Integrity protection is normally only effective with access control in place to prevent the mechanism from being circumvented.

5.5 User Identification and Authentication

In order to ascertain whether the person using a system is to be granted access, and if access control is in place, in order to determine what access is to be granted, it is necessary to have an identification and authentication mechanism. Such a mechanism usually depends on something the user is, knows, or has, in order to determine whether the identified access is authentic.

Biometric devices are commonly used to determine what a user is. Typically, these include devices for retinal scanning, fingerprint or handprint analysis, finger size and/or shape analysis, handwriting analysis, and in the future we anticipate the possibility of DNA analysis. These mechanisms are normally designed to meet reasonable probabilistic requirements on authentication, and are relatively easy to bypass if the attacker is sufficiently resourceful.

Passwords, pass-phrases, challenge response systems, and algorithmic authentication is most commonly used to determine what a user knows. These mechanisms are normally adequate if properly used and maintained, but most systems don't provide adequate protection against users with inadequate security awareness and training.

Watch sized authentication devices are commonly available for providing authentication by what a user has. These mechanisms usually either generate a time variant 'key' or act in a challenge/response mode wherein the system provides the challenge, and the device provides a response which is then verified by the computer for its authenticity. Such mechanisms typically depend on the quality of the pseudo-random number generators and/or cryptographic mechanisms used in their implementation for assurance. There are common methods to bypass such mechanisms, duplicate and/or forge such mechanisms, replay past transactions, or piggy back on an accepted authentication.

5.6 Access Control Mechanisms

Access control mechanisms provide a means for preventing access to information by unauthorized users. They normally require an identification and authentication mechanism in order to establish appropriate accessibility, and make decisions on a case-by-case basis at runtime. In **DOS** based systems, access control must modify the normal operation of system calls in order to make access control decisions.

Since **DOS** carries a great deal of baggage for backward compatibility and newer **DOS** versions keep adding new access mechanisms, providing access control through **DOS** requires a fairly substantial amount of memory in order to be effective. The smallest mechanisms use almost 2K bytes of resident memory, while many mechanisms take 8K bytes or more. Since the total available memory under **DOS** is often only 640K bytes and many applications use almost all available memory, this may unduly stress the available resources.

Access control is done at many different levels of granularity, typically providing block, record, file, directory, or disk based control. Larger granularity makes management easier, but forces us to structure the system appropriate to protection requirements. Along with access control comes a host of problems in controlling the mechanism, bypassing the mechanism under controlled circumstances, and systems administration.

5.7 Limited Function Schemes

In many systems, function is limited to prevent users from running undesired programs or using the normal **DOS** program facilities. Many users, especially in large organizations, use only a menu system and a select set of programs provided by management. In this environment, limited function is practical, but in most limited function implementations, a clever user can bypass the limited function and get access to general purpose **DOS** applications. At a minimum, a user with access to floppy-disk bootup can get run a general purpose program.

5.8 Auditing Facilities

Audit capabilities generally provide records of ‘security relevant events’ (whatever they are). Typically, audits provide records of operating system calls with specified parameters, activities of the protection mechanisms, and/or activities of applications programs.

Records of operating system activities tend to be voluminous and difficult to analyze and understand. They also tend to require excessive performance degradation and memory usage, if comprehensive.

Audit trails of protection system activities are far more common, and are most commonly kept only in cases where they are requested and where protection modifies the default system operation. Again, comprehensive audit trails tend to be voluminous and degrade performance.

Application programs are generally responsible for their own audit trails, and it is not normally within the scope of the protection mechanism to analyze application specific audits.

Any time audit trails are kept, there must be a mechanism for examining and clearing them, and it is usually appropriate to provide automation for management of these mechanisms. Audit trails often grow without bound and end up consuming so much system space that they result in system failures. Without automation for detecting growing audit trails and managing them effectively, their utility is severely restricted.

5.9 Hardware Enhanced Protection

Hardware is used to enhance assurance. In the case of intentional attack, hardware enhances protection by making physical access necessary in order to succeed in some attacks that would otherwise require only software. Against accidental failures, hardware enhances protection by covering faults that would otherwise cause failures.

The major problems with hardware based protection are that the cost of hardware is very high, interfaces are required for a large number of different hardware configurations in order to cover a substantial portion of the **PC**s in most environments, and hardware enhancement does not usually provide basic capabilities, but only increases the degree of assurance.

Hardware based protection, and hardware facilities in general, are far less expensive if designed in from the inception of a system. As a guideline, adding a basic facility to a CPU chip in the design phase has almost no impact on the eventual cost of the resulting system, whereas adding a board to an existing system typically costs on the order of US$500, and adding external hardware is generally twice as expensive as adding internal hardware.

5.10 Defense in Depth

As a rule of thumb, protection is enhanced by placing multiple layers of defense together. For example, providing integrity protection is far more effective if there is access control in place, because the integrity mechanism cannot be as easily altered. At the same time, integrity protection is vital to the proper operation of any access control mechanism, since the integrity protection can detect and correct unauthorized modification to the access control facility.

6 How PCs Interact With LANs

Almost all of the interactions in modern *LAN*s are based on pseudo-disks, logical devices, file transfer protocols, and terminal communications protocols, each of which operates through the network interface.

6.1 The Network Interface

Access to a *LAN* is gained through a hardware interface between the **PC** and the *LAN*'s communications media. Media may vary over a wide range, typically including wires, coaxial cables, optical fibers, electromagnetic waves in space, or sonic waves in the air.

Some network interfaces have identity codes built into them, while others have loadable identity codes, and still others have no identity capabilities at all. Some use hardware protocol mechanisms, while others are software driven, and still others use combinations of both.

Network transport mechanisms provide the means for moving information from point to point. They depend on the topology of the network and the protocol for information interchange. In a 'star' network, there is normally a central site that routes all information between nodes. In a 'token ring', information is sent through all of the machines in the network, and is extracted by the recipient based on a message header. In a mesh network, information is routed through multiple computer systems before reaching its recipient. In an 'Ethernet' type network, information is put onto a common bus, from which it is extracted by any subscriber based on header information.

Network protocols vary depending on the application, topology, and components. Some machines support multiple protocols, and can thus act as protocol converters and gateways between other machines. There are several levels of protocol (sometimes called 'layers') in most networks, varying from the transport protocol that specifies how signals in the communications media are to be sent and received, to application protocols that specify the syntax and semantics of messages sent between applications. There are usually many levels between the transport and application layers, each with its own protocol.

6.2 Pseudo-Disk Device Drivers

This application protocol intercepts **DOS** disk I/O calls and, if they are made to the network pseudo-device, replaces them with calls to the network, returning appropriate values based on the information returned by the network.

This has the effect of making the network act as if it were one or more disks on the **PC**. The network device that acts like a disk is called a 'file server'. To send something to the network file server, you simply output it to the pseudo-disk, and to get something from the file server, you simply input it from the pseudo-disk. This makes the network transparent to most **DOS** applications, and also makes protection transparent when it is implemented at the file level.

Assuming the protocol on the file server properly makes decisions about the accessibility of information and that low-level disk input and output operations are not permitted over the network, this protocol can provide excellent access control. This is greatly aided by the process separation between the file server and the **PC**. In essence, the file server acts like a timesharing system's file system, and is subject to all of the problems and features thereof. Protection problems are therefore primarily a matter of proper protection policy and implementation.

6.3 Logical Device Interfaces

This application protocol intercepts **DOS** device driver I/O calls and, if they are made to the network pseudo-device, replaces them with calls to the network, returning appropriate values based on the information returned by the network.

This has the effect of making the network pseudo-device act as if it were on the local **PC**. For example, a device driver for a network printer takes printer port output, transforms it into packets under the network protocol, and transfers the printed information to the network printer. When the network printer returns control signals, they are transformed into packets which are in turn transformed back into the appropriate form to act as if they were printer signals returning to **DOS**. Since the **PC** application cannot tell the difference between a real printer and the pseudo-printer, the network can implement a spooling system for sharing the physical printer, print the output on the closest operational printer to the user, or store and subsequently transform the output for printing on a different physical device when a device of the desired type is not available, without modification to other software.

Logical devices have almost no protection capabilities.

6.4 File Transfer Protocols

File transfer protocols are usually implemented as programs that interact with the network hardware so as to contact a process in another computer and transfer a file to or from the other process. This protocol normally requires a user identity and password in order to gain access to the remote computer and run the corresponding protocol on the remote processor. In many systems, the typing of commands on the remote processor is automated to provide a relatively transparent interface, but in some cases, the entire process is manually controlled via terminal communications protocols.

File transfer protocols are normally implemented without any explicit access controls, and they are thus limited primarily by the mechanisms of the machines performing the transfers. In almost all **PC** based systems, this protection is essentially non-existent. For example, it is normally simple to transfer a replacement for the system command interpreter to a remote **PC** via a file transfer protocol. In most cases, file transfers in **PC** networks are between **PC**s and file servers, mainframes, or timesharing systems, and thus the remote system's protection is the only protection provided.

6.5 Terminal Communications Protocols

Terminal communications protocols are normally programs that interact with the network interface so as to transmit characters between two processes on different computers on a character by character basis. This is commonly used to permit **PC**s to communicate with mainframes and other remote systems.

Terminal communications protocols can be exploited in the same manner as a user logged into a system can be exploited. For example, if a user walks away from a **PC** when it is connected to a remote machine an attacker can type on the user's behalf.

7 Basic LAN Insecurities

Most *LAN*s have a variety of different protection problems that can be exploited by attackers. The reasons for these problems are endemic in the nature of *LAN*s as they exist today, and if nothing is explicitly done to compensate for these problems, they will persist.

7.1 Widespread Arbitrary Observation

The problem of observation is inherent in the fact that information in most *LAN*s is broadcasted to many recipients, each of which can selectively examine or ignore the information as dictated by the hardware and software mechanisms in place.

It is a simple matter to purchase *LAN* analysis hardware or **PC** software that performs *LAN* observation, storage, and analysis. With this sort of mechanism in your possession, packets can be observed at will to determine their contents as they are sent between **PC**s in the network. The only limitation is the topology of the *LAN*. For cost reasons, almost all current *LAN*s with from 30 to 1,000 nodes use an Ethernet or token ring architecture in which all packets are available to all nodes.

Effective protection against observation in the typical *LAN* of today can only be attained through encryption of network traffic. This has two basic problems; cost and performance.

The cost of a *LAN* encryption package that runs on every node and server is about US$50 per node. In addition, there is a very hard key management problem in that there must be a secure way to generate 'session' keys to allow two nodes to communicate securely for a period. This introduces complexity and management difficulty.

Performance degradation is a function of the performance of the cryptographic mechanism. For relatively insecure cryptographic methods, performance can be quite good using software only, but as assurance goes up, so does cost per performance. For a reasonably secure network using hardware assisted cryptography at every node, the cost is in the range of US$300 per node, and this still doesn't eliminate the key management problem.

One solution to key management problems is a set of special protection nodes which provide key management facilities. By centralizing key management, we can dramatically reduce the number of keys, provide good central control, and keep the network manageable. We can physically protect the small number of key management nodes and provide secure communications between regular nodes and key management nodes with greatly reduced key management complexity. A key mangement server should cost from US$5,000 to US$15,000.

7.2 Widespread Arbitrary Modification

The problem of modification is inherent in the fact that the same media is normally shared among numerous **PC**s. Just as common access allows observation, it introduces the possibility of insertion and modification of network traffic.

Insertion consists of placing unauthorized packets on the network. This can be used to 'replay' previous transmissions (e.g. financial transfers), to 'piggyback' on top of another user's access (e.g. add a request to perform a function authorized by another user), or to communicate information to unauthorized recipients (e.g. copy confidential memos).

Modification consists of changing the contents of existing transmissions. This is most easily accomplished in a token ring network where each transmission enters each computer in sequence, and is then resent to the next node in the ring. Modification can be used to change the contents of a message (e.g. the number of components in an EDI request), the destination of a message (e.g. send confidential files to the wrong recipient), or to make a transmission unreadable (e.g. corrupt the contents on the fly).

7.3 Widespread Service Interruption

The problem of denial is inherent to the nature of shared media. The most common forms of denial come from modification to make packets unreadable, insertion to reduce available network performance, and media or channel disruption to make the network unusable.

Modification of packets on the network can be used to make them unreadable by other recipients in several ways. We can change the packet header so as to change the recipient identity to a non-existent user, change the contents or internal checksum so as to produce error codes in packet unpacking, or change the packet type so that it is interpreted in the wrong way. This can be extended to any layer in the packet protocol, so that information moves deeper into the recipient's computer before becoming unusable.

Insertion for denial typically consists of placing so many packets on the network that the media or processing time is dominated by the attacker's packets, thus reducing the performance of other communications in the network. In extreme cases, this can cause widespread overflow of network queues and even cause system crashes for systems connected to the network. Similar results can sometimes be caused by exploiting characteristics of the network protocol so as to cause a positive feedback loop, which overloads network communications.

Media or channel disruption is usually accomplished by physical or electromagnetic means. Physically, most media can be temporarily disconnected or permanently severed.

Electromagnetic disturbances can be used to introduce sufficient noise of proper characteristics such that communications becomes highly unreliable. This in turn reduces the effective bandwidth of the media, thus reducing or denying services.

7.4 Unreliable Audit Trails

Audit trails in *LAN*s cannot be reliably maintained because *LAN* traffic can not be reliably traced to a physical source. Using the attack methods described above, we can easily insert audits of non-existent events, modify audits to reflect an altered version of reality, delete audit packets from the network, or prevent transmission of audit information from local **PC**s.

7.5 File Servers Act Like Timesharing Systems

Most network file servers provide protection mechanisms and process separation between **PC**s in a similar fashion to a timesharing system, so all of the problems we have with protection in timesharing systems exist in *LAN*s. For example, computer viruses spread through *LAN* files servers very quickly just as they spread through typical timesharing systems very quickly. Similarly, there are substantial covert channels in file servers, shared areas are often subject to arbitrary observation or modification, simultaneous access problems threaten integrity, undocumented or improperly implemented *LAN* calls, create protection breakdowns, etc.

7.6 Peer Network Problems

File servers as they exist today, provide peer to peer capabilities to make services available at an equivalent level to equivalent users located at different nodes. Thus they extend trust over the *LAN*, but they do not normally extend protection over the *LAN*. The net effect is that piggyback attacks are very simple, computer viruses spread quickly between peers, access codes can often be guessed, and access on a local node often grants more widespread access.

7.7 Stepping Outside The System

On **PC**s, the physical interface to the network is available for exploitation by an attacker. The network interface hardware on most **PC**s is able to send and receive arbitrary sequences of bits to or from the network, and thus it is possible for an attacker to operate outside the bounds of the normal network protocols. This has broad protection implications that we will now explore.

As we discussed above, most network mechanisms provide some degree of protection provided they are implemented and used in a reasonably secure fashion, and that the other network devices provide appropriate protection when they are remotely used. Against most users, this provides a sufficient level of protection to prevent mild abuse, but against any serious attacker, these mechanisms represent, at most, an inconvenience.

The three most common ways to go outside the system in a **PC** based *LAN* are; modifying the **PC** software so that it acts improperly; directly interfacing to the network interface and thus gaining access to low level network signals; and exploiting interactions between the protection mechanisms.

Since the **PC**s in a network contain much of the network communications software, it is almost always feasible to modify the network software on a single **PC** and attempt to gain access to the network in this way. As an example, we can easily get another user's network password by introducing a Trojan horse into a network login program residing on a local **PC**.

Direct access to the network interface is also feasible in most **PC**s. By using a program that observes, modifies, inserts, and/or deletes network packets, it is almost always feasible to forge other users' identities on the network. We can often observe passwords sent in login packets, replay packets, derive authentication codes from packets and present forgeries, delete packets, corrupt accounting information, watch private information, or deny service to other network users.

In many cases, different protection procedures are implemented at different network sites or even different computers in the same network. There are often ways to exploit the incompatibilities of different protection systems in order to succeed in an attack that might not even work on either configuration alone. For example, in a peer network with physical security in one site and logical security in the other site, it is often possible to physically enter the logically secure site, pass a corruption to the physically secured site without bypassing local logical protection, and have that corruption return to the logically controlled site at a peer equivalence that would otherwise be inaccessible in the logically protected network.

8 What LAN Protection Packages Do

There are a small but growing number of available packages to enhance the protection provided by *LAN*s as they are delivered. The most common packages provide enhanced privacy via encryption, enhanced integrity, enhanced access control, enhanced authentication, audits, backups, and management automation.

8.1 Encryption for Privacy

Privacy in *LAN*s is often enhanced through the use of encryption of *LAN* traffic. Depending on the requirements of the application, we can encrypt portions or all of the network traffic, we may use a variety of different cryptosystems, we will have problems relating to the management of cryptographic keys, and there will remain a variety of overt and covert channels for violation of the protection requirements.

What To Encrypt

If we encrypt all network traffic, we are usually faced with severe performance penalties. For example, all nodes on the network would have to try to decrypt every packet in order to determine if they were the proper recipient. This is not automatically done by existing network interface hardware, so built-in hardware capabilities for detecting the recipient address could not be used.

If we do not encrypt header information of this sort, traffic analysis can be done by an attacker to gain intelligence information indicating various conditions. For example, if every time a major stock purchase was approved, electronic mail was sent from the president of a company to all board members, merely detecting that routing information could indicate the deal, and allow stock manipulation.

The same analysis can be carried out at each level at which we choose to consider encryption for privacy. The issue involves price, performance, convenience, protection provided, and exposure levels.

The Cryptosystem

Another important decision is what cryptographic system to use for protection under which circumstances. There are a large number of systems available, and most systems which attain high performance do so with proprietary algorithms that tend to be easier to attack than the few well known cryptographic algorithms. In addition, there are public-key and private-key algorithms available, each of which is more appropriate in various circumstances.

Private key systems are systems in which the encryption and decryption keys are kept private between the communicating parties. The most common systems of this sort are one-key systems in which any party with a key can encrypt and/or decrypt all messages.

Public key cryptosystems provide a lock-and-key arrangement, wherein the lock can be given to a large number of people, and only the key-holder can read the information once locked. The most common public-key systems also provide for digital signatures if we distribute copies of the key which unlocks the lock, while keeping the lock secret.

Combinations of cryptosystems are sometimes used in order to exploit the favorable characteristics of each. For example, public-key systems tend to be very poor in performance, while private-key systems tend to present severe key management problems, so public keys are sometimes used for key distribution of private session keys. Thus we get a much reduced key management problem while keeping performance relatively high once a session has been initiated.

Key Management

One of the most critical aspects of using cryptography in a network is the issue of key management. Simply put, cryptosystems keep secrets by using secret keys to lock and unlock information. We have to assure that the proper keys are in the proper places at the proper times in order for protection to be effective while granting all authorized access.

Key management with private key systems normally requires secure distribution and storage, automatic use, and appropriate changing and destruction of keys. If all parties on a *LAN* are to be able to communicate in privacy, $n^2 - n$ keys are needed for n parties. This is because each party must have a different key for each of the $n - 1$ parties with which communication is to take place. For a *LAN* with 1,000 nodes, this comes to 999,000 keys! Even for a *LAN* with only 35 nodes, this comes to 1190 keys.

Secure key distribution is required in order to get keys to each of the pairs of communicating parties. This normally requires physical protection at some level. Protection is also required for key storage. If we don't protect the stored copy of a key, it can be easily stolen. Keys must be periodically redistributed if we are to limit the exposure from a revealed key, and this required repeated key distribution.

Automatic key use is facilitated through hardware interfaces between the physically secure storage media and the hardware or software of the cryptosystem. If this is completely automated, then we must rely completely upon the security of the key storage media.

Keys must be changed periodically to limit exposures, old keys must be destroyed if privacy is to maintained over time, and keys of users who lose authorization must be destroyed or invalidated in order to prevent their subsequent use.

Public keys have significant advantages over private keys for the purposes of key distribution, because a single lock can be used to securely receive information from an unlimited number of keyholders. Thus the total number of keys required for a network with n users is only $2n$, (one locking key or 'lock' for short and one unlocking key per user). Even more importantly, the lock does not have to be kept private, so all of the locks can be stored on a publicly accessible area of a file server. We only have to protect the privacy of 1 key per user!

Key distribution in a public key system is also far easier than in a private key system, because each user can automatically generate their own private-key/public-key pair, and place their public key on the file server. Updating keys is also simple for the same reason. To deny access, keys can simply be deleted from the file server, thus eliminating their further use. Simple techniques are also available for assuring file server integrity, preventing forged keys from being placed on file servers, and addressing other similar concerns. Public keys can also be used to securely distribute private-key pairs for use in a single session, thus providing high performance once communication is established.

Key Management Servers

Even with public keys, there is still a substantial issue of key management, and we still have to secure all of the private keys on each user's system. A still more easily managed method of providing automation and management of cryptographic keys for a large network is the 'Key Management Server'. A key management server centralizes all key management so that individual nodes do not have to manage them.

The simplest way to perform key management with a server is to provide a single public key for all network users and applications. When a user wants to establish communications, a pseudo-randomly generated session-key is generated by the user and encoded using the file server's public-key. This session key is then used for all further communications. The user normally includes a password with the first transmission to prove identity and establish server-based access control parameters.

To establish communications with another node, the server tells the node that communication is desired. That node then establishes a secure communications link with the key management server, which gives each of the communicating parties copies of the others' key. The parties can then securely communicate till the end of the session.

All of the protection is concentrated in the key management server, with the advantage that more money can be spent providing good security for the server, and the disadvantage that a successful breakin to the server grants the attacker full access. In this arrangement, the key management server can also provide controls over which user can access which application, audit the use of applications, and perform a wide variety of other protection management functions. Multiple key management servers can be provided for enhanced reliability, and key management responsibility can be distributed to limit the impact of a successful breakin to a given server.

Proper Cryptosystem Use

The best cryptosystem in the world is of little value if it is not used in a secure manner. For example, replay attacks are possible any time the same key encodes information without encoding a time and/or usage number. For this reason, each use of a cryptosystem should have usage and/or time stamps.

Another common problem with cryptosystems is excessive use of the same key. Almost any cryptosystem can be broken given enough samples of encoded information. Poor key selection also makes most systems easy to break, and this is of course closely related to the problem of password selection. For this reason, keys should be selected automatically and with a cryptographically strong pseudo-random number generator.

Other aspects of cryptosystem use can lead to abuse, and the designers of such systems should be well versed in this technology. Unfortunately, most designers of protection systems are not aware of most of the historical results or current research, and this leads to many inadequate cryptosystems.

8.2 Encryption for Integrity

Integrity protection for *LAN* transmissions is often provided through the use of authentication codes. Most hardware provides some form of coverage against random transmission errors, but this in ineffective against intentional attackers. For that reason, most common integrity protection mechanisms against intentional attack use cryptographic checksums.

Cryptographic checksums are hard-to-forge representations of information content, designed to make the probability of undetected modification small enough for the application. They can be thought of as digital fingerprints if the analogy is helpful. We take a fingerprint of the information to be stored or transmitted, and store or send the fingerprint with the information. To verify integrity, we check the information against the fingerprint. The attacker may be able to modify the fingerprint and/or the information, but without knowing the proper key, is unlikely to guess a modification corresponding to a legitimate pair.

This method also covers transient errors, permanent faults, and other corruptions whether intentional or otherwise, with the same degree of assurance provided against intentional attack.

Program integrity has only recently become substantially covered with the introduction of 'integrity shells'. Integrity shells are mechanisms that verify cryptographic checksums just before information use. With proper design, they can automate the detection and correction of corruptions, thus preventing transitive spread through the network and automating most aspects of repair.

8.3 Augmenting Basic Access Control

The basic access control provided in *LAN*s is very similar to that provided in timesharing systems. Unfortunately, a typical timesharing system can be completely taken over by a computer virus in a matter of minutes to hours, and other mechanisms such as Trojan horses operate freely in timesharing systems. These problems trace back to three basic problems; proper protection management is often infeasible due to the large number of protection bits and the lack of tools for managing them; protection is usually at an inappropriate granularity to the application; and the protection provisions on most systems are not easily translated into the actual protection they provide.

Protection Management

Protection management of access control primarily consists of translating high level protection requirements into low level access control decisions. In a typical **PC**, there are about 1,000 files, and 3 protection bits per file (write, system, and hidden). Thus the simplest use of standard (and ineffective) **PC** protection bits requires control of about 3,000 bits per **PC**. For a modest network of 35 **PC**s, this corresponds to 105,000 bits! In a typical *LAN* file server, there are from 10,000 to 100,000 files, and about 10 bits of protection information per file (read, write, and execute for owner, group, and world). Thus a typical file server has from 100,000 to 1,000,000 bits of protection information.

Despite the substantial complexity of just managing the bits once we understand how they should be set, there are few substantial protection management tools on the market. Without proper tools for automating the process of setting and verifying all of those bits, this is clearly an unmanageable situation. Even more to the point, the number of protection bits does not even loosely correspond to the information content of the desired protection. If we are to be effective in protection management, we must move towards systems where we specify protection in terms of realistic management desires.

Granularity

Protection schemes provide protection over a variety of different sized things. Hardware protection can often be applied to a physical device such as a disk. Software tends to provide protection of smaller sized things such as disk partitions, directory trees, directories, files or records in files. The sizes of the things being protected dictates the granularity. Finer grained protection requires more storage space and management effort, while larger grained protection makes some policies difficult or impossible to implement. Since most systems have fixed protection granularity, there is rarely an exact match between desired protection and afforded protection, but good management tools can often provide mixed granularity control over fixed granularity protection bits. Unfortunately, such tools are not commercially available, provide little assurance, and have very poor performance. As a result, protection systems tend to provide granularity appropriate to policy decisions made at an early design stage, and follow a protection philosophy based on those policies.

Actual Protection

Even with strong management tools, protection settings rarely provide an accurate reflection of the actual protection state of a system. For example, the 'read' and 'write' privileges are transitive in nature on any general purpose computer. That means that an attacker can often read information even though they do not have read access to the file it was originally contained in. Similarly, a computer virus can spread to areas of a system that the originator did not have write access to.

The general protection implication of the protection state of a machine is undecidable, and thus it is not always feasible to derive the full implication of the protection state of a machine unless protection is specifically defined so as to provide controllability. Most systems grant all users implied access to all information, because protection is controlled in an ad-hoc fashion. The most general structure for modeling implied protection in a general purpose transitive information network with sharing is a *POset*.

8.4 Augmenting Identification and Authentication

Standard *LAN* based identification and authentication provides only a simple one-time password entry. In some systems, passwords are encrypted, but this doesn't provide any coverage of packets sent after the password is authenticated.

The simplest password enhancement is an improved password setting program for the *LAN* that requires the passwords not be too easily guessed. Further enhancements such as pass phrases and challenge response systems can be used to strengthen passwords against random guessing. Hardware and software combinations can also be used to provide cryptographically strong time or use variant passwords.

Even with an unbroken password, it is often possible to piggyback on subsequent packets after a user is logged into a file server. Authentication must cover all packets in transmission if it is to be effective.

8.5 Automating Backups and Audit Trails

Backups on a *LAN* are substantially more difficult than on a timesharing system or even a single **PC**. On most timesharing systems, a small number of systems administrators provide all backup and recovery services, while on a single **PC**, users who act prudently perform backups for themselves, and others remain at risk, but in a *LAN* there is no single person who can perform all of the necessary backups without automation above and beyond the default provided on most *LAN*s.

To augment network administration and provide enhanced network services, most *LAN*s now have either built-in or add-on facilities for running programs on the local machine at *LAN* login. This is commonly used to load TSR programs for *LAN* auditing, security, virus checks, and *LAN* backups.

Default *LAN* audit trails rarely trace more than the source and destination of file transfers, *LAN* logins, and access violations, but there are several *LAN* add-on audit systems that log every program execution on every machine, each file access, and in some cases, even every keystroke.

The main problem with audit trails in a *LAN* is management and interpretation. For a *LAN* with substantial auditing capabilities, 100 **PC**s where each user runs 20 programs a day, accesses 100 files per day, and logs into the *LAN* once in the morning and once in the afternoon, there are 2,000 program runs, over 10,000 file opens, and 200 logins per day. Simply examining the over 12,200 audit records produced each day is too expensive to justify and has nebulous benefit. We also have to clear these audits either from the server or the individual **PC**s, since they typically take about 100 bytes each (including time and date stamp), or about 1.2 Mbytes per day. If we audit every keystroke, this goes up incredibly. In order to be effective, auditing must only store information with high content, and there must be automation for audit trail analysis and management.

8.6 Management Automation

Most *LAN* managers are kept quite busy just keeping the systems on the *LAN* operational. For example, each week a typical manager of a 100 **PC** *LAN* has 1 to 3 failed **PC**s requiring a complete reload, several employee changes requiring protection changes, several new software packages to evaluate, backups and restores of the file server, several other hardware and software maintenance tasks, two or three new software packages to install, several meetings and reports, several software incompatibility problems, and another job to do besides managing the *LAN*. Most *LAN* managers have no prior training for their job and are made *LAN* manager because they seemed like they knew something when the last *LAN* manager left. In environments with a minor attack underway, these numbers mat tripple for a month or more. In an environment with a destructive virus in place, the entire network may fail several times per week, requiring complete system reloads of the server and all nodes!

Most *LAN* protection packages provide some automation to assist in their use. Typically, there will be a different tool for managing each protection package, so that in extreme cases, there will be 5 to 10 tools for management that the *LAN* manager must learn in order to perform nominal protection management. Most tools provide relatively low level capabilities, require a fairly deep understanding of the technology in order to use effectively, and are relatively inflexible. Even the most flexible and comprehensive management tools require some programming skill in order to be effectively used for *LAN* management.

9 Management Decisions About PC and LAN Security

Nothing is free. From a management standpoint, protection is and should be treated as a business decision, not an emotional decision. There are very real exposures in most environments, and very real costs associated with reducing the risks related to those exposures.

9.1 Cost Analysis of PC Security Techniques

Costs dominating **PC** and *LAN* protection consist primarily of management costs, licensing fees, purchasing costs, computer space and time, people time and inconvenience, compatibility with the environment, and losses from attacks. There are very mathematically clean ways of dealing with the probability of attack, the expected loss, and the cost of reducing that expected loss, but unfortunately, these techniques can rarely be applied with accuracy in a practical example. This may seem to make decision making quite difficult, but there are some things we can do analytically to make rational decisions.

The basic risk analysis technique derives an expected loss by multiplying the number of incidents per unit time by the loss per incident. We then assess each defensive technique in terms of the cost of use and the degree to which it reduces expected loss to get it's cost effectiveness. Finally, we implement each cost effective defense in order of effectiveness. The problem with this technique is that it is usually infeasible to accurately determine the number of incidents per unit time, loss per incident, cost of using defensive techniques, and degree to which defenses reduce these factors. The normal solution is to make assumptions and derive results. A more sophisticated method analyzes how results vary as parameters vary in order to assess a degree of certainty for each decision.

In many cases, this analysis produces a few very clear decisions and many less clear decisions. In most **PC** and *LAN* installations today:

- Uninterruptable power supplies, periodic backups, door locks, and fire detection equipment are very cost effective because the probability of incidents and loss per incident are quite high, while costs are not extreme.
- Password protection, access controls, integrity shells, limited software cryptography, and basic auditing are less cost effective but still justified becauses the cost of the defense is quite low, while the loss per incident and probability of incident is moderate.

- Hardware enhanced protection is not cost effective because it provides only slightly greater assurance then software protection, while costing from 5 to 10 times as much.

9.2 Some Typical Decisions

We now examine the basis for some typical decisions about protection we have just described. Our figures are only approximations, and we make many assumptions that you may not agree with. Each case is different and should be analyzed on its own merits.

Uninterruptable Power Supplies

For most **PC** based *LAN*s, the typical cost per system including cabling comes to between \$2,000 and \$5,000. The added cost per **PC** for an uninterruptable power supply (UPS) is about \$250. There are about 10 power fluctuations per year which will cause system crashes in a typical environment. The loss from an uncovered power failure of this sort is typically from \$10 to \$100 per system. In most cases a reboot and thorough system check is all that is required, but in some cases, boards will have to be replaced and substantial downtime will result. The expected loss is thus about \$500 per year, while the cost of a UPS is on the order of only \$25-\$50 per year, and thus a UPS is quite cost effective.

Good Backups

A typical system requires substantial restoration from backups about twice per year. This is typically caused by operator error, power failures at the wrong time, disk and processor failures, maintenance errors, and intentional attacks. In a *LAN* with a file server providing centralized backups and software to automate incremental backups from the **PC**s to the file server, the cost of backups are under \$5,000 for the entire *LAN* plus operator time for file server backups. Without good backups, each loss tends to be about \$1,000-\$5,000, while with good backups, the loss tends to be on the order of \$100 to \$500. For a *LAN* with 100 **PC**s, the loss comes to about \$2500 per year per **PC** without good backups, while good backups reduce this to only \$250 per year, including the cost of the backups.

Fire Detection

Fire detection equipment can cost as little as \$20 per **PC**, while fully automated fire detection and suppression equipment can cost hundreds of thousands of dollars for a medium sized installation. In case of fire, the loss can be total, running from \$1,000 to \$5,000 per system for each of hardware and stored information, with further loss of business resulting from the loss. Fires are not as usual as power fluctuations and loss of information, occurring far less than once per 10 years in most environments. Fire prevention and detection has the added advantage of reducing insurance costs and protecting people's lives, and for that reason, it is almost always cost justified, but not purely from an information protection standpoint.

Software Protection

From a standpoint of cost, software provides comprehensive protection against intentional and accidental abuse for \$50-\$250 per computer, and may even reduce the costs associated with other aspects of *LAN* management. Comprehensive software based protection packages provide password protection, access controls, integrity shells, limited software cryptography, basic auditing, automated *LAN* backup and recovery, and protection management tools. The best software defenses have a relatively small impact on performance, install with relative automation, do not require upgrades, and provide automation for most aspects of maintenance.

The problem in evaluating this technology is determining the expected loss and degree of loss reduction under various defenses, and determining the cost of use. Without good auditing, it is infeasible to determine how many attacks take place and what they cost. Unlike fires and power failures, there may be no obvious indication of these problems. The examples we are most aware of are viruses which cause widespread denial. One easily cured virus per year causing widespread denial in a *LAN* almost certainly costs on the order of \$50 per **PC** per year. This alone justifies such a defense.

Hardware Enhancement

Hardware enhanced protection typically costs from \$300 to \$750 per **PC**, and increases the assurance provided by software, but making hardware changes typically requires complete board replacement. The question that remains unanswered is how much the increased assurance impacts the expected loss. The expected loss per attack would have to exceed the cost difference between hardware and software to justify the cost difference on a successful attack. In the case of protecting a 100 node *LAN* with a cost difference of \$500 per **PC**, attacks that fail with hardware protection and succeed with software protection would have to generate losses of \$50,000 in order to justify the cost difference, assuming that there were not attacks covered by software and uncovered by hardware.

In our experience, only a very small number of attackers are able to bypass software and not bypass hardware. Furthermore, software can easily be improved to meet emerging attacks, while hardware is expensive and difficult to modify. Hardware protection is rarely as comprehensive as software protection, and no commonly available hardware protection is completely software independent. Nevertheless, there are certainly cases where hardware based protection is preferable. In particular, high exposure situations with very few operational changes over time cry out for hardware enhanced protection.

10 Future Directions

The future of **PC** and *LAN* protection is generally good. Protection tools are getting broader in scope so that fewer tools are required in order to provide effective protection, manufacturers are integrating more protection tools into their *LAN* operating systems and hardware, thus reducing the need for add-on tools, and professional *LAN* managers are becoming more common.

10.1 Basic Encryption

Encryption hardware is becoming less expensive, and several systems now have built-in hardware encryption. For example, there are now disks with hardware encryption in the controller, *LAN* interfaces with encryption and authentication codes, and modems with built-in encryption.

10.2 Protection Servers

Special purpose protection servers are currently being designed and prototypes are already being demonstrated. The most hopeful technology at this time is the key-mangement server described earlier.

10.3 File Server Improvements

Most current file servers have major protection problems, and the major manufacturers are addressing these problems on an ongoing basis. Protection is also becoming an important marketing tool in selling file servers and *LAN*s, and as a result, the major manufacturers are becoming more and more interested in protection issues.

10.4 Gateway Improvements

Gateways between *LAN*s rarely provide any protection whatsoever, primarily providing translation services between network addresses, low level packet protocols, and varying media. There are no major efforts underway at this time to change this situation.

10.5 The Impact of Network Standards

There are several network standards that include protection issues, although the most common method for including protection is as an addendum to the standard rather than as an integral part. For example, the ISO's OSI standard has a security addendum, but it is possible for a vendor to fully comply with the standard without any protection whatsoever.

10.6 Analytical Methods

Much improvement is required in analytical methods for network protection analysis. There are some researchers performing basic modeling work in order to help improve the situation, and as these results become published, they result in substantial improvements to the technology base and eventually to the products we see on the market. Unfortunately, there is rarely adequate funding for this work, and those who are funded are most often not permitted to publish their results because of the organization supporting the research and the sensitive nature of the results.

10.7 Management Tools

Protection management tools are improving dramatically and rapidly, but they have a very long way to go. Several tools that have recently been introduced are an order of magnitude better than previous tools in the same area, and some tools are beginning to provide very comprehensive coverage. There is a particularly bright future in this area.