User Friendly Security

Robert A. Karlin
Karlins' Korner
7628 Van Noord Ave.
N. Hollywood Ca.
91605

SECURITY: ... 2. something that gives or assures safety, tranquillity, certainty, etc.; protection; safeguard ... (Webster's New World Dictionary of the English Language Second College Edition: 1976)

From the beginning of the online business data processing environment, the question of protecting that environment has been asked many times. The answers have run the gamut from complete indifference to extreme paranoia. The propounders of these answers have collected numerous arguments, most of which have little or nothing to do with the basic question. In order to adequately explore the question of safeguarding the business data processing environment we will use the following definition of security:

SECURITY: The protection of the business data processing environment from damage WITHOUT SERIOUSLY IMPACTING THE ABILITY OF THE DATA PROCESSING USER TO CONDUCT HIS BUSINESS.

With the emergence of large DP staffs, many professionals have forgotten that the business data processing environment is not an end in itself but a service provided to enhance the efficiency of a particular business. Most security is installed and administered by staff members far removed from the user. Especially in these cases, security officers must balance the need for protection against the impact on the user's operation. In this paper we will explore the kind of considerations that must be taken into account when establishing or maintaining a secure environment.

SECURITY AND RISK ANALYSIS

In order to properly implement security in a data processing environment, one must know what one is protecting, and what one is protecting it from. There are no global rules on how much security is necessary, or what form it should take. Each particular installation must analyze its own environment, identifying the perils to be protected against, and assessing the cost of protection against the worth of what is being protected. We can group the perils in the data processing field into the following categories: Unauthorized Access to sensitive information, Accidental Damage, and Malicious Destruction. For the purpose of this discussion we will ignore the problem of Unauthorized Use (that is, use of hardware or software without either damage or the compromising of sensitive data) because with the exception of response time considerations, the problems of unauthorized use (such as the productivity of an employee who spends his day playing pong) should be handled in other areas (such as personnel) without making security either the scapegoat or the cure-all. Employees who cannot behave in a responsible manner will turn to other forms of diversion, while security restrictions to prevent this type of problem will do nothing more than lower the morale of those staff who are responsible employees. In addition to these considerations, it is impossible to assign a meaningful value to losses suffered due to the unauthorized use of hardware or software, and without a value the imposition of security is a meaningless gesture.

Each different type of peril has different formulae for calculating the loss that could be suffered. Some losses can be more political than actual, and proper security for these cases may consist primarily of education. Other losses, while real, may be incalculable. In these cases, one must be more concerned with the recoverability of the environment than in its protection. True disaster recovery will be part of any efficient security system, and if a disaster recovery plan is in place at the outset, the job of instituting effective security is made that much easier.

Full scale risk analysis for an existing shop can be both costly and time consuming. Most companies opt for ignoring the problem of designing a viable security plan altogether and therefore end up instituting security measures on a haphazard basis. This creates havoc for users and programmers alike, costs much in time and manpower, and usually does not protect the environment from many of the perils besetting it. To make the burden of risk analysis easier to bear, a surface analysis of the environment can establish the types of applications within the environment, and recommend the appropriate security for each type of application based on the both the value of the applications within each type, and the ease in tailoring the security tools available to that application type. New applications standardly can include a risk analysis, while old applications can be retrofitted as time allows, and the risks warrant. In this manner, all systems can be brought to a standard level of protection with far less impact on the current users of the systems.

When performing risk analysis it is easy to focus on the

programmer as the main security risk. Aside from the obvious morale problems in a shop where each employee is treated as either a hardened criminal just waiting to sabotage the system, or a fumble fingered oaf unable to read a file without deleting it, a security system designed to keep programmers in small boxes can double development time, and prevent timely problem resolution. While it is appropriate to place some restraints on the programming staff, it is necessary to allow enough freedom to the staff to do its job. In general, one must be realized that one's programming staff works for the same company, and is not 'the enemy'.

TYPES OF SECURITY

Security comes in a number of forms, and not all forms are appropriate for a particular task. Each application must be examined to determine the most cost effective form of protection. Ease and cost of implementation, susceptibility to damage, and sensitivity of the data must all be taken into consideration.

The first and simplest form of security is the assignment of unique user identification. This security measure is the foundation on which many of the more sophisticated measures are based. Until Until unique ids are assigned, no responsibility can be assigned, and no audit trail can exist. Common user ids for departments or applications remove one of the cornerstones of a good security implementation.

Once unique user ids have been assigned, each user can be restricted to a single on-line session at a time. This prevents the usurption of a user id while the user is logged on, and reinforces good security habits by forcing the user to log off when he leaves his terminal. (If you don't think this is the case, watch a user try to sign on to show his boss a problem).

Password protection of the user id is the next level of security. Passwords should be assigned by the user himself, and changed at reasonable intervals. Requiring passwords to be changed too often, or assigning meaningless gibberish will result in passwords that have

been written down and placed in desk drawers or taped to the terminal. Once the user id has been protected, additional passwords should be used sparingly if at all, since additional passwords will also encourage passwords to be written and stored instead of remembered.

Once the user id is secure, files and applications can be write-restricted to certain users. This helps preserve the integrity of the files. Data bases and other master files should be protected from any unauthorized write access. Transaction files should have a broader access, encouraging the correction of erroneous data by transactions (which are auditable) instead of direct manipulation of the data base (which is not).

Read-protecting data should be limited to instances in which the data itself is sensitive. Many applications overlap, and the integration of data and elimination of redundancy can be greatly hindered by a common policy of read restrictions.

For extremely sensitive material, encryption is superior to read-protection, but costlier to implement. The combination of the two is very secure.

Nonvolatile files can be protected by restricting access to batch programs only. This allows a complete audit trail for all activity against these files, and an extremely easy recovery path in the event of file corruption.

Restricting access to certain users ids, programs or files, can

also be accomplished by terminal location. This form of protection must be applied carefully, since certain minor disasters that affect the location of the acceptable terminals can have major consequences. Also, late night and weekend remote problem resolution can be hampered. Restricting terminals to certain users by user id can pose the similar problems.

Any form of restriction carries with it certain costs not associated with implementation. Restrictions that hamper problem resolution can have serious cost consequences. Restrictions that force users to alter their way of business unnecessarily can also increase operating budgets tremendously. These factors must be taken into account in any effective security implementation.

UNAUTHORIZED ACCESS

Probably the most misunderstood area of security is unauthorized access. Unauthorized access involves the compromising of data or programs, as opposed to Unauthorized use, which is the use of hardware and software without the compromising of data or programs. It is particularly easy to apply the strictest security measures to a whole installation where only small portions of the data within the installation are significant. On the whole, there is usually very little sensitive data in an installation. This data divides easily into three types: economically sensitive data, legally sensitive data and morale sensitive data.

Economically sensitive data consists of information that could cause monetary loss if divulged, such as proprietary software, sales commission rates, future market studies, etc. Security in these areas is more to make the user feel secure, than to actually protect, since it is generally more important to the user himself than to any competitor, and since most of it is available on the corporate rumor mill anyway. In the user community, availability on a need to know basis is appropriate; for the programming staff, nondisclosure agreements are usually sufficient to protect the installation. On those areas that are truly sensitive, including merger plans, inside information, etc., nondisclosure agreements can sometimes be the only protection, since the data will be available to a programmer at the first application program failure.

Legally sensitive data presents a greater problem. Bank account balances, credit information, personnel records, etc. can all be the basis for costly suits if divulged. This data should be protected from any unauthorized user, and most of the programming staff. Only those whose responsibilities directly include production problem resolution should have access to this data, and again, nondisclosure agreements are a necessity.

Morale sensitive data can be the most difficult security access problem. The largest area of morale sensitive data is payroll information. Any company that does its own payroll is asking for trouble. If there is no way out of it, responsibility for payroll problem resolution should be relegated to one's most trustworthy staff, and no one who handles payroll data should be drastically underpaid.

Another area of morale sensitive data can be online interoffice memos. Most mail packages do not encrypt these missives, and many use data bases that are accessible to all users.
In these cases, staff must be cautioned from using online mail for any message that could be inappropriate for general release.

The general rule of thumb for unauthorized access is to assume that staff members are responsible people, and will, in general, behave in an appropriate manner. Truly sensitive material should be protected to prevent temptation, but minimal security measures are sufficient for most purposes.

ACCIDENTAL DAMAGE

The most prevalent security problem is accidental damage to software or data. All shops have experienced some form of accidental damage, and we include under this heading anything from the program bug to purging the wrong file. Each of us has experienced the joy of attempting a coherent application repair at 3 a.m., only to find in the morning that our fix has gone awry. Yet completely eliminating access to the production data can be a cure worse than the disease, as production program inconsistencies wreak havoc in our data bases, while we sit helpless to correct the problem.

Our first goal in handling accidental damage is to admit not only the possibility but the probability of error. For each application area, we must assess the extent of damage that is, if not acceptable, at least tolerable. How late in the day can the users be allowed access to their system before a true crisis sets in? How far back can the user recreate his input? Can the user survive if the nightly batch jobs were not run? Until these questions are answered, we cannot truly assess our risk, and apply the correct amount of security.

Our second step is to codify the types of damage that can occur accidentally. These usually break down into the following categories: Loss or corruption of production program, Loss of input data, Corruption of input data, Corruption of Database, and Loss of Database.

Most security problems relating to production programs occur while attempting to correct other types of problems. The best solution to this type of occurrence is the establishment of a separate operations group responsible for production turnover. This group could also double as documentation librarian and production problem support, and is an excellent way of apprenticing new programmers. Production turnover procedures must be rigorous, with write access to the production program group denied to all but the staff responsible. If the shop is too small to warrant a staff for this purpose, the system manager should take on this responsibility. Even in a shop with one or two programmers, the underlying security should be put in place as if a production turnover staff existed. Moving production programs (as well as JCL, standard copybooks, etc.) should be accomplished by preexisting job streams, parametrically modified to the task at hand and capable of creating an archive version of both source and executable program files. An audit trail of some sort, either a listing that is filed, or a file that is extended, can also be created by this stream. Proper program security includes the ability to easily back out any program change, in addition to identifying what was changed. Proper security also includes being able to easily identify the current program version, through compile date and/or version number prominently displayed each time the program is executed.

Preventing loss of input files, on the other hand, can be a nightmare that no amount of security will prevent. Identification of input is essential in safeguarding a production environment. All

programs that create data that is input to other programs must create an audit count of at least the number of records passed. In addition, some other form of hash or logical count should be instituted. These counts can be displayed on control reports that are filed, or added to an audit file along with a time/date/program stamp. Online input should be logged somewhere, with an easily accessible way of determining if the data logged has been applied to the production environment in case of system or program failure. Just supplying a user with a transaction log in the event of a crash can save hours of manual labor trying to recover a day's input. If the integrity of the system warrants it, transaction backups can be taken anywhere from hourly to daily. Here again, we must weigh the cost of the loss against the amount of security to apply.

Corruption of data bases or input data is usually (though not always) accomplished by a program bug. A perfect program of more than a hundred lines has yet to be written, and no test procedure can effectively test all possible occurrences. All systems should have a method of correcting bad data within the system using builtin safeguards and audit trails, but all programs should also be perfect. The time will come when the input data must be tweaked, and good security must allow it. It is far more important that changes to production data by other than production programs be identified than be prevented. If it is too difficult to change the data within the security system, it will be changed outside of it, and any audit trail will be irretrievably lost. The minimum audit trail must be to log the access to the production data. The minimum security must require the

bad data be backed up prior to being tweaked, for even bad data is better than none at all. If successful, the change should be signed off by someone other than the implementor, preferably the user.

Loss of data base through hardware or software can not be prevented. The only possible security for this type of problem is regular backups and transaction logging. Logging can be implemented through programming on files not managed by a Data Base Management System, and all systems that use such files should be evaluated for the necessity of such measures.

Loss of a data base through human error can usually be prevented through the use of appropriate access security. Privileged capabilities should be restricted to those for whom it is necessary. Even these users should have both privileged and non privileged access, and use the former only when necessary. Privileged capabilities should never be restricted to a single user id shared amongst those who need it, since this eliminates a major audit trail. Each user who may need privileged access should have his or her own user id, the use of which may be audited easily.

Our last step is the logging and analysis of all accidental damage to the production environment. Only by examining the pattern of past errors can we improve our security. We must restrain ourselves, however, from implementing random security measures in response to any particular event. Security must be established as a coherent structure of policies and procedures, not a haphazard collection of unrelated actions.

MALICIOUS DESTRUCTION

Rex Stout once commented that it was impossible to prevent a determined murderer. It is not difficult, however, to make certain he is caught. The same logic of course applies to most business data processing environments. One cannot secure one's system from those whose responsibility it is to ensure timely and accurate service to one's users. One cannot secure one's system from the user who must update it. Any attempt to place severe enough restrictions to actually protect may result in a morale problem deep enough to precipitate the very acts one is trying to protect against. The two major efforts in this area must be: first, to establish a secure enough environment to enable reasonable detection of sabatoge; and second, to isolate the perpetrator.

Securing the environment involves many of the steps outlined for accidental damage. A separate staff responsible for turnovers, sufficient backup procedures, including offsite retention of files back far enough to cover most contingencies, and separate production libraries are all good measures to help secure the environment. Other common sense items include removing access to hardware and software before an employee is informed of his termination, giving two weeks pay in lieu of notice IN ADDITION to any severance due, and enforcing password changes at reasonable intervals (two months is adequate). Creating an atmosphere where the majority of employees feel they have been reasonably well treated is probably the best safeguard. Allowing

employees to feel that they are professionals, even when terminating them, will also help foster professional conduct.

Isolation of the perpetrator can be enhanced immeasurably by assigning sole responsibility for each area to different staff members. All senior and intermediate staff should have an area of total responsibility, and should be held accountable for knowing the current state of their area, including recent problems and changes. This provides a single area to audit on employee termination, either by the corporation, or by the employee's own decision.

In general, though, one cannot run one's business in the fear of sabatoge by disgruntled employees. Though stories of computer crime fill the newspapers, very very few employees actually resort to such tactics. Most staff are professional, and even if they are not, the effect on one's career of being discovered is enough to discourage even the most foolhardy. Only in anger will these employees attempt to damage your installation, and proper management is more important in preventing this form of sabatoge than any security implementation.

After delineating the perils that would affect your installation, it's time to analyze the extent of the security necessary for your installation.

First, isolate global perils, those elements that could bring your complete operation to a halt. The responsibility for these elements reside with your system manager and technical staff. Securing the system from these people is extremely counterproductive and probably impossible anyway. Establish instead a monitoring procedure to protect the system, regular backups to recover with, and 'walk throughs' of all system changes amongst the responsible parties. New releases of vendor software should be publicized prior to installation, and copies of the prior release should be available to return to. Never let a vendor, including HP, IBM, etc. install software without first, explaining the new release and installation procedures to you; second, explaining the backout procedures to you; third, allowing you to do a complete backup of your current system; and fourth, allowing you to talk to another installed site.

Second, identify your essential time critical applications, that is, those applications that MUST (not should) be completed or online at a particular time, or your operation goes down the tubes. These can include daily payrolls, order picking tickets, point of sale applications, etc. Determine the types of perils that these

applications are subject to. Establish the minimum time it would take to recover from each type of peril and determine if a manual backup can be designed for complete disasters. Determine the minimum security necessary to protect against the majority of these perils and implement it on an application specific basis. This may involve programming the security into the application itself. Remember, these applications MUST be available at a particular time, and so you have no real choice in whether or not to implement security here.

Third, identify those essential applications that are not time specific, that is, they must be done but you have some leeway in recovery. Security here can be looser than in the previous categories, but these are still essential systems. With this category, you must determine the maximum time that you can live without each application, that is, how many hours or days have you to fix any problem. Application considerations for this category should be geared toward the auditability and recovery of data, as opposed to stringent internal application security.

Fourth, group those applications that should be run, but are non-essential or can be produced at a later time. Many user reports fall into this category. Some complete applications may fall here. Usually, standard system security is sufficient to safeguard these applications.

And if you have any applications in the fifth category, that is those applications that do not need to run at all, why are you still running them?

In conclusion, by applying security at the application level, in response to the perceived need of the application itself, you will find that you will need less security than if you try to apply security to your system as a whole. I will leave you with the following olio of security guidelines.

* When possible, allow users to assign their own passwords.

* If you are assigning passwords, do not make them overly complex, or someone will tape them to the terminal.

* Try not to require numerous different passwords. Use the user id to ascertain access.

* It is more important to log changes to your environment than to prevent them.

* It is more important to provide a tool to the user, than it is to protect him from the consequences of that tool. This does not relieve you of the responsibility of explaining those consequences to him, but he's a big boy, and should be allowed to make up his own mind.

* When in doubt, back it up.

* When in doubt, log it, count it, apply it ... but back it up first.

* Security should not be painful. If it is, you're doing it wrong.

* And finally, your employees' morale is the best security in any environment.