CHAPTER 10: SECURITY

What MOS's Security Can Control

The security feature of PC-MOS provides access control to three types of system entities: files, directories, and partitions. Through proper configuration, a computer system may be set up so that the users can operate with no concern of others interfering with their work, either through intentional tampering or by accidental means.

Individual files can be secured so that they are effectively invisible to all but those with the appropriate clearance. When security access control is applied to a directory, all files and sub- directories within that directory are automatically placed under access control. Thus it is not necessary to apply security to each of the files in that directory. The entire disk storage system may also be secured through a feature known as the master password.

Access control to workstations may be achieved by including the SIGNON command in the startup batch file for a workstation task. This is important in the case where a locally connected workstation is placed in a public location, and especially important for workstation tasks which are connected to a modem for dial-up access by remote workstations. MOS's security system can also be used to control which tasks the user will be able to PAMswitch to using Alt-Numberpad entries or the SWITCH command.

Physical Access Control

While the logical access controls built into PC-MOS are quite valuable, when security is an issue, you must not overlook physical access control. Your computer system should be kept in a secured location, along with printers which will be used for sensitive documents and all connection boxes for the ports used by workstations. Since someone can bypass security by booting your host from their own MOS floppy on which they've set up their own $$USER.SYS file, controlling physical access to the server is important. It is also important to keep the master disks and backup disks in a secured location.

You should also note that using the keyswitch to lock off the master console's keyboard doesn't deactivate a mouse, so don't leave your computer in a mouse- aware application when you go out for lunch. Unplugging a mouse is not recommended since this can create noise spikes which will upset the computer. There would also be nothing to stop someone from attaching their own mouse to your port, so the best solution is to exit the mouse-based application altogether.

Access Levels

MOS controls access to secured entities through a system of 4 access levels, numbered 0 - 3, and 26 classes, denoted by the letters A - Z. Secured entities are assigned class letters and signed on users are assigned certain access privileges to each security class. Access levels will be discussed first.

When a user has level 0 access to a file, directory, or partition, this means they have no access at all. Such entities will be effectively invisible to them, almost as if they did not exist. Although the MOS MAP command will list all tasks in a system, users with level 0 access to a task will not be able to PAMswitch into it or remove it.

Regarding secured files and directories to which a user has level 0 access, using DIR will show nothing. Even if a user happens to know the name of a secured file or directory, commands such as TYPE and CD will act as if the item doesn't exist. The only exception to this is if you try to create a file with the same name as a secured file which happens to be within your current directory. Since it would not be acceptable for anyone but an authorized user to overwrite an existing secured file, such an attempt would result in a file creation error message. This rule applies to directories as well.

By assigning an access level of 1, a system administrator may designate certain files as execute-only. This can be used to prevent unauthorized copying or modifying of software packages which are installed in your system. The operating system will be able to load such files into memory to execute the code contained in them but no other type of read or write operation will be permitted.

When applying this type of protection to a group of files which comprise a software package, you must determine which files in the package are program code modules and which are data files. If security is configured so that an application's data files have an execute-only type of access, the application will not be able to function properly.

Also, certain packages need to be able to modify their own code files when their setup menu is used to select default operating parameters (e.g. screen colors, margins, etc.).

If all the modules containing executable code can be isolated into a directory by themselves then you can set this directory's class to one which the system's users have an execute-only access level. Full read/write access to the directory where the data files are kept would be required. This relies on the DBMS supporting this arrangement. Some applications will be designed to manually open and read in files known as program code overlay modules. If such files are classified as execute-only, the application will not be able to load them.

Level 2 access provides read-only and execute-only privileges. Each higher access level adds on to the previous, so don't think that the only use of level 2 is for files which fit in both the "read-only" and "execute-only" categories. Level 2 could be useful when you need to maintain records for other users to access but must control who is allowed to make updates. This could be applied to files in a database type of application as long as you can precisely determine which database operations require write access and which don't.

Files to which a user has access level 2 may not be deleted or changed in any way. In the case of a directory, all files within that directory are auto- matically placed under level 2 access control, except for any individual files to which a user would only have a level 1 or 0 access. Also, note that the access level is an attribute which is specific to the user and not the file. Another user with level 3 access to a file, could modify or delete it, while a user with level 2 access could only read that file.

Level 3 access imposes no restrictions on the operations which may be done to a file, directory or partition. Typically, a system is configured so that the system administrator has level 3 access to all entities and the other users only have this full access to certain subgroups of entities. MOS determines the access level each user has to each secured entity through the entity's class.

Security Classes and $$USER.SYS

To become secured, a system entity has a class letter from A to Z assigned to it. Upon signing on to MOS's security system, a user is assigned an access level to each of the 26 possible security classes. The access level associated with a given class letter determines the interaction a user may have with entities of that class. The ways in which a securable entity gets a class assigned to it will be covered later.

Through a file named $$USER.SYS, the access level each user has to each of the 26 possible classes is defined. This file is basically a translation table which charts the relationships between users, classes, and access levels. See figure 10-1 for an illustration of this two-stage relationship and a sample $$USER.SYS file. It is up to the system administrator to assign the appropriate classes to system entities and to construct a $$USER.SYS file which properly completes the associative process.

Given:

user: JOHN ——→ has ——→ level 3 access to class A

and

file: XYZ.DAT ——→ has ——→ class A

Then:

user: JOHN ——→ has ——→ level 3 access to file: XYZ.DAT

JOHN's entry in the $$USER.SYS file might look like:

```
*USER     PASSWD      ABCDEFGHIJKLMNOPQRSTUVWXYZ c
*----     ------      --------------------------- -
JOHN,     TRITE,      32100000000000000000000000,A
```

Figure 10-1

The two stage relationship of MOS's security.

Given the file XYZ.DAT with a class of A, someone signed on to the user ID "JOHN" will have unrestricted access to this file since there is a 3 in the class A column of the $$USER.SYS file for user JOHN. As can be seen in JOHN's $$USER.SYS definition line, this user would be able to read or execute any files with a class of B, only execute files of class C, and be prevented from any interaction with files with classes D-Z. Note that a user signed on as JOHN could copy a class B file to a file of a new name (or into a different directory) using class A for the target file, and then modify the copy of the file.

Space Class

When you create files in a system in which security is not being used, these files have what is referred to as a "space" class. In other words, non-secured files have an null value of class so there is no way to associate any access level restrictions to them. All users have unrestricted access to entities with a space class regardless of whether they are signed on to security or not. Any files you had before installing MOS on your computer, or since installing MOS but before implementing security, will have a space class. This is also true of files in software packages you purchase.

The only case where some explicit action must be taken to assign a file a space class is when a file is currently secured and you wish to change it to a space class so that anyone can access it. The COPY command is used with a /C option to do this (explained later). Directories and partitions which have not been secured will also have a space class. When using MOS's security, you can secure only portions of your machine and leave other portions with a space class.

SIGNON and SIGNOFF

The SIGNON command is the gateway through which a user must pass to gain access to secured entities. Through the SIGNON process, a user acquires certain access levels to each security class as defined in the $$USER.SYS file. You may want to design your system so that the startup batch file for certain tasks invokes the SIGNON command after setting up the PATH, PROMPT and other system attributes.

You may also want to regulate PAMswitching so that a new user coming to a workstation which is at the SIGNON prompt cannot switch to another task without first entering a user ID and password. See the discussion of MOSADM SWITCH OFF near the end of this chapter.

Once a user has signed on, they should not leave their workstation unattended without re-invoking the SIGNON command, or issuing a SIGNOFF command. SIGNOFF could be used if it would be permissible for this unattended workstation to be used by people with no user ID and password. These users would only be able to access entities with a space class.

Note that the SIGNOFF command must never be issued in a partition with some class other than a space class. For example, if your partition has a class of B and you enter SIGNOFF, you no longer have any privilege to enter keystrokes into that partition -- the partition is effectively dead. The only thing you will be able to do is to PAMswitch to another partition, providing there is another space class partition in existence. SIGNOFF should only be used in a space class partition to cancel a previous SIGNON, so that normal space class operation may resume.

If only authorized users are to be permitted to operate a workstation, be sure that it is always resting at a SIGNON prompt. To promote good user habits and insure consistency, you could make a simple batch file, SOFF.BAT, which does a CLS and SIGNOFF and one called SON.BAT which does a SIGNON. If PAMswitching control is not provided by using secured partitions, see the discussion for using the MOSADM SWITCH OFF command near the end of this chapter.

How Classes are Initially Assigned

Another attribute which is assigned to a user through the SIGNON process is the default output class. This is the class assigned to any new files or directories which are created while a user is signed on to security. Files created with COPY CON, or through redirection (e.g. ECHO abc xyz.dat), or from an application program will inherit this value as their class letter. Applications do not have to be modified to produce secured files. Provided you have access to the files you need to run an application, MOS's security feature will be transparent to that program's file operations.

The letter used for the default output class is initially set from the right-most column of the $$USER.SYS file and is reported by the SIGNON command once the password has been entered. Refer to the $$USER.SYS file shown in figure 10-1 for an example. In addition, the CLASS.COM utility can be used to review and change the default output class. A $$USER.SYS line can be set up where access to secured files is achieved but the default output class is a space class. This means that when new files are created, they will not be secured. Leave the right-most column empty to have newly created files and directories inherit a space class.

It is also possible to set the default output class to a letter to which you have no access, although this would probably be confusing. This could be done through the $$USER.SYS definition or with CLASS.COM. Since a non-signed on user can use CLASS.COM to do this, or to change a directory which currently has a space class to some other class, it is wise to keep CLASS.COM in a directory to which only signed on users have access, or to classify the file itself.

The issue of securing MOS's system files and utilities will be covered in more detail further on in this chapter.

In a multiuser environment it doesn't do much good to implement security for files and directories if PAMswitching is not controlled. A manager and his or her staff might be sharing a computer system but if any of the staff members can PAMswitch their workstation to watch their manager's task, having security on the manger's files won't achieve full privacy. When a partition is created with the ADDTASK command, a security class may optionally be assigned to the new task which will determine which users can PAMswitch into this task. Users with level 0 access to the task's class would not be able to PAMswitch to it at all while those with an access level of 1 or 2 could PAMswitch to watch task but would not be able to enter any keystrokes except to PAMswitch back out again. Users with level 3 access will have unrestricted use of this new task.

In the case of a workstation partition, there is already a console attached to the task, so PAMswitching in is not the only way to access the task. Regardless of whether a user first enters a secured partition through PAMswitching or by using the workstation which is initially attached, it is essential that this new task's startup batch file invoke the SIGNON command. Until a user has gone through the SIGNON procedure in this task, no other keystrokes will be accepted.

Using COPY /C to Change a File's Class

When you create a new file using a statement such as COPY CON XYZ.DAT or ECHO something XYZ.DAT, this new file's date and time would be derived from the current value of the system clock. However, if you were to issue the command COPY XYZ.DAT A:, the new copy of XYZ.DAT would inherit the date and time of the original file, this being part of the standard operation of the COPY command. By the same token, if you were to create XYZ.DAT while signed on to security, this new file's class would be derived from the current value of the default output class. In addition, the command COPY XYZ.DAT A: would produce a target file with the same security class as the source regardless of the current default output class (again due to the explicit actions of COPY).

Through the use of a /C option, COPY can be made to produce a target file of a different class than that of the source. If the file XYZ.DAT is currently of class E, then the command COPY XYZ.DAT D: /CT could be used to produce a copy of the file on drive D: with a class of T. The /C option can be added to any form of the COPY command to affect the class of the target file. To remove security from a file, changing it to a space class, use the form COPY XYZ.DAT D: /C-. Also, using /C without any class letter will cause the target file to inherit your current default output class.

There is one special form of the COPY command which is only allowed when the /C option is used: COPY XYZ.DAT /C? (where the ? would be replaced with a class letter, a "-", or left blank to assume the current default output class). If this syntax were used without the /C option, the message "Copying a file onto itself not allowed" would appear.

Including /C will result in the conversion of the file from its present class to the new one specified. This is much more convenient than having to copy a file off to another temporary name and then delete the original and rename the temporary file back to the original name.

An additional special capability of COPY /C is the ability to copy a file to a class to which you currently have no access. For example, user JOHN, with no access to class E, could enter COPY XYZ.DAT /CE or COPY XYZ.DAT NEWXYZ /CE and produce a target file which would be invisible to JOHN. This feature could be useful when office staff members need to forward information to their manager or when students are sending their work to their instructor (like sliding a written report under your professor's locked office door). One difference between this case and copying a file within your scope of control, where overwriting is done automatically, is that when you are copying a file to a class you have no business viewing, you have no business overwriting existing files.

Using CLASS.COM

When you create a new directory, your current default output class is assigned to the new entity, just as when a new file is created. However, when you first apply security to a machine and want to secure parts of an existing directory structure, you need a way to assign a class to a directory which is already in existence. The CLASS.COM utility can be used to accomplish this. In addition, CLASS.COM can be used to change the class of the current partition or the current default output class. Refer to the PC-MOS User Guide for more details.

One important thing to note about CLASS.COM is that it must never be used to change the class of your current partition to a value to which you don't currently have any privileges. Doing so effectively kills the task since the only keystrokes that will be accepted from that point on are Alt-numberpad entries for PAMswitching. However, you will only be able to PAMswitch out of this dead task if some other task exists to which you have access privileges.

Securing the System Files

When installing MOS's security feature, you must, of course, secure the $$USER.SYS file. In addition, since the CONFIG.SYS file can be used to specify the location of the $$USER.SYS FILE, through the USERFILE= statement, it must also have a non-space security class. If CONFIG.SYS were not secured, anyone could put in their own USERFILE statement, specifying their own version of $$USER.SYS, and gain level 3 access to all classes.

The files MOSADM.COM and CLASS.COM should also be changed to a secured class to prevent unauthorized users from changing the behavior of the system. The MOSADM utility can be used to change a task's priority and slice, and to control PAMswitching access and the caching subsystem. Allowing a space class user to experiment with CLASS.COM could easily result in confusion, and is best prevented before it occurs. Note that you must remember that these files are secured when installing a new revision of MOS.

Secured Printing

If you have a printer which will be used to output financial records or other sensitive information you'll obviously want to physically secure the printer (e.g. place it in a private office). You will also want to prevent unauthorized users from looking at the print files waiting to be processed by PRINT.COM. The simplest approach is to use a dedicated spool directory and a dedicated PRINT.COM partition. You can then secure both items to produce and manage print files which would be invisible to anyone without the proper access level.

Print spooling involves two file accesses to the spool directory. The background processes of the SPOOL.COM program must have access to the spool directory when print output is redirected to a spool file, and the PRINT.COM program must have access to spool files if it is to copy their contents to a physical printer.

Securing a spool directory means that the user who sends print output to this directory must have gone through the SIGNON process in order to have access to the class of the spool directory. Likewise, the PRINT.COM partition must be taken through SIGNON when the system is initially booted up. This means that the system administrator, or their designate, must be present each time the system is booted up in order to activate a secured spooling subsystem.

Since operating a secured print spooling subsystem requires a manual signon of the PRINT.COM partition, the fewer secured print spoolers you must maintain, the better. If only one task will be generating print output of a sensitive nature, the simplest approach may be to connect a dedicated printer for this task which is kept in a secured location. Then there is no need to go through a signon process for the PRINT.COM partition.

The above approach addresses the case where the print data is sensitive and must be kept out of the reach of unauthorized browsers. Another situation which can exist is the where a user is signed on to MOS's security and is producing spooled printer output, but they don't need to send it to a secured PRINT.COM partition. This could be the case when signing on is necessary to access or modify a data base but the printed reports are not sensitive and you'd rather not have to bother with signing on a secured PRINT.COM partition each time the system boots up.

Release 4.00 of MOS provides a solution to this problem through a new option to the SPOOL.COM program, the /S option. It is now possible to cause the SPOOL.COM TSR to change the security status (the class) of the spool files it creates. This new class does, of course, have to be one to which the user currently has a write access privilege. In the simplest case, you could use the form: "/S" (a slash and an S with no other character) to have your spool files produced with a space class. In this way, a user can operate from within a secured task but send printer output to a non-secured PRINT.COM partition.

Securing Remote Tasks

If the your only need for security is to prevent unauthorized users from accessing a workstation partition connected to a dial up phone line through a modem, you could create a $$USER.SYS entry such as the following:

```
*USER     PASSWD      ABCDEFGHIJKLMNOPQRSTUVWXYZ c
*-----    ------      -------------------------- -
JOHN,     TRITE,      00000000000000000000000000
```

The access levels used for classes A through Z are meaningless if you will never have any secured files, directories, or partitions. In the case of a remote workstation task using the SIGNON command, the partition is actually "secured" but it was not given a class through the optional ADDTASK parameter.

Unless explicit action is taken when the SIGNON command is invoked in a partition, PAMswitching will still be enabled. This would allow a user to bypass the SIGNON prompt by PAMswitching to another task. Preventing a PAMswitch was not made an implicit part of the signon command to provide the flexibility of selective control. In order to prevent someone from connecting to this partition's modem and simply PAMswitching away to another task, the startup batch file for this partition would use the MOSADM SWITCH command as follows:

```
MOSADM SWITCH OFF
SIGNON
MOSADM SWITCH ON
```

Having PAMswitching disabled while a modem based task is resting at the SIGNON prompt means that other consoles which are directly connected to the host cannot switch into that task. MOSADM SWITCH OFF prevents switching out of the task as well as into it. If it is necessary to access this task for some reason, use the MOSADM SWITCH ON n form of the command where n is the number of the task. When an explicit task number is specified, the PAMswitching state of one task may be controlled from another. Once you have finished, be sure to return the task to its secured state.

In summary, there are two ways of providing PAMswitch control -- through the MOSADM SWITCH ON | OFF command and through assigning a security class to all partitions. Although the method of having all partitions assigned a security class could be used, this is a less flexible approach. Note that CLASS.COM would need to be used to give the foreground partition a class for this to work. Given that the MOSADM.COM utility should be secured in a system where MOS's security feature is to be used for more than controlling remote workstation access, these two types of PAMswitching controls are mutually exclusive. In the simpler case shown above, MOSADM.COM must be available before the SIGNON process has been completed.

Beginning with release 4.00, a feature was added which can be very useful with respect to securing remote tasks. For serial terminal workstations which are remotely connected with a modem, it is possible to automatically invoke a task restart when the modem indicates a loss of carrier. This option is controlled on the parameter line of the serial driver. Consult the documentation for your specific serial driver for details.

There are two primary reasons for using this feature. First, in the event your on-line session is interrupted by a phone line failure, having the remote task automatically restarted will insure that the modem on the remote end is restored to its auto-answer state. This is typically taken care of by invoking the MODEM.COM utility from within the task's startup batch file. The second reason for configuring a remote task to automatically reboot when a loss of carrier is detected is to insure that an abnormal disconnection doesn't leave your remote task open for anyone else. Having the task restarted will insure that the startup batch file will re-issue the SIGNON prompt.

One consequence of using this feature is that you could not call up the computer at your office from home, start a long job of some sort and then disconnect. As soon as you disconnect, the task would be re-started and your long job would be terminated. If you find that you want to use the auto-restart on carrier drop feature but also need to be able to start long jobs and then disconnect, start the job in another task besides the one you call in on. The RJE technique discussed in Chapter 5 may be helpful in such a situation.

Tips and Caveats

When using security, you must be careful when copying files off to a floppy diskette to send off to some other user. If your default output class is something other than space class, the file created on the floppy disk will be invisible unless the other user is also using security and has the appropriate access to the file's class. You can use COPY /C- to insure the target file has a space class.

A simple way to approach security is to classify directories and partitions as needed but use a default output class of space. This provides privacy without having to worry about class to class translation in file transfers.

When one user wants to transfer a file to another within the same system and each user is working in a directory which is inaccessible to the other, an intermediate directory must be used. The directory chosen would be one to which both users have access. This method requires the sender to copy the file to the intermediate directory and the recipient to copy it from that directory to their own. Providing the file's class is space or some value that both users have access to, the REN/M command will be more efficient than COPY since it re-assigns the file's directory entry rather than copy its entire contents from one place to another on the hard disk. See the discussion of MOVETO.BAT in Chapter 4.

If this type of file transfer will be a common operation, create a batch file to automate it. If unattended operation is required on the recipient's part, an RJE partition could be setup to watch for and collect files in the intermediate directory. Note that a SIGNON would be required for this RJE task just as with a secured PRINT partition. See Chapter 4 for details on the RJE technique. If you don't want this transfer scheme to be vulnerable to access by others, secure the intermediate directory so that only the two users involved have access to it, or have the user's assign a security class to the file while it is in the intermediate directory.

One final tip: use the $U operand in the PROMPT string to include the user name in your command prompt. This will provide a reminder that you are signed on, and under what ID. Also, entering CLASS ** will report your current default output class and the class of your partition.

Case Study - Designing a Security Configuration

In this last section, we will examine a typical office situation which includes a manager (who also serves as the system administrator), a secretary, and a number of inventory clerks. When planning for security, first list the users and entities to control in an access chart such as in figure 10-2.

	Office Manager	General Secretary	Inventory Clerks	Class
System files	yes	no	no	A
Accounting DBMS	yes	no	no	B
Inventory DBMS	yes	yes	yes	C
Correspondence	yes	yes	no	D
Spool1	yes	no	no	E
Spool2	yes	yes	yes	space
Spool3	yes	yes	yes	space

Figure 10-2

A Security Planning Chart

A $$USER.SYS file for this configuration could be:

```
*USER     PASSWD      ABCDEFGHIJKLMNOPQRSTUVWXYZ c
*----     ------      -------------------------- -
ofmg,     bluto,      33333333333333333333333333,B
sect,     singer,     00330000000000000000000000,D
iclk,     fetch,      00300000000000000000000000,C
```

The first aspect of a security configuration to be considered will be at the directory level. After directory level security has been established, at least on paper, you can more easily determine any particular class assignments which might be needed for individual files. Once the access to each entity for each user has been decided, the class column can be filled in. For any entities which may be accessed by all users, a space class can be used. An exception to this would be if protection from casual access by others is required.

Regarding partition classes; although other class letters could be used, those established for directories will often suffice. In this example, the office manager's partition would be assigned class A, the secretary would use class D and the inventory clerks would use class C. This applies for their main working partition as well as any background partitions each user may own. The entities involved in this example are as follows:

System Files

This should always be considered when classifying access to system entities. Such files include the $$USER.SYS FILE, MOSADM.COM, etc.

Accounting DBMS

Private to the office manager. Although the access to this entity is the same as for the system files (e.g. office manager access only) a separate class letter was chosen for flexibility. If at some point the office manager decides to have the secretary, or a new assistant, manage the financial records, having a separate class makes the security access modification simpler.

Inventory DBMS

Inventory clerks post transactions, the office manager prints reports, and the secretary fills in for inventory clerks when necessary. If access to reports needs to be controlled -- isolate the report generation module in a class A directory. If it is an integrated DBMS package, you will have to rely on the security controls in the package, if any. It is secured to prevent casual visitors to the warehouse from tinkering.

Correspondence

Word processing is done by the secretary.

Spool1

Operates two printers in the office manager's office. One is kept loaded with checks, and the other is used to print inventory and accounting reports.

Spool2

Operates a printer which is loaded with a pull from stock request form. These reports are generated by the office manager. The secretary can also fill in on this function. Although it is not intended for the inventory clerks to generate these forms, they are not restricted from accessing the spool directory since they are the recipients of these forms anyway.

If not allowing an inventory clerk to generate a pull-from-stock order is critical, use an inventory DBMS which supports security, or one with the appropriate split of separate modules so the inventory clerks can enter transactions completed and do other maintenance, but not generate pull-from-stock orders.

Spool3

Operates a general purpose printer.