Malcolm Bull
Training and Consultancy Publications

MB-Guide to

Security

Malcolm Bull

MB-Guide to Security

MB-Guide

to

Security

by

Malcolm Bull

MALCOLM BULL Training and Consultancy Services

ISBN: 1 873283 14 8

Introduction

This *MB-Guide to security* is produced for those who need a quick introduction to the security aspects of the Pick operating system.

This *MB-Guide* contains:

1) a look at those parts of the operating system which require security protection.

2) a look at those features of the operating system which offer security protection.

3) some general tips which are of interest to the System Manager, or others who are concerned with security.

You may find the following titles in the *MB-Guide* series useful in conjunction with the present volume:

Basic programming
Creating and using Procs
File design
File-save and file-restore
Files: monitoring and sizing
Group format errors
Operations and systems management
Program design
Using backing storage
The system debugger

This *MB-Guide* is not intended to present a complete description of the subject but merely to place it in context and give the reader enough information to use the facilities and to survive.

Best use can be made of this *MB-Guide* if it is read in conjunction with the reference literature which is provided for your system. You should amend your copy of this guide so that it accurately reflects the situation and the commands which are used on the implementation which you are using. By doing this, your *MB-Guide* will become a working document that you can use in your daily work.

I hope that you enjoy reading and using this *MB-Guide* and the others in the series. A list of the current titles is given at the end of this guide.

Malcolm Bull

1 What do you mean by security?

A computer installation which uses the Pick operating system makes data processing a much more accessible service than the old mainframes which were used via the computer operations department; a terminal on every desk represents an entry into the computer system. The fact that some of these terminals may be at remote sites or in public areas of the office means that we need to protect these entries into the computer system. Furthermore, the fact that - unlike a personal computer system with its own diskettes and its own data - all these terminals access the same database means that any damage which can be done may have far-reaching consequences. So security must be an important part of any Pick installation.

What do we mean by security? Security means different things to different people:

? You may want prevent unauthorised people - visitors, untrained staff and others - logging on to the system.

? You may want to keep a close check on who is using the system at any one time.

? You may need to know who is doing what.

? You may need to know who has been logged on the computer during the night, over the weekend.

? You may need to know who has been doing what.

? You may need to control the work which the authorised users are allowed to do.

? You may need to control the files which the authorised users are allowed to handle.

? You may need to prevent unauthorised users looking at the files.

? You may need to prevent unauthorised users changing the contents of the files.

? You may need to identify who has been changing which files.

? You may wish to avoid simultaneous updates on a busy file.

? You may need to know what changes have been made to the files.

? You may need to know whether data have been lost or corrupted.

? You may need to be able to recover data which has been lost or corrupted.

? You may need to know how to recover if the machine goes down.

These problems are not specific to Pick. Some of the solutions are. In this *MB-Guide* we look at these solutions.

2 Account security

We discuss general aspects of accounts and account definition items below:

* non-loggable accounts, that is, those accounts which are used by the system but which are not available for logging on.

* passwords,

* system privileges,

There are, however, a number of specific points.

2.1 Account names

Account names can be chosen arbitrarily, provided that they do not conflict with the standard account names. Choosing suitably-obscure names might afford some protection to your accounts.

However, account names should be memorable (so that the users don't have to keep checking what they are supposed to type in), but they should not be predictable (lest an unauthorised user attempts to break into the system).

An account name may contain any characters (other than the system delimiters), and the name is case-sensitive, that is

```
FRED
```

is quite different from

```
FrEd
```

Like passwords, account names should not be written down at and users should not write down their passwords. It is a hacker's dream to find a POST-IT notes stuck to the terminal with an incongruous word - possibly an account name - written on it.

2.2 Account protection

Accounts may be protected from logging on in a number of ways:

* by setting the system privileges to null, thereby rendering the account completely inaccessible at log on.

* by having a password imposed, thereby restricting access to the account,

We discuss these points later.

Accounts may be protected from other users accessing their files in a number of ways:

* by setting lock-codes on the account definition item (on the SYSTEM) file, thereby protecting the MD and all the accounts files,

* by setting lock-codes on the individual files of the account.

We discuss lock codes later.

2.3 Synonym accounts

If it is required to give a user access to an account but to restrict the work which he/she can do, then a synonym account may be the solution.

A synonym account offers an alternative logon code for an account, and the synonym account may have its own security, that is, system privileges, password and lock-codes, and any of these may be different from those of the main account. An account may have any number of synonym accounts, each with its own security.

Thus, if we have an account called PAYROLL which owns several files, we can create synonyms called WAGES, W.ADMIN and so on for that account. Anyone logging on to the PAYROLL account (the main account) may have SYS2 privileges and lock-codes to access all of the account's file; anyone logging on to the WAGES synonym may have SYS0 privileges and lock-codes to access only a few of the account's files, whilst anyone logging on to the W.ADMIN synonym may have SYS1 privileges and lock-codes to access another subset of the account's files. Their respective account-definition items on the SYSTEM file might look like this:

Name:	PAYROLL	WAGES	W.ADMIN
Attribute 1)	D	Q	Q
Attribute 2)	12345	PAYROLL	PAYROLL
Attribute 3)	29		
Attribute 4)	1		
Attribute 5)	A]B]C]D]E]F]G]H	A]B]C	A]F]G]B
Attribute 6)	A]B]C]D]E]F]G]H	D]E]F]G	A]B]F]G
Attribute 7)	E8D8F8A8	1A2D3F4E	B1A2EED4
Attribute 8)	SYS2	SYS0	SYS1
Attribute 9)	U	RU	RU
Attribute 10)	10	10	10

2.4 SYSTEM file

The SYSTEM file (or the equivalent MDS and USERS files on Advanced Pick implementations) should be protected from as many users as possible.

It should be protected by any/all of the means described described. It is necessary to protect the file from logging on since it is itself defined on the SYSTEM file by an account definition item, and is therefore recognised as an account.

Wherever possible, there should be lock-codes imposed on the file.

2.5 SYSPROG account

The SYSPROG account (or the equivalent DM account on Advanced Pick implementations) should be protected from as many users as possible.

It should be protected by any/all of the means described above.

Wherever possible, there should be lock-codes imposed on the files which are defined on the SYSPROG account.

3 System privileges

System privileges restrict the action which a user can perform, irrespective of the verbs which are on the MD of the account.

The privileges are established when the account is created and are held in attribute 8 of the account definition item.

As an account is being created, the user is asked to specify the required system privileges for the new account. This will be any of those shown below. The CREATE-ACCOUNT utility uses this information immediately to determine the contents of the MD for the new account.

The possibilities are:

SYS0 is the lowest level of security and allows the account to use only the P, G, END and OFF debug commands and any other features except those specific to SYS1 or SYS2.

SYS1 allowing the account to edit the master dictionary, and to perform magnetic-tape operations and any other features except those specific to SYS2.

SYS2 offering all Pick facilities, including file-save and file-restore operations, use of the assembler, use of the DUMP processor, use of any debug command.

SYS3 is found on some implementations and is only available for the SYSPROG account.

and allow the System Manager to restrict the general level of activity which users of that account may perform. Since there are only three (at the most four) such levels, the distinction between them is fairly coarse.

In general, each account should be given the lowest possible security level.

If a user attempt to perform an action for which the account's privilege level is inadequate, then the message:

YOUR SYSTEM PRIVILEGE LEVEL IS NOT SUFFICIENT

is displayed and the operation will be abandoned.

If attribute 8 of the account definition item is set to null, then it will not be possible to log on to the account.

McDonnell Douglas Reality implementations also hold the **additional workspace assignment** in this attribute in a form such as (20).

3.1 Privileged accounts

The SYSPROG account is the most important user-account on a Pick system. It is, however, only privileged insofar as the MD contains a number of verbs (for example, CREATE-ACCOUNT and DELETE-ACCOUNT) which are not available on the MD of the other accounts. Very few system activities actually check that the user is logged on to the SYSPROG account.

3.2 Privileged ports

Many system activities (such as POWER-OFF and REBOOT) may only be executed from port 0. For this reason, the system monitor terminal (port 0) is usually located near the CPU.

4 Logging on

There are several points which can be raised in connection with the security at logging on:

* some thought should be given to the choice of account names, as described above.

* always use passwords. We discuss passwords later.

* where it is available, use the B code in attribute 9 of the account definition item to inhibit the break key after logging on.

* for all non-loggable accounts (we discuss these below) assign a password or, better still, set attribute 8 of the account definition item (the system privileges) to null. This will return the message

  USER ID?

  if anyone attempts to log on to the account.

Whilst no computer system can be 100% safe, any hindrance that you can place in the path of unauthorised access should

be used.

4.1 Logon Procs and closed systems

When a user logs on to an account, then after validating the account-name and password, and before passing to TCL level, the operating system will scan the M/DICT for a Proc which has the same name as the account. For example, when logging on to an account called WAGES, a Proc called WAGES on the M/DICT will be invoked, if there is such a Proc. If there is no such Proc, then the user passes to TCL level. A Proc which has the same name as the account is called a **log-on Proc**.

A typical use of the log-on Proc is

* to throw the users straight into an application system,
* to ask for extra user-identification such as a secondary password,
* to set special spooler assignments or terminal characteristics, or
* to display a message to users of that account.

The first of this is possibly the most common use, with the log-on Proc calling up a standard application program so as to confine the users to this program. Such a program would guide the users, prompt them, control them, and vet and restrict the activities which they perform. In this manner, the users need not know about TCL commands or how to use the TCL language.

Nevertheless, if the users press the <BREAK> key, they may interrupt the processing and get back to TCL level. In the situation where it is required to shield the users from TCL or to prevent them from using the TCL facilities, this is clearly undesirable. However, it can be arranged that any attempt to break out in this way will return the user to the initial log-on Proc. If you organise your processing in this way, it is called a **closed system**.

The log-on Proc on the M/DICT for the WAGES account might look like this:

```
000  WAGES
001  PQ
002  HRUN WAGES.BP MASTER.MENU
003  P
004  HOFF
005  P
```

A closed system is set up by creating an account definition item on the SYSTEM file, like this:

```
000  WAGES
001  D
002  599034
```

```
003  37
004  1
005
006
007
008  SYS2
009  R
010  10
```

Note the

    R

in attribute 9. It is this R - for **re-entrant** - in attribute 9 of the account definition item which prevents users of this account from reaching TCL by breaking out of the log-on Proc. If a user does press the <BREAK> key, the normal debug prompt * or ! will appear. If the user enters P or OFF, then these will have their normal effect, but if the user enters END then control will be passed back to the log-on Proc and not to TCL.

The use of account synonyms means that each synonym will have its own logon Proc to the same account.

4.2 Further identification

Whilst the logon process does check that account names and password are correct, it does not check that that everyone logging on to an account is a valid user of that account. Application systems are used which get around this by having - as a part of the log-on Proc for each account - a standard routine which demands a further **secondary password** or other authorisation from the user, rejecting the user if any breach be detected. Such a routine might log the following information on a file:

* The name of the account which was used.
* The date and time when the user tried to log-on.
* The terminal which was used.
* The secondary password which was entered. Each person could have their own secondary password.
* Whether or not the secondary password was accepted.
* How many attempts were made before the would-be user was rejected by the system.
* What these attempts were. An inspection of this information might reveal that it was a problem with the keyboard rather than actual fraud.

During this process, the <BREAK> key should be disabled to prevent users breaking out of the vetting program and thus getting to TCL unhindered.

5 Account security - passwords

Passwords should be imposed on every account and account synonym, and these should be changed

* when personnel leave the department,

* at least once a month.

Passwords should also be imposed on *non-loggable* accounts such as

```
ACC
BLOCK-CONVERT
EXECUTE-CONTROL
PROCLIB
SYSTEM
SYSTEM-ERRORS
```

since any account can be used as a gateway into the system. It is possible to write a short program to generate random passwords and impose these on such accounts. As mentioned earlier, a far safer solution is to render such accounts inaccessible by setting attribute 8 of the account definition item (the system privileges) to null. This will return the message

```
USER ID?
```

if anyone attempts to log on to the account.

Wherever possible, the System Manager should be responsible for allocating and changing passwords for all accounts.

5.1 Choosing passwords

The choice of password is completely arbitrary. Typically, they are the names of friends, hobbies and interests, flavour of the month and so on. The more arcane, unpredictable and unlikely, the better.

A password can be of any length and may contain any keyboard characters:

* spaces,

* <CTRL> characters.

All passwords should be fairly memorable. For example,

```
DRABBLE
Plesiosaur
```

are better than

```
X123DD345
AOSJRRYEEW
```

Passwords should not be written down at any time. They

should be notified to the users by verbal means.

Users should not write down their passwords. It is a hacker's dream to find a POST-IT notes stuck to the terminal with an incongruous word written on it.

5.2 Using the password

If the user logs on by typing the account name and password at one time:

```
WAGES,pancakE
```

there is the chance that this may be seen by unauthorised spectators. For this reason, users should be encouraged (if not instructed) to logon just by entering the account name and then enter the password at the second stage when it will be invisible.

5.3 Secondary identification

When we discuss LOGON PROCS, we see that it is possible to allocate a logon Proc to all accounts which could invoke an applications program to perform further checking beyond that of the operating system itself. Such a facility could:

* ask for a secondary password,
* check that the account is valid for that terminal,
* check that it is possible for that account to be logged on at that time of day.

These checks must all be handled by applications software.

5.4 Looking at passwords

The password is held in attribute 7 of the account definition item (on the SYSTEM file).

Reality implementations hold the password in a clear, legible form, exactly as it would be typed in.

Passwords on most other implementations are held in an encrypted form. This is a string of 8 hexadecimal characters. The encryption is performed by means of the user

```
U3060
```

and there is no simple means of de-coding a password.

Since a password of any length is encrypted into 8 hexadecimal digits, there is not a one-to-one correspondence between encrypted and clear passwords, and there is no certain means of decoding a password, although it is possible to produce an acceptable password which would encrypt into any required hexadecimal string.

If (for any reason) there is a non-hexadecimal character in

the encrypted password, this will be taken to indicate the termination of the password string. If there is a non-hexadecimal character in the first position of the encrypted password, then the operating system will interpret this as there being no password on the account.

5.5 Changing passwords

Passwords are changed by one of two methods:

1) some systems offer each account a

   PASSWORD

   utility which allows any user to change the password for that account.

2) some systems offer a similar facility to the

   SYSPROG

   account only. This asks for the name of the account whose password is to be changed.

Depending upon the implementation, the PASSWORD utility may ask the user to type in the current password correctly before it may be changed.

It is important to remember that only the final 8 characters of a password are significant. So, when a password is changed, be sure that the ending is different.

Reality implementations are different in that the password can be changed simply by using the Editor to amend the contents of attribute 7 of the account definition item on the SYSTEM file.

5.6 Removing passwords

It may happen that the person who changed the password has forgotten what it was. in this case, the remedy is for the System Manager to use the editor to set attribute 7 of the account definition item to null, or to put a non-hexadecimal character in front of the password.

6 Monitoring the users

In a large installation with many terminals at many sites and in many places, there is often a need to know who is logged on to the system. This is particularly important when there are modems and external lines linking in to your computer system.

The following TCL verbs will tell you who is currently logged on to the system.

LISTU
: displays details - the account name, port-number and other information about all the users who are currently

logged on.

WHO
displays the details - port-number and account-name - of specific ports or specific accounts.

WHO *
displays details of all ports on the system.

```
 0 SYSPROG
 1 BASIC2
 2 UNKNOWN
 3 ACCOUNTING
 4 UNKNOWN
 5 UNKNOWN
 6 UNKNOWN
```

WHO 'WAGES'
displays details of all ports which are logged on to the WAGES account.

You might even want to know what the various are doing. The following TCL verbs may be of some help in this.

WHERE
displays details - the processing status and the contents of the return address stack - showing what work the users (and the spooler ports) are doing.

```
 00 0420 FF30    228.593 235.4B9
*01 0400 FF30    121.000 121.1AB
 06 04C0 BF30    170.229 170.147
```

The significance of the contents of the address stack will be described in the documentation for your implementation of the operating system. In the above example

* Frames 228 and 235 are the Basic run-time frames, so this user is executing a Basic program.

* Frame 121 is a frame used by the WHERE routine, as you might expect since this identifies the terminal issuing the WHERE command.

* Frame 170 is a spooler frame. Line 6 is, in fact, the spooler.

WHERE 'SYSPROG'
will display details of all ports which are logged on to the SYSPROG account.

WHERE 1
will display details of port number 1.

WHERE Z
will display details of all ports on the system, whether they are in use or not.

```
 00 0420 FF30    231.05B
*01 0400 FF30    121.000 121.1AB
 02 0440 FD30      4.002
 03 0460 FD30      4.002
 04 0480 FD30      4.002
 05 04A0 FD30      4.002
 06 04C0 BF30    170.229 170.147
```

WHAT
displays more comprehensive information about the system, the users, and the various Basic program locks and system locks and the control tables which it uses.

Some implementations have a verb such as:

PEEK 9

which will echo the actual screen which is currently displayed at port number 9. This enables you to watch exactly what that user is doing.

Since these verbs obviously yield considerable information about the system, there may be a case for removing from the non-privileged accounts.

6.1 Logging users off

Unauthorised users may be forced off the system by means of the

LOGOFF

command. This will ask you for the number of the terminal which is to be logged off. It will then interrupt any processing at that terminal, as if the BREAK KEY had been pressed, and then log the user off.

6.2 Who has been doing what?

A very simple solution to the question of "who did what?" is available on those implementations which have an automatically-invoked TCL stacker. When the stacker is active, the last 20 TCL commands (50 on OA and unlimited on AP) are stored and can be recalled later. This implies that the last 20 (or so) TCL operations which were carried out on any terminal can be inspected.

Once a program has been invoked, a process logging system is the best way of finding out what your users did whilst a program was in execution. Pick does not have such a recording system, but packages can be written and some are commercially available to monitor the activities of users and record these on a file - or on backing storage. The

information recorded by such a routine might include:

* The account-name.
* The time and date of log-on.
* The programs which were used.
* The time at which each program was used.
* The identity and location of the terminal which was used.
* The names of the files which were handled.
* The time at which the files were used.
* The item-ids of the items which were used.
* The time at which the item was accessed.
* The appearance of the item before the change.
* The appearance of the item after the change.

A fairly simple system could be produced to record, monitor and report on this information.

6.3 Viruses

Viruses can be implanted into a system by way of software and by any user logging on, whether from a local line or remote lines. Since all users have access to some of the standard files (ERRMSG and PROCLIB, for example), these can be quite easily corrupted so that when a user issues an innocent command such as LISTFILES or SET-FILE, all hell can be let loose.

Some solutions to this are:

* Only load software provided by known suppliers.

* Avoid pirate copies.

* Vet all software loaded on to the system.

* Control access to the system, particularly from remote sites and via modems.

* From time to time, reload the standard files (from the original Pick system diskettes) or from clean backup copies.

Protecting your files - lock codes

In the simplest situation, the files which **belong** to one account are inaccessible to users of other accounts. For example, a user who is logged on to the PERSONNEL account cannot look at the contents of the INVOICES file which has been created on the SALES account.

However, there are many occasions on which several accounts might wish to share information. File Q-pointers will allow users to share files in this way. This, in turn, raises further problems: how can you allow your files to be accessed by another account? How can you restrict that access? How can you stop people amending files which they are only supposed to look at?

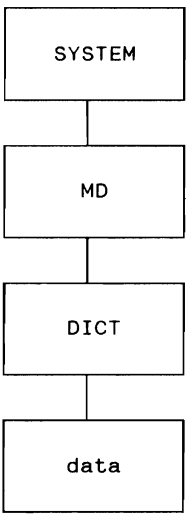
The Pick solutions to these problems are **lock-codes.**

By imposing lock-codes on your files you can stop other people looking at them or changing them. Lock-codes are held on any - or all - of:

* Your account-definition item on the SYSTEM dictionary. A lock-code here will lock all files on that account.

* The file-definition item (the D item) on the master dictionary which identifies the file which is to be locked - this will lock the DICT section of the file. A lock-code here will lock all data sections which depend upon that DICT section.

* The data-level identifier item (the D item) on the dictionary of the file - this will lock the data section of the file.



lock-codes for the MD are held on the account definition items

lock-codes for the DICT section are held on the file definition item

lock-codes for the data section are held on the data level identifier

In order to access a file owned by the WAGES account from, say, the SALES account, then the SALES account would set a Q-pointer to the required file on the SALES MD

```
     item-id: q-pointer-name
attribute 1: Q
attribute 2: WAGES
attribute 3: STATISTICS
```

In order to be able to access the file via such a Q-pointer, the SALES account must hold (in its account-definition item) matching lock-codes for (any and all) the necessary lock-codes on the route (from the SYSTEM file entry for the WAGES account) to the required file section.

A Q-pointer does not have any lock-codes.

There are two lock-codes, known symbolically as L/RET and L/UPD:

L/RET: is held in attribute 5 of the definition item and will restrict attempts to **retrieve data from the file**; this includes such activities as using Access, reading, printing, and copying from the file. Some examples are shown in the table below.

L/UPD: is held in attribute 6 of the definition item and will restrict attempts to **update the file**; this includes such activities as writing and editing the file. Some examples are shown below.

The lock-codes on the account-definition item may be multi-valued, as shown in the examples below. The lock-codes on the file-definition items are not multi-valued.

Whenever you attempt to access a file, whether it is one of your own or one belonging to another user, the lock-codes along the path to the file will be checked. Against those on the account-definition item for your account. If a mis-match is found, then the process will be terminated with a message:

```
FILE 'INVOICE' IS ACCESS PROTECTED
```

Open Architecture does not perform such checks on **your own** files.

The following table shows how the comparison is made

User's code on SYSTEM	Code on file	Result
ABC	ABC	MATCH - access allowed
ABC	AB	MATCH - access allowed
AB	ABC	NO MATCH - access prohibited
ABC]XYZ	ABC	MATCH - access allowed
XYZ]ABC	ABC	MATCH - access allowed
ABC]XYZ	AB	MATCH - access allowed
XYZ]ABC	AB	MATCH - access allowed
ABC	null	MATCH - access allowed
null	ABC	NO MATCH - access prohibited
null	null	MATCH - access allowed

The comparison is carried out from left to right, and stops when a mis-match is found (in which case the action will be aborted) or when the code on the FILE (in column 2 above) has been exhausted (in which case the action will be allowed to proceed).

On Ultimate, the lock-codes actually specify the names of the accounts which are to be allowed to access the file.

In many installations, it is the responsibility of the

SYSPROG account to assign lock-codes to accounts. Lock codes may be assigned in response to the prompts issued by the CREATE-ACCOUNT process when the account is created. Account lock-codes may subsequently be changed by editing the account-definition item on the SYSTEM file, or by a system utility.

File lock-codes may be set and changed by editing the file-definition item on the M/DICT and / or the data-level identifier item on the file dictionary.

We can illustrate this by the following examples. For simplicity, we have only shown the L/RET code and we have only locked the DICT section of the relevant files.

1) The following account definition items (held on the SYSTEM file) will lock **all** files on the WAGES account and they will be inaccessible to the SALES account:

	item-id	WAGES	SALES
	attribute 1	D	D
	attribute 2	12345	98765
	attribute 3	19	19
	attribute 4	1	1
L/RET	attribute 5	ABC	
L/UPD	attribute 6		
	attribute 7		
	attribute 8		
	attribute 9	L	L
	attribute 10	10	10

2) If we establish the following account definition items, the account SALES may access **any** of the WAGES files:

	item-id	WAGES	SALES
	attribute 1	D	D
	attribute 2	12345	98765
	attribute 3	19	19
	attribute 4	1	1
L/RET	attribute 5	ABC	ABC
L/UPD	attribute 6		
	attribute 7		
	attribute 8		
	attribute 9	L	L
	attribute 10	10	10

3) Using the following account definition items for the WAGES and SALES account and the file definition for the STOCK file on the MD of the WAGES account, the account SALES can access any of the WAGES **unlocked** files but it cannot access the STOCK file on the WAGES account:

	item-id	WAGES	SALES	STOCK
	attribute 1	D	D	D
	attribute 2	12345	98765	23456
	attribute 3	19	19	5
	attribute 4	1	1	1
L/RET	attribute 5	ABC]XYZ	ABC	XYZ
L/UPD	attribute 6			
	attribute 7			
	attribute 8			
	attribute 9	L	L	L
	attribute 10	10	10	10

4) Note that the following situation will render the WAGES account unable to access its own STOCK file:

	item-id	WAGES	STOCK
	attribute 1	D	D
	attribute 2	12345	23456
	attribute 3	19	5
	attribute 4	1	1
L/RET	attribute 5		ABC
L/UPD	attribute 6		
	attribute 7		
	attribute 8		
	attribute 9	L	L
	attribute 10	10	10

Note also that, by extension of this principle, if a **synonym** of an account has different lock-codes from the main account, then it will be prevented from accessing the files belonging to the main account.

As mentioned above, Ultimate employs a more powerful means of controlling retrieval and update locks: the CREATE-ACCOUNT verb allows you to specify (and change) a list of accounts which are to be allowed to access the MD of the account; the CREATE-FILE verb allows you to specify (and change) a list of accounts which are to be allowed to access the file. The account names are held as a set of subvalues in the appropriate attributes (5 and/or 6) of the appropriate account definition items and file definition items.

This principle suggests an alternative method, in which each account definition item has the L/RET and L/UPD lock-codes set to the name of the account, and the D-pointers on the appropriate files contain just the names of the those accounts which are permitted to access that file.

7.1 Read or Write actions?

The following table was prepared using an R83 system, an OA2 system and an Ultimate system. It shows the effect of attempting to use various functions on a set of files with combinations of L/RET and L/UPD security lock-codes.

Y indicates that, when this action was attempted with that lock-code, the lock-code **was** checked and the process **aborted** with the message saying that the FILE IS ACCESS PROTECTED.

N indicates that the lock-code was **not** needed and the process proceeded unhindered.

The + indicates that the action of the OPEN statement (not opening the file) prevented any further testing of the file-handling statements.

	R83		OA2		Ultimate	
	L/RET	L/UPD	L/RET	L/UPD	L/RET	L/UPD
LIST	Y	N	Y	N	Y	N
COUNT	Y	N	Y	N	Y	N
EDIT	Y	Y	Y	N	Y	Y
DELETE-FILE	Y	Y	Y	N	Y	Y
COPY from	Y	Y	Y	N	Y	Y
COPY to	Y	Y	Y	N	N	N
Basic OPEN	Y	N	Y	N	N	N
Basic READ	+	N	+	N	N	N
Basic WRITE	+	N	+	N	N	N
Basic DELETE	+	N	+	N	N	N

As this table shows, there are differences between the implementations, and none seems entirely satisfactory.

8 Data security

We have already seen that lock-codes can be imposed to prevent unauthorised access to accounts and/or files. Nevertheless, it may be important to know whether or not files have been changed.

A file statistics report may and should be produced after each file-save. This will list the accounts which have been saved and give statistics for each file on each account. The statistics report should be scrutinised immediately for group format errors and the appropriate action taken if any are discovered.

The file statistics report will also show the number of items and the size of each file. A comparison with previous reports will show whether an individual file has changed.

The CHECK-SUM command enables the user to generate a

hash-total for any attribute or attributes on a file. This may be used to see whether the file has undergone changes since the last CHECK-SUM was obtained. A periodic CHECK-SUM on files will quickly detect any changes to the files.

Since CHECK-SUM does not offer any means of comparing one set of check-sum statistics with the previous set, a program must be written to produce, store and compare the statistics for those files which are - or should be - static but which are important to the system.

If it is required to detect whether items have been added, changed or deleted from the file, then other statistics must be recorded to provide this information.

8.1 Data Protection Act 1984

There are also certain legal requirements concerning the security of your data. For example, if you hold details of individuals on your computer system, then you must register your installation under the Data Protection Act and you must take suitable precautions to protect your data. Details of the act and its implications and registration forms can be obtained from the Data Protection Registrar, Springfield House, Water Lane, Wilmslow, Cheshire SK9 5AX (telephone: 0625-535777).

8.2 Lost or corrupted data

There are several ways in which data may be lost or corrupted: a file or an item may be deleted by accident or a group format error may occur.

All Pick users must realise the importance of producing regular back-up copies of the system and its files. If this is done thoroughly and frequently - at least once a day - then the consequences of accidental or intentional loss of data can be minimised.

As we see in the *MB-Guide* to file-saves and file-restores, there are several verbs associated with the archiving, saving and recovery of files:

```
SAVE
FILE-SAVE
ACCOUNT-SAVE
T-DUMP

:FILES
ACCOUNT-RESTORE
SEL-RESTORE
T-LOAD
```

If back-up copies of the file are available, then they can be used to recover corrupted data.

8.3 Privileged data

Much of the standard data which is held on the system is provided for all users, and, for this reason, is available to all accounts. The standard files PROCLIB, ERRMSG and BLOCK-CONVERT are typical of such files. These are system-wide files and if items are lost or corrupted, this will affect all users. These standard files should be backed-up and verified at regular intervals. They can also be protected by imposing read-only security lock-codes on the files, thus preventing anyone from changing the contents.

Probably the most important information of the system is that held in the ABS frames. This contains the entire system software and any damage done there could be catastrophic. As we describe in the *MB-Guide to operations*, the System Manager should verify the contents of the ABS frames by means of the

VERIFY-SYSTEM

verb. R83 version 3.1 issues this command as a part of the boot-up routine and also in the PICK-SETUP account. If there are any errors, the ABS frames should be reloaded.

Note that some of the standard Pickware products, such as ACCU/PLOT, CompuSheet+ and Jet and also certain other pieces of proprietary software, load their object code into allocated areas of the ABS frames and will cause the VERIFY-SYSTEM activity to report errors where there are none.

Furthermore, if you do reload the ABS frames you will destroy the frames loaded by the CompuSheet+ and other software. In such cases, it is advisable to de-install all the software before you reload the ABS frames, otherwise you will use up all the lives on the software diskettes.

8.4 Programs and data

Protecting data records against simultaneous updates is of considerable importance in which there is high activity on certain files. For example, a ticket reservation system must be able to protect itself against the situation in which one user may wish to look at and/or change a record whilst another user is looking at the same record. Failure to do so, may result in overbooking at best and lost data at worst.

The usual solution is to use the locking facilities offered by the Pick operating system. Depending upon which implementation of the operating system you are using, it may be possible to

* lock a single record and thereby prevent a second user from handling that record whilst someone else is using it, or

* lock only the file group, thereby preventing a second user from handling any records which are in the same physical group as a record which someone else is using.

Obviously, the first of these is the better situation.

Such locking is achieved by means of the Basic statements

```
READU
MATREADU
READVU
```

which read the required record and set a lock (either on the item or the group). When the record is written back to the file, the lock is released. If a second user attempts to read a record (or from a group) which is locked, then his/her terminal will beep and the program will wait until the lock is released, at which point the read will proceed.

These forms of the READ statement have a LOCKED clause which is provided to allow the programmer to specify some action which is to be taken in the event of a record (or a group) being found to be locked. Typical actions at this point might be to announce the fact that the record is locked and then ask the user if he/she wishes to wait or to abandon the action.

The WRITEU / MATWRITEU / WRITEVU statements can be used to write a record back to the file without releasing the lock.

The programmer must take care to protect against *deadlock* or *deadly embrace* in which program A cannot proceed because program B is excluding it, whilst program B cannot proceed because program A is excluding it. The programmer must also take care to use the RELEASE statement to release any records which are READ (by one of the above forms of the READ statement) but which are not written back to the file.

8.5 Program security

A further type of program security might be that which requires to restrict the use of a particular program by only one user at a time. For example, it may be that a certain program updates a set of files, but these updates should only be run once. The Basic language has the LOCK / UNLOCK statements which can be used in such situations.

The LOCK statement has the format

```
LOCK expression {THEN statement(s) / ELSE statement(s)}
```

where **expression** evaluates to the number of the lock which is to be set: numbered 0 to 63 on most implementations, or 0 to 47 on others. Note that the THEN/ELSE clauses are optional here. Here are some examples of the statement.

```
LOCK 29

LOCK NEWLOCK

LOCK KEY ELSE PRINT KEY:' IS ALREADY LOCKED'
```

```
LOCK LOCKNO THEN GOSUB 1000 ELSE GOSUB 2000
```

These Basic program locks are arbitrarily-numbered, system-wide locks and are intended to prevent two users executing a program which sets the same lock. Since they are system-wide, there is obviously a need for some central control of the use of the locks, otherwise you may find that a stock control program inhibits the use of the wages program, simply because the respective programmers decided to use lock number 13 to prevent simultaneous use of their program.

If the lock is found to be locked by another user, then the program will wait until it is available and the terminal will continue to beep until the locked is released, or it will execute the statement(s) provided in any THEN/ELSE clause in the Basic statement.

The ELSE clause specifies the action which is to be taken if the same lock has already been locked by another user. In this case, the program will not wait for the lock to be released.

Some implementations, such as McDonnell Douglas, also offer Proc processing locks. These are applied by use of the

```
PLn
```

statement. Like the Basic program locks, they are arbitrarily-numbered, system-wide numeric locks and are provided to prevent two users performing a process which sets the same lock.

9 Tape/diskette security

There are certain points about the use of backing storage media which relate to security:

* always use a cycle of several tapes for file-saves and for account-saves.

  The traditional system employs three tapes: grandfather, father, son. This is described in the following Section.

* if it is possible, always make and verify two copies of the complete file-save tape before you use one to perform the file-restore.

  The time between initiating a file-restore and its final, successful completion is one of the most sensitive times in any System Manager's life.

Of course, the same points apply if diskettes are used as backing storage.

9.1 A cycle of file-save tapes / diskettes

It may require several diskettes or several tapes to hold a complete file-save. For the following explanation, we shall

use the term **cycle of volumes** to mean all the diskettes or all the tapes which are needed to hold each file-save dump. The principles described here apply equally to those implementations which only perform individual account-saves.

It is usually not sufficient to use the same cycle of volumes over and over again for your file-saves. It is too dangerous to put all your eggs (or data!) in one basket.

A simple solution is to have three cycles of volumes - grandfather, father, son - and use these in a cycle so if you perform a file-save at the end of each day's business, you would use the cycles like this:, like this:

```
Monday ..................... cycle 1
Tuesday .................... cycle 2
Wednesday .................. cycle 3
Thursday ................... cycle 1
Friday ..................... cycle 2
Saturday ................... cycle 3
```

If you only work a five-day week, then the pattern might look like this:

```
Week 1 : Monday ..................... cycle 1
Week 1 : Tuesday .................... cycle 2
Week 1 : Wednesday .................. cycle 3
Week 1 : Thursday ................... cycle 1
Week 1 : Friday ..................... cycle 2

Week 2 : Monday ..................... cycle 3
Week 2 : Tuesday .................... cycle 1
Week 2 : Wednesday .................. cycle 2
Week 2 : Thursday ................... cycle 3
Week 2 : Friday ..................... cycle 1
```

and so on. You might even use five tapes.

Some installations have even more complicated patterns.

Those implementations which have transaction logging - in which each amended item is flagged by the operating system - are able to save only the changed items. In this case, the cycle might be:

```
Monday ..................... transaction cycle 1
Tuesday .................... transaction cycle 2
Wednesday .................. transaction cycle 3
Thursday ................... transaction cycle 4
Friday ........................... full file-save
```

The transaction save performed on Monday evening contains all the data which changed during Monday; the transaction save performed on Tuesday evening contains all the data which changed during Monday and Tuesday; the transaction save performed on Wednesday evening contains all the data which changed during Monday, Tuesday and Wednesday, and so on. Thus, in order to recover the system at any time, it is necessary to restore last Friday's file-save tape and follow

this by the most recent transaction tape.

A common practice is to make two copies at the end of each week and at the end of each month and lodge one copy in a safe place (such as a bank or a distant office) so that, in the event of the computer site being damaged, the data will still be recoverable.

Whatever system you use, you should always label the volumes clearly to show their contents, the date and time when they were produced, and the number of each volume within the cycle.

A similar cycle of diskettes / tapes would also be used for the account-saves for each account which is saved individually.

It is possible to write a simple front-end routine before invoking the file-save to write and check a simple tape/diskette label indicating

* whether the tape contains a full file-save or, if it a single account-save, the account name.

* the number of this tape within the cycle.

Such a front-end processor may use the tape label option which precedes the file-save (and account-save) to write this information to the tape. On my own installations, I use a control file to record the following information for each account on the system:

+ the number of tapes in the account-save cycle for this account.

+ the number of the diskette containing the latest account-save in the cycle.

+ the time and date when the last account-save was produced.

+ the nature of the device, the drive and the density of the medium used for this account-save.

+ a brief description of the account.

+ whether or not the account has been used since the last account-save was performed.

+ whether or not the files of the account have changed since the last account-save was performed.

  These last pieces of information are updated by special shell routines which vet all my TCL processing and note when I log on to each account and also when I amend its files either directly or via Q-pointers.

My account-save processor then displays the account details in age (and whether used) sequence. When I have selected a

specific account for saving, it tells me which device to load. When I have mounted the tape/diskette, it checks that this is the correct volume. Finally, when the account-save has been performed and verified, it updates the account record.

We discuss these matters in more detail in the *MB-Guide to file-save and file-restore.* and the *MB-Guide to operations and system management.*

9.2 Physical security

There are certain aspects of physical security some are essential, others should be considered. These include

* housing the computer and the tape/diskette library in a separate room which is accessible only to authorised personnel.

* storing file-save and other tapes/diskettes off-site and away from the main building. Thus, if the computer burns down, there is still a copy of the data to continue on another machine.

  This is particularly important for the Pick system diskettes (or tapes) which are used to load and re-load the system and for other non-copyable software such as CompuSheet+ and Accu/plot. Since Pick will not let you use back-up copies of the system diskettes to reload the software, if the originals are damaged, you will have to but a completely new set (or at best, a set of upgrade diskettes).

  An acquaintance of mine lodges his copies sensitive data and the Pick system diskettes with his bank. This is acceptable until the system goes down after midday on a Saturday.

* locking the computer. Many computers, especially personal computers, have a key which prevents them from being booted up.

* removing the means of access to the computer. A client of mine removes all the keyboards from their terminals and locks these in a safe at the end of the day.

All documentation - and especially that relating to passwords, account-names and the file-statistics - should be controlled by the System Manager.

10 Miscellaneous security aspects

In this section, we look at a number of unrelated points which should be considered by anyone concerned with system security.

10.1 TCL stackers

A TCL stacker keeps a record of all the commands which the users issue. This is good news and also bad news.

The good news is that - by monitoring the stacks - it is possible to see who did what and identify any unauthorised processing.

The bad news is that if the users issue any LOGTO commands using the account name and password, these will be visible to anyone browsing through the stacks.

The stacks are held as normal items on a file. The name of the file - and whether it is a single file for the system or an individual file for each account - depends upon the implementation. Typically, the file(s) will be called

```
STACK
STACKS
TCL-STACK (on AP)
TCL+LIB (on Reality)
LTCL (on OA)
```

10.2 Debuggers

The two debugging pieces of software - the interactive system debugger and the Basic symbolic debugger - are powerful tools for enabling the experienced user to locate and fix problems in his/her programs. They are also useful to the ordinary user in interrupting and abandoning an action which has been invoked in error.

Unfortunately, these tools can wreak great damage if they are misused. This is especially true of the system debugger which, amongst other facilities, allows a user to amend the contents of any frame of virtual memory on the system - a great security risk.

In practice, both tools offer some protection in that the commands which they accept are suitably arcane and it is highly unlikely that the ordinary user would stumble upon valid commands accidentally. Nevertheless, it is worth considering whether or not to disable the system debugger by means of the

```
DB
```

command issued from within the system debugger by a user logged on the SYSPROG account. When the debugger has been disabled in this manner, only the P / END / OFF and G commands of the system debugger will be available. The Basic symbolic debugger is not affected by this action.

10.3 DUMP verb

The DUMP verb allows a user to inspect the contents of any frame of virtual memory. Although the facility is not normally available to users with system privileges below

SYS2, care should be taken to see that it is not abused.

10.4 Editors

Any editor, be it the Pick EDITor or the JET-EDITor, is one of the most powerful tools and, since it allows the user to edit any item on any file, it is also one of the most dangerous.

It should therefore be made inaccessible to all but the most needy users. The Editor can be rendered inaccessible on some systems by reducing the account's system privileges to SYS0, but a safer solution is by removing the ED and EDIT verbs and any other dangerous verbs from the MD. It is perfectly feasible to copy the ED and EDIT verbs to some other items before removing them from the MD. For example, by copying EDIT to an item CHANGE on the MD, then the CHANGE verb will give all the facilities of the EDitor but will be accessible only to those who know of its existence.

This renaming of facilities should be carefully controlled by the System Manager.

10.5 Terminal pages

Many terminals have a facility for storing the most recent display screens and allow the user to browse through these by means of the Page-up and Page-down (or similar) keys. This means that (in some cases) the last 64 screens will be available for inspection, including the account name and password which were entered when the user logged on. If the user is to leave his/her terminal unattended for any time, and when he/she has finished using the system, it may be appropriate to log off and then switch of the terminal, thereby losing the stored pages.

10.6 User-exits

A user-exit is a means of passing control to one of a number of Assembler language routines supplied with the system. Users may also write their own Assembler language routines - called modes - and invoke these in the same manner.

The general form of a user exit is:

```
Uefff
```

where **fff** is the address of the frame at which the Assembly language program has been loaded, and **e** is the entry point into that coding, and these references are encountered in several contexts:

* as the argument in the Basic ICONV and OCONV function,
* as a statement in Procs, and
* as Access conversions and correlatives in attribute definition items.

A number of user exits are available. Most of the user exits which are available are heavily dependent upon the hardware. Many user-exits may differ between implementations and considerable damage - such as causing the entire system to hang - can be caused by attempting to use a code which is not available on you system.

Index

MB-Guides

MB-Guides are designed to serve as introductory texts to a range of fundamental topics within the Pick operating system. They will be available for the following subjects:

MB-Guide to Access conversions and correlatives
MB-Guide to Access sentences
MB-Guide to Basic programming
MB-Guide to Creating and using Procs
MB-Guide to using the Editors
MB-Guide to File design
MB-Guide to File-save and file-restore
MB-Guide to Files: monitoring and sizing
MB-Guide to Group format errors
MB-Guide to Operations and systems management
MB-Guide to Pick on the PC
MB-Guide to Program design
MB-Guide to Security
MB-Guide to The Basic symbolic debugger
MB-Guide to The spooler
MB-Guide to The system debugger
MB-Guide to Using backing storage

The format of the *MB-Guides* is such that they may be easily updated and amended to reflect the current state of the operating system. In order that this and the other Mating MB-Guides continue to meet the needs of the users, we would appreciate your comments on this guide and your suggestions for further titles in this series.

MB-Master self tuition courses are also available on a wide range of topics related to the Pick operating system:

Access techniques
Advancing in Basic
Moving to Basic - a conversion course
Pick systems management
Programming in Basic
Starting Access
Starting ACCU/PLOT
Starting CompuSheet+
Starting Jet
Starting Pick
Starting Runoff
Starting SB+
Systems development
Writing Procs

MB-Guides

The booklets in the *MB-Guide* series cover a range of fundamental topics of interest to users and those responsible for running Pick systems.

Each *MB-Guide* deals with a specific aspect of the operating system and the booklets represent an economical introduction to the various topics and the whole series forms an integrated presentation of the subject matter.

The booklets are intended to be a working document and, for this reason, space is provided for the user's notes, and the reader is encouraged to amend the booklet so that it applies to his/her own system.

It is anticipated that the series of *MB-Guides* will be of special interest to new users, and it should prove useful for software houses and others who are responsible for the instruction of their clients and staff in the fundamental aspects of the Pick operating system.

MB

Malcolm Bull

Training and Consultancy Publications