Chapter Ten

OPENVMS FILE SECURITY

Introduction

One of the most important things a user needs to learn is how to protect data from unauthorized access or accidental deletion.

This lesson explains the reasons for the need to regularly back up and archive files. It also describes how to display and control file protection, as well as how to erase files that are no longer needed.

Objectives

To perform basic data protection tasks, a user should be able to:

- Use UIC-based protection to protect files and directories.
- Save copies of files.

Using File Protection

Overview

You can prevent others from accessing your files with *user identification codes* (UIC’s). A further level of protection can be added through the use of *access control lists* (ACL’s). Typically users only make use of UIC-based file protection.

This section discusses:

- Security profiles
- User Identification codes (UIC’s)
- Protection codes
- Manipulating protection codes
- Access Control Lists (ACL’s)

Security Profiles

Protected Objects

The OpenVMS operating system controls access to *protected objects* (any objects that contain shareable information). Files and directories are examples of protected objects.

Every protected object has a set of access requirements; they specify which individual(s) have the right to access the object and in what manner.

Contents of a User Security Profile

Users processes and applications have *security profiles*, which contain the following information:

- User Identification Code (UIC), identifying the user
- Rights identifiers held by a process
- Privileges, if any

Contents of an Object's Security Profile

The security elements of any object comprise its security profile. An object's security profile contains the following information:

- UIC of the owner of the object
- Protection code
- Access control list (ACL), if any

This information determines which users can access the object, and what types of access are permitted.

User Identification Codes (UIC's)

The OpenVMS operating system uses a file's UIC and protection code to determine who can read, write, execute, and delete the file. Each user has a unique UIC that identifies him to the system; it is assigned by the system manager when the user account is created.

UIC Components

A UIC has two parts: the group number and the member number. The UIC is often displayed in its numeric format:

```
[group,member]
```

When the system manager assigns the UIC to a new user, the group number will often represent a division, department, or project in the organization.

Member numbers are often assigned sequentially as new users join a group. Although several users can belong to the same group, each will have a different member number.

UIC Formats

A UIC has either a numeric or alphanumeric format.

The *numeric format* of a UIC is a pair of numbers in brackets, with the following rules:

- The group number can range from 1 to 37776 (octal).
- There can be from 0 to 177776 (octal) members in each group.
- An example of a numeric UIC is:

```
[100,30]
```

The *alphanumeric format* of a UIC is easier for users to remember. The system manager has the ability to assign names to group numbers so a UIC can also be represented in the following format:

```
[ACCOUNTING,WILSON]
```

where the group number 100 represents the accounting department and member 30 in the accounting department belongs to user Wilson.

The following example illustrates the use the SHOW PROCESS command to display the UIC assigned to your account.

```
$ SHOW PROCESS

17-AUG-2001 22:00:25.32 User: Wilson   Process ID: 60201CB6
                        Node: DONALD   Process name: “Wilson”
Terminal:           RTA2:     (DONALD::WILSON)
User Identifier:    [100,30]
Base Priority:      4
Default file spec: FAL$DISK_USER:[WILSON]
Number of Kthreads:1

Devices allocated: DONALD$RTA2:
$
```

Example 10-1 - Displaying Your UIC

When the file is created, the OpenVMS operating system assigns a UIC to the file. The file’s owner is the creator of the file, unless otherwise specified.

Use the DCL DIRECTORY command with the /OWNER qualifier to see the UIC attached to a file.

```
$ DIRECTORY/OWNER WISHLIST.TXT

Directory FAL$DISK_USER:[9RB]

WISHLIST.TXT;1          [100,30]

Total of 1 file.
$
```

Example 10-2 - Displaying the UIC of a File

Protection Codes

Protection codes describe the kind of access permitted to a file, directory, or other object.

Types of File Access

UIC-based protection allows for four types of file access.

This protection code...	Allows this kind of access...
Read (R)	COPY, TYPE, PRINT, DIRECTORY
Write (W)	EDIT (modify)
Execute (E)	RUN, @, SUBMIT
Delete (D)	DELETE, PURGE, RENAME

All protected objects also support control access, which allows a user to examine and modify the security elements (ACL, protection code, and UIC). While control access is explicitly stated in an ACL, it never appears in the UIC-based protection code.

User Categories

An OpenVMS operating system recognizes four categories of users who need to know the UIC of the file or object to be accessed.

- **SYSTEM (S)** is a category given to users who need special privileges to perform system management tasks.
- **OWNER (O)** is the user with the UIC that matches the one attached to the file. Usually the owner is the user who created the file, but not always.
- **GROUP (G)** includes all users who belong to the same group as the owner (that is, they share the same group number in their UIC's).
- **WORLD (W)** includes all users with access to the system. The other categories are subsets of the WORLD category.

If the access requested is granted to any category to which the requesting process belongs, the access will be allowed. For example, if a group member requests to read the file, the request will be granted because all group members belong to the category WORLD, and that category has read access to the file.

Assigning UIC's to Categories

The UIC of a file or directory is compared with the UIC of a user requesting access to it.

Category	Group Number	Member Number
SYSTEM	0 through 10 (octal)	Does not matter
OWNER	Matches the group number of the UIC	Matches the member number of the file UIC
GROUP	Matches the group number of the UIC	Does not matter
WORLD	Does not matter	Does not matter

The following figure shows the elements of a protection code as it would be specified by the SET SECURITY/PROTECTION command.

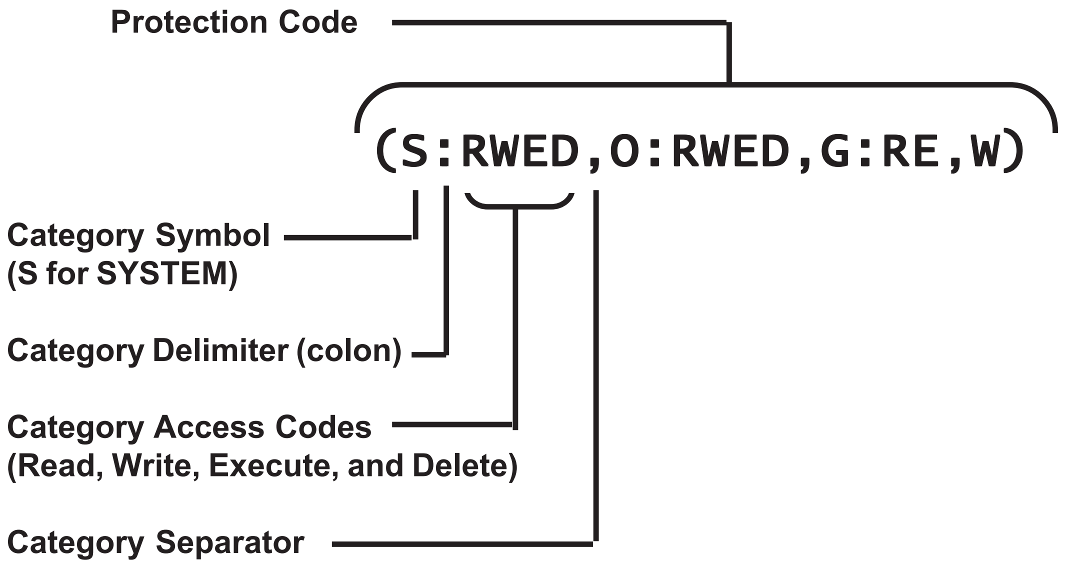
Protection Code
(S:RWED,O:RWED,G:RE,W)
Category Symbol (S for SYSTEM)
Category Delimiter (colon)
Category Access Codes (Read, Write, Execute, and Delete)
Category Separator

Figure 10-1 - Elements of a Protection Code

Access Control Lists (ACL's)

An ***access control list (ACL)*** is a collection of entries that grant or deny access by specific users or groups of users to a protected object.

- Use ACL's when you want to grant access to a protected object for specific users.
- Use ACL's to grant or deny access by individual users or to groups of users. This access is granted independently of their UIC.

Object Classes

You can place ACL's on the following types of object classes in the OpenVMS operating system:

- Common event flag clusters
- Devices (hardware)
- Files (including directory files)
- Group global sections
- Logical name tables
- Queues
- Security classes
- System global sections
- Volumes

Access Control Entries (ACE's)

Information is placed into access control lists using access control entries (ACE's). There are different types of access control entries; some control access to protected objects and others report access.

- ACE's controlling access
  - Identifier ACE's control object access.
  - Default Protection ACE's set a default protection code through a directory structure.
  - Subsystem ACE's control access to protected subsystems.
  - Creator ACE's set the ownership access for new files created in a directory.
- Auditing ACE's
  - Alarm ACE's send event messages to a security operator terminal when a process attempts to access an object.
  - Audit ACE's send event messages to a log file when a process attempts to access an object.

Displaying File Security Information

Use the DIRECTORY command with various qualifiers to display security information for a file.

To display this information...	Use this command:
UIC protection code	DIRECTORY/PROTECTION
Owner	DIRECTORY/OWNER
Access control list (if any)	DIRECTORY/ACL
All of the above	DIRECTORY/SECURITY

The SHOW SECURITY command displays the same information as DIRECTORY/SECURITY. The following is the syntax:

```
$ SHOW SECURITY file-specification
```

The following four examples use various DIRECTORY qualifiers to display a file's protection.

```
$ DIRECTORY/PROTECTION  SAMPLE.FILE

Directory  FAL$DISK_USER:[STUDENT11]

SAMPLE.FILE;2             (RE,RWED,RE,)

Total of 1 file.
$
```

Example 10-3 - Displaying a Directory with /PROTECTION

```
$ DIRECTORY/OWNER  SAMPLE.FILE

Directory  FAL$DISK_USER:[STUDENT11]

SAMPLE.FILE;2             [12,200]

Total of 1 file.
$
```

Example 10-4 - Displaying a Directory with /OWNER

```
$ DIRECTORY/ACL  SAMPLE.FILE

Directory  FAL$DISK_USER:[STUDENT11]

SAMPLE.FILE;2

Total of 1 file.
$
```

Example 10-5 - Displaying a Directory with /ACL

```
$ DIRECTORY/SECURITY  SAMPLE.FILE

Directory  FAL$DISK_USER:[12,200]

SAMPLE.FILE;2        [STUDENT11]    (RE,RWED,RE,)

Total of 1 file.
$
```

Example 10-6 - Displaying a Directory with /SECURITY

Although there is no access control list for SAMPLE.FILE, we know the following:

- SYSTEM and GROUP both have read and execute access.
- OWNER has read, write, execute, and delete access.
- WORLD has no access.

The following example shows the output from the SHOW SECURITY command.

```
$ SHOW SECURITY SAMPLE.FILE

FAL$DISK_USER:[STUDENT11]SAMPLE.FILE;2 object of
class FILE
     Owner:    [12,200]
     Protection:    (System: RE, Owner: RWED,
Group: RE, World)
     Access Control List:   <empty>
$
```

Example 10-7 - Using the SHOW SECURITY Command

Default Protection

Your top level directory has a default file protection code associated with it. When you create a file in your top level directory or any subdirectory, the OpenVMS operating system applies the default protection to the new file unless you specify an alternate protection code. You can specify a different protection code for a file when you create the file or you can change the protection on a file after it is created.

The default protection is not the same as the protection code on the directory. Default protection is the protection code that will be applied to new files created in the directory or in any subdirectories unless the creator has specifically designated a protection code for the directory. The protection code on the directory determines what user categories can access the directory.

Use the SHOW PROTECTION command to see the default protection for your process. This is the protection code that will be assigned to any new files you create, unless you specify a different protection code for them.

```
$ SHOW PROTECTION
   SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
$
```

Example 10-8 - Displaying Default Protection

Use the SET PROTECTION command with the /DEFAULT qualifier to change the default protection for your process.

The following example changes the default protection to allow SYSTEM read, write, and execute access and allow GROUP no access.

```
$ SET PROTECTION=(S:RWE,G)/DEFAULT
$ SHOW PROTECTION
   SYSTEM=RWE, OWNER=RWED, GROUP=NO ACCESS, WORLD=NO ACCESS
$
```

Example 10-9 - Changing Default Protection

Because the OWNER and WORLD categories were not included in the SET PROTECTION command, their access is unchanged.

Notes: Changing Default Protection

When changing a protection code, you need only enter the portion of the code that you wish to change.

Changing the default protection will only affect the protection code on new files you create. New versions of any file that existed before the directory protection was changed will take the protection code of the previous version of the file.

```
1  $ SHOW PROTECTION
     SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
2  $ DIRECTORY/OUTPUT=DIRECTORY1.LIS
3  $ DIRECTORY/PROTECTION DIRECTORY1.LIS

   Directory FAL$DISK_USER:[9RB]

4  DIRECTORY1.LIS;1             (RWED,RWED,RE,)

   Total of 1 file.
5  $ SET PROTECTION=(S:R,G:R)/DEFAULT
6  $ SHOW PROTECTION
     SYSTEM=R, OWNER=RWED, GROUP=R, WORLD=NO ACCESS
7  $ DIRECTORY/OUTPUT=DIRECTORY2.LIS
8  $ DIRECTORY/PROTECTION DIRECTORY*.LIS

   Directory FAL$DISK_USER:[9RB]

9  DIRECTORY1.LIS;1             (RWED,RWED,RE,)
   DIRECTORY2.LIS;1             (R,RWED,R,)

   Total of 2 files.
   $
```

Example 10-10 - The Effects of Changing Default Protection

Notes: The Effects of Changing Default Protection

1. Display the current default file protection.
2. Create a new file.
3. Display the protection on the new file.
4. The new file has the default protection.
5. Change the default file protection so SYSTEM and GROUP have only read access.
6. Display the new default file protection.
7. Create another file.
8. Display the protection on both files.
9. The second file was created with the new protection.

Manipulating Protection Codes

Setting File Protection

Use the SET SECURITY/PROTECTION command to control a file's UIC protection code. The SET SECURITY/PROTECTION syntax is:

```
$  SET  SECURITY/PROTECTION=(prot-code)  file-spec
```

The following command sets a file's protection to allow:

- SYSTEM and OWNER read, write, and execute access
- GROUP and WORLD no access

```
$  SET  SECURITY/PROTECTION=(S:RWE,O:RWE,G,W)  SAMPLE.FILE
```

Protection on Directory Files

To guard against accidental deletion, directory files are created with a different protection than that given to other files. By default, delete access is denied to all user categories.

```
$  DIRECTORY/PROTECTION  COMMANDS.DIR

Directory  MICKEY:[STUDENT11]

COMMANDS.DIR;1          (RWE,RWE,RE,)

Total of 1 file.
$
```

Example 10-11 - Protection on Directory Files

You can change the protection on a directory file the same way you change the protection on any other file.

Procedure for Deleting a Directory File

Because directories are protected against deletion, special steps must be taken when you need to delete a directory file.

Follow these steps to delete a subdirectory:

- Remove all files from the directory, either by deleting them or renaming them to another directory.
- Set your default to a higher level directory.
- Change the protection on the directory file so that you (the owner) can delete it.
- Delete the directory file (.DIR).

The following example shows the steps in deleting a subdirectory.

```
$ SET DEFAULT FAL$DISK_USER:[9RB]
$ DIRECTORY [9RB.DOCS]

Directory FAL$DISK_USER:[9RB.DOCS]

CLASS.LIST;4   CLOCK.EXE;1    COLOR.COM;4    COUNT.EXE;1
LOGIN.COM;39   MY_NOTES.TXT;14 NOTES.COM;4   TRANS.DOC;3
EDTINI.EDT;7

Total of 9 files.
$ DELETE [9RB.DOCS]*.*;*
$ DIRECTORY [9RB.DOCS]
%DIRECT-W-NOFILES, no files found
$ SET SECURITY/PROTECTION=O:RWED DOCS.DIR
$ DELETE DOCS.DIR;1
$ DIRECTORY DOCS.DIR
%DIRECT-W-NOFILES, no files found
$ DIRECTORY [.DOCS]
%DIRECT-E-OPENIN, error opening FAL$DISK_USER:[9RB.DOCS]*.*;* as input
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
$
```

Example 10-12 - Deleting a Subdirectory

Common Errors When Deleting Directory Files

Errors can occur when you try to delete a directory file that is protected against deletion or that still has files (either in the directory or in its subdirectories).

You cannot delete a directory if you do not have delete access to it. Attempting such a deletion will result in an error message similar to the one that follows:

```
%DELETE-W-FILNOTDEL, error deleting DONALD:[STUDENT01]CODE.DIR;1
-RMS-E-PRV, insufficient privilege or file protection violation
```

To delete this directory file you must first change the protection code on the subdirectory to give yourself (the owner) delete access. You can do this by issuing the `SET SECURITY/PROTECTION=O:RWED` command with the name of the directory file appended to the end of the command.

You cannot delete a directory file that still has files in the directory it is managing. Attempting to delete this directory file will result in an error message similar to the following:

```
%DELETE-W-FILNOTDEL, error deleting DONALD:[STUDENT01]CODE.DIR;1
-RMS-E-MKD, ACP could not mark file for deletion
-SYSTEM-F-DIRNOTEMPTY, directory file is not empty
```

To delete this directory file, you must first delete all files in the directory, as well as all files in any subdirectories of this directory.

Saving Copies of Files

Overview

On a large system such as the AlphaCluster at PFPC, the system manager will establish a schedule for making backup copies of files. If you work on a small stand-alone system, you may have to back up your own files. You should perform this operation on a regular basis to make sure that you do not lose important files due to system problems or your own actions.

This section discusses the Backup utility.

The Backup Utility

Use the Backup utility to restore files to disk and to copy files from disk to disk or from disk to tape. You can make copies of:

- Individual files and their locations in a directory structure
- A directory structure and the files in it
- All directory structures and files on a volume

The Backup command has the following syntax:

```
$  BACKUP input-specifier  output-specifier
```

You can use several types of qualifiers to modify the default behavior of the BACKUP command.

- Add a qualifier to the BACKUP command to modify the action of the entire command.
- Qualifiers following the ***input-specifier*** allow you to select files for backup and specify how the input is processed.
- The ***output-specifier*** can also have qualifiers.

Magnetic tape is frequently used as a storage medium for file copies because it is inexpensive and easy to store and transport.

The PFPC system does not back up user databases.
You must back up any databases that you are using.
This is a tester's and programmer's responsibility!

Save Sets

The Backup utility stores files and directories in a specially formatted file called a *save set*. A save set is a single file that can contain the contents of many files and their associated directory structures. For example, a save set can contain all the files and the directory structures in an entire disk.

The special format makes it possible to save the contents of a file, as well as the relationship of a file to a directory structure. For example, the Backup utility can save the the file FILE.DAT and the fact that FILE.DAT was located in the directory [SMITH].

To make a save set of all the files in a subdirectory named CODE, issue the following command:

```
$BACKUP  [.CODE]*.*  CODE_DIR.BCK/SAVE_SET
```

You can create a save set file on a disk or tape volume. A save set is a normal OpenVMS system file with a standard OpenVMS file name. You can copy it, rename it, or delete it using the same DCL commands you use to manage other files.

Use the DIRECTORY command to see the save set created by the BACKUP command, as in the following example:

```
$ DIRECTORY CODE_DIR.BCK

Directory FAL$DISK_USER:[9RB]

CODE_DIR.BCK;1

Total of 1 file.
$
```

Example 10-13 - View a Save Set Using the DIRECTORY Command

Use the BACKUP utility command BACKUP/LIST to list the contents of a save set, as in the following example:

```
$ BACKUP/LIST CODE_DIR.BCK/SAVE_SET

Listing of the save set(s)

Save set:           CODE_DIR.BCK
Written by:         ROBBINS
UIC:                [000200,000030]
Date:               17-AUG-2001 21:20:34.32
Command:            BACKUP [.CODE]*.* CODE_DIR.BCK/SAVE
Operating system:   OpenVMS AXP version E7.2-1
BACKUP version:     V6.2
CPU ID register:    80000000
Node name:          _DONALD::
Written on:         _$1$DUA1:
Block size:         32256
Group size:         10
Buffer count:       28
[9RB.CODE]DOCS.FOR;4               20   17-AUG-2001 15:55
[9RB.CODE]LOGICAL_3.FOR;38          2   12-AUG-2001 13:56
[9RB.CODE]PLAN.FOR;1               15   13-AUG-2001 14:10
[9RB.CODE]RBR_COUNT.FOR;14          1   11-AUG-2001 19:05
     •
     •
     •
Total of 22 files, 135 blocks
End of save set
$
```

Example 10-14 - List the Contents of a Save Set

Restoring a File From a Save Set

To retrieve a file from a save set and restore it to a directory, use the BACKUP command with the save set as the input specifier. If the input specifier is a save set on disk, use the /SAVE_SET qualifier to the input specifier. To restore a selected list of files (rather than the entire save set), use the /SELECT qualifier, as in the following command:

```
$ BACKUP CODE_DIR.BCK/SAVE_SET/SELECT=RBR_COUNT.FOR *.*
```

Summary

Concepts

Using File Protection

- User processes and applications have security profiles, which determine whether they have the authority to access particular objects.
- Objects have security profiles, which specify who can access them and what kind of access is permitted.
- Every file has a user identification code (UIC) that identifies the user who created or owns it.
- Every file has a protection code that establishes the type of access a particular category of user has to the file.
- Use DCL commands to display and alter the protection of your files.

Saving Copies of Files

- Archiving and regular backup of files help to protect data from loss or corruption.

Commands

Using File Protection

SHOW PROCESS
Display process information, including the UIC assigned to your account.
DIRECTORY/PROTECTION
Display protection code information about a file or files in a directory.
DIRECTORY/OWNER
Display the owner information of a file or files in a directory.
DIRECTORY/ACL
Display any Access Control Lists attached to a file or files in a directory.
DIRECTORY/SECURITY
Display security information about a file or files in a directory.
SHOW SECURITY
Display security information about a file.
SHOW PROTECTION
Display default protection for your process.
SET PROTECTION/DEFAULT
Change the default protection for your process.
SET SECURITY/PROTECTION
Change the protection on a file.

Saving Copies of Files

BACKUP
Runs the Backup utility.