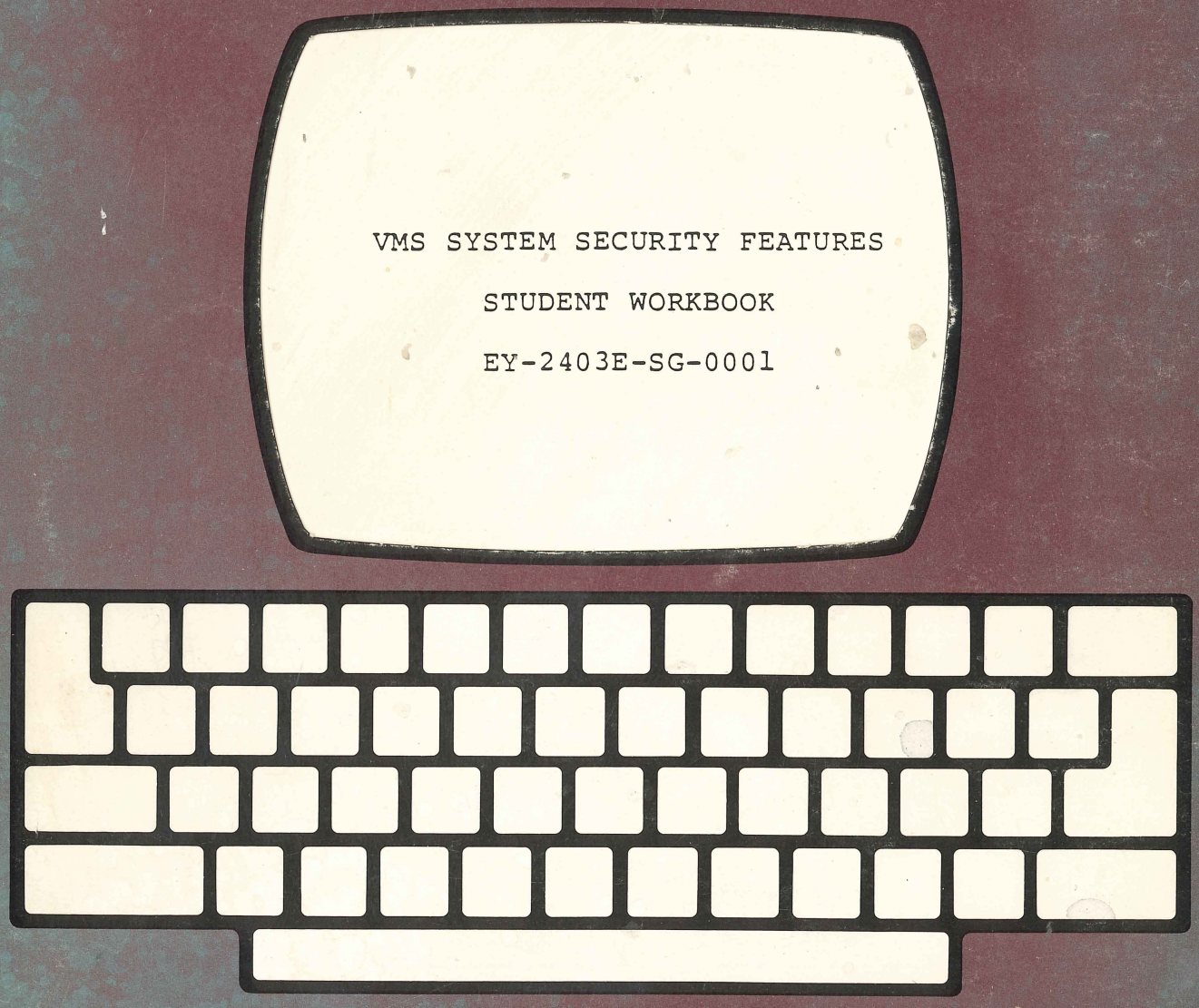
digital

educational services

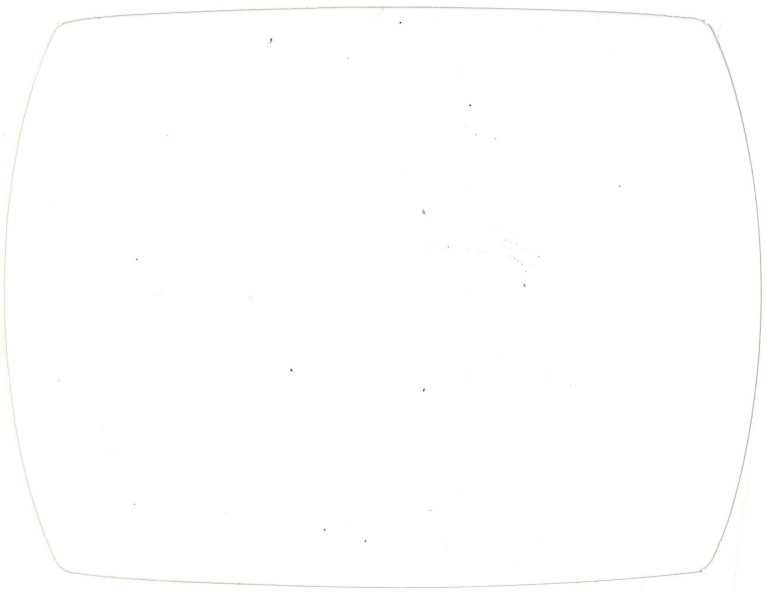


VMS SYSTEM SECURITY FEATURES

STUDENT WORKBOOK

EY-2403E-SG-0001

your future – today's education

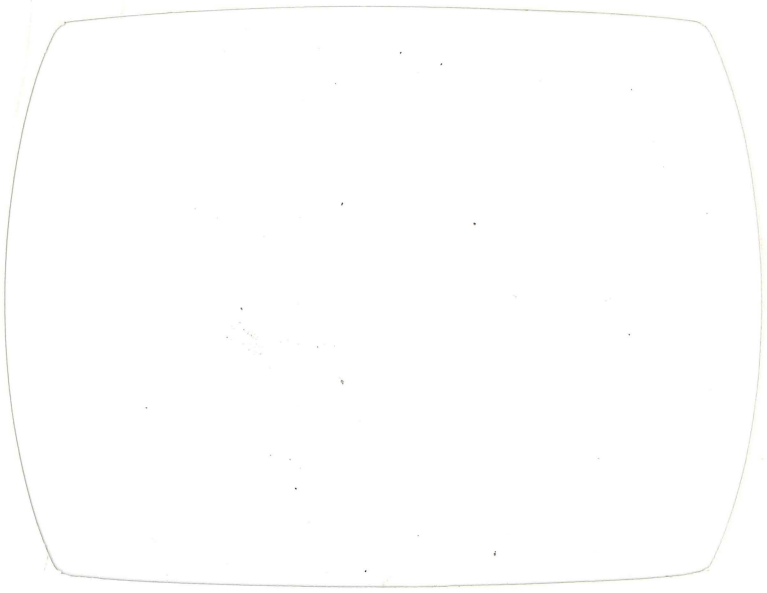


IAN BURGESS

VMS SYSTEM SECURITY FEATURES

STUDENT WORKBOOK

EY-2403E-SG-0001



VMS SYSTEM SECURITY FEATURES

STUDENT WORKBOOK

EY-2403E-SG-0001

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, March 1985

This book was produced on a DIGITAL Word Processing System. Book production was done by Educational Services Development and Publishing in Bedford, MA.

CONTENTS

EY-2403E-SG-0001

VMS System Security Seminar

Student Handout

Prepared by Educational Services
of
Digital Equipment Corporation

EXAMPLES

FIGURES

TABLES

PREFACE

COURSE DESCRIPTION

This course describes the methods and structures used to implement the security features of VAX/VMS V 4.0.

These methods and structures include file protection features, VMS privileges, the system rights database, process rights lists, access control lists, security auditing, data encryption methods, and login controls.

The DCL commands, utilities, and system services related to monitoring and controlling system security are explained. Methods of monitoring attempts to breach system security are discussed.

INTENDED AUDIENCE

System managers, security analysts, project managers and programmers interested in learning about VAX/VMS V 4.0 security features.

PREREQUISITES

* Completion of the VAX/VMS Utilities & Commands course.
* Completion of the VAX/VMS System Management course or
* Completion of Utilizing VMS Features course or
* 6 months or more of VAX system management experience.

PREFACE

OBJECTIVES

UPON COMPLETION OF THE COURSE THE PARTICIPANTS WILL BE ABLE TO :

- EXPLAIN BASIC SECURITY PRACTICES AND PITFALLS
- IMPLEMENT A SECURE METHOD OF PASSWORD AND LOGIN CONTROLS
- CONTROL LOCAL AND REMOTE ACCESS TO A VAX SYSTEM
- PREVENT USERS FROM OBTAINING UNAUTHORIZED INFORMATION BY 'DISK SCAVENGING'
- EXPLAIN AND CONTROL ACCESS TO FILES AND OTHER STRUCTURES ON THE SYSTEM
- CREATE AND MAINTAIN THE SECURITY RIGHTS DATABASE, ACCESS CONTROL LISTS AND ACCESS CONTROL LIST ENTRIES
- IMPLEMENT A SECURITY AUDITING SYSTEM TO CONTROL AND BE WARNED ABOUT FILE ACCESSES, ACL USE, LOGINS, MOUNTS/DISMOUNTS AND/OR UAF CHANGES
- USE THE APPROPRIATE DCL COMMAND, VMS UTILITY AND/OR SYSTEM SERVICE, NECESSARY TO IMPLEMENT AND MAINTAIN SECURITY CONTROL PROCEDURES
- EXPLAIN AND CONTROL ACCESS TO THE SYSTEM THROUGH DECNET
- CONTROL ACCESS TO A VAXCLUSTER
- BE ABLE TO USE THE DATA ENCRYPTION FACILITIES OF VMS

TOPICS NOT DISCUSSED

- HOW TO BREAK INTO A VAX SYSTEM
- BASIC SYSTEM MANAGEMENT FUNCTIONS
- UTILIZING VMS FEATURES FROM A PROGRAMMING LANGUAGE
- VMS INTERNALS
- DECNET MANAGEMENT OR PROGRAMMING
- VAXCLUSTER MANAGEMENT OR PROGRAMMING

CHAPTER 1
OVERVIEW OF COMPUTER SYSTEM SECURITY

LIST OF TOPICS

- THE SECURITY THREAT
  - Proprietary Information
  - User Irresponsibility and Untrustworthiness
  - Degree of sharing among users
  - Sophistication/Capabilities of the users
  - Browsing
  - Scavenging
  - Operating System Penetrations
  - Indirect Attacks
- CRACKING VMS - 'WAR STORIES'
- TYPES OF SECURITY CONTROLS
- PHYSICAL SECURITY
- REPORTING SECURITY PROBLEMS
- CONCLUSION

1.1 THE SECURITY THREAT

Analysis of the needs of secure system must start with an assessment of the threats to the security of the system. The greater the threat to the system the more sophisticated the security measures will need to be.

Before you think about protecting your data you must have data that needs protecting. The kind of information that may need to be protected is proprietary information.

DEFINITION OF PROPRIETARY INFORMATION

ANY INFORMATION USED IN YOUR BUSINESS AND NOT IN THE PUBLIC DOMAIN WHICH CONCERNS YOUR PRODUCTS, SERVICES, STRATEGIES, TECHNIQUES, PROCESSES OR PERSONNEL AND WHICH, IF IN THE POSSESSION OF A COMPETITOR OR POTENTIAL COMPETITOR COULD LESSEN YOUR COMPETITIVE POSITION OR PROFITABILITY AND/OR INCREASE THE COMPETITOR'S MARKET POSITION OR PROFITABILITY.

It is this information that needs to be protected. However, the existence of sensitive data alone does not imply a high threat to that data.

The actual security threat may be evaluated in several areas.

TABLE 1-1: ITEMS OF CONCERN FOR EVALUATION OF A SECURITY THREAT

- USER IRRESPONSIBILITY AND UNTRUSTWORTHINESS
- DEGREE OF SHARING AMONG USERS
- SOPHISTICATION/CAPABILITIES OF THE USERS
- BROWSING/PROBING
- SCAVENGING
- OPERATING SYSTEM PENETRATIONS

USER IRRESPONSIBILITY AND UNTRUSTWORTHINESS

User irresponsibility may cause data to be accidentally deleted or compromised. It is not necessarily maliciousness but rather carelessness of the user. This may come from poor application design or inconsistent controls on the users.

The main action a security manager can take to protect the system against this kind of attack is to restrict each user to a particular area of access at a specific time, with specific capabilities.

Not all users can be trusted, and user maliciousness can compromise your system. Information is power, and a computer provides a tremendous concentration of information and potential power. The security manager needs to be aware of the sensitivity and value of the information on the system. Sensitive information alone does not mean you will have a security threat, but the more valuable the information, the more motivation a user may have for trying to access that information.

Most people are basically honest. However, many people given the right circumstances would breach computer security.

TABLE 1-2: CIRCUMSTANCES THAT COULD CAUSE A BREACH OF SECURITY

- ATTITUDE
- NEED
- OPPORTUNITY

TABLE 1-3: COMMON MOTIVES FOR COMPUTER CRIME

- GREED
- FINANCIAL PROBLEMS
- A CHALLENGE
- "ROBIN HOOD" SYNDROME
- REVENGE
- TRADE SECRETS
- WORLD DOMINATION
- FREE COMPUTER TIME

TABLE 1-4: POTENTIAL ADVERSARIES

- DISGRUNTLED EMPLOYEES
- ACTIVISTS
- FINANCIERS
- FOREIGN GOVERNMENTS
- VENDORS
- COMPETITORS
- HACKERS

DEGREE OF SHARING AMONG USERS

The amount of sharing of data among users of the system will affect the potential threat to security. If no data must be shared it will be easier to protect it. The location of users who will have to share the data will also be important.

In most systems some sharing of the data will be necessary, but by restricting the number of users who need to see the information, you may reduce the threat to that information.

SOPHISTICATION/CAPABILITIES OF THE USERS

The threat to sensitive data may come from very sophisticated systems programmers or from data entry personnel, and anyone in between. You must assume that the threat may be from any level of sophistication, and take appropriate measures to counteract each of those types of threats.

You must also assume that the perpetrator has sophisticated computer equipment at his/her disposal.

The value of the information to you and to the attacker will determine the amount of security you will place on the system.

BROWSING

Browsing or probing by the user is the simplest type of internal security threat. This happens because a portion of the system is insufficiently protected. Browsing may be a casual attempt to just learn more about the system or it may just be started as a 'lark'. It usually involves a somewhat knowledgeable user. Table 1-5 lists several types of browsing that that users may do. This probing may happen from authorized or unauthorized users.

TABLE 1-5: TYPES OF BROWSING

- THEFT OF INFORMATION
- THEFT OF SERVICES
- DESTRUCTION OF DATA
- INVASION OF CONFIDENTIAL INFORMATION

NOTE

USER PROBING is a serious problem and should not be lightly dismissed. In may sites it will be the area that is most likely to be exploited. This probing can be from employees or outsiders. As soon as the system allows dial-in or network access this probing can come from anywhere.

User probing is usually the result of sloppy system management.

Access to files should be controlled so that only those who need the files can access them. VMS V4 provides a very complete method of controlling file access. This will be discussed in Chapter 6.

SCAVENGING

Scavenging is the act of retrieving information from old and/or discarded information on the disk or in memory. If the information in memory is given up, but not zeroed then the next user can read that information. The same idea applies to disk files that have been deleted and the blocks released, but the information not erased.

VMS provides methods of preventing scavenging and these methods are discussed in Chapter 5. VMS prevents memory scavenging.

OPERATING SYSTEM PENETRATIONS

An operating system penetration involves locating and exploiting a 'bug' or an ill chosen 'feature' in an operating system that allows a user to gain control of the system, privileges, or access to files he/she would not normally be allowed.

This penetration point can occur because of a 'bug' in software, hardware, or microcode. It often happens because a perpetrator has infiltrated the development team and he or she has planted this penetration point.

All systems are checked for bugs and for security weaknesses, testing can only find bugs, not prove the absence of all bugs. The testing team would need to find all bugs, while a penetrator need only find one bug that can be exploited.

INDIRECT ATTACKS

Indirect attacks are pieces of software placed in the system by the system developers or application programmers. This software will at some time compromise the security of the system.

Personnel in development positions are in a prime position to compromise the information. You must ask to what degree has it been necessary to trust an individual developer. Has his/her work been independently checked.

Indirect attacks take one of two forms : Trojan horses and trap doors.

Trojan Horse -

The 'Trojan horse' is the ultimate weapon of the computer system penetrator. A 'Trojan horse' is a clandestine modification to a piece of software that a user will execute. The modification can copy information from the user's sensitive file and make the information available to the penetrator. The user does not know that anything has gone wrong, because the program will also perform all of its advertised functions.

The 'Trojan horse' represents one of the easiest penetration paths. Any outside utility or product that you install could contain such a 'Trojan horse'. Some of these attacks can be avoided by not accepting 'free' software without having the sources, and by reading the sources before putting them on your system.

Trap Doors -

A trap door is a custom designed operating system bug. If the penetrator can gain access to the system development facility, on-line diagnostic system, or microcode floppy disk such a trap door could be installed. This access could be by infiltrating an agent or tapping the communication lines. Once the trap door is installed it will remain dormant until activated by a special call to a system routine known only to the penetrator.

Such a trap door could be installed in the operating system of any major computer manufacturer. While good management procedures cannot guarantee the absence of trap doors or Trojan horses, Table 1-6 lists some methods which will help you to prevent these intrusions.

NOTE

Digital has taken steps to protect VMS and layered products from trap doors. RDC is a case of an on-line diagnostic system which is somewhat protected against this abuse.

TABLE 1-6: 'TROJAN HORSES' AND VMS

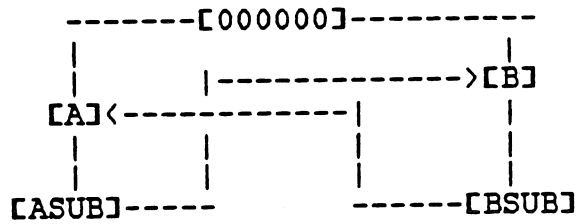
- DECUS TAPES
- BOGUS DISTRIBUTIONS
- WORLD WRITEABLE FILES IN SYS$MANAGER: AND SYS$SYSTEM:
- UPDATE OF NETWORK DATABASE OVER NETWORK
- ACCEPT NO GIFTS ← OR CHECK CAREFULLY
- KNOW YOUR PERSONNEL
- INDEPENDENTLY CHECK YOUR PROGRAMMER'S WORK

1.2 CRACKING VMS - 'WAR STORIES'

TABLE 1-7: VMS V2 LOOPHOLES

- PASSWORD WAS MAPPED BY THE CRC INSTRUCTION
- CTRL/Y OUT OF AN INSTALLED PRIVILEGED IMAGE AND RETAIN PRIVILEGES
- RUN AN INSTALLED IMAGE WITH THE DEBUGGER AND DEPOSIT/COMMAND YOUR PROGRAM
- SYS$SYSTEM:PAGEFILE.SYS WAS WORLD READABLE
- EDIT .DIR FILES AND CAUSE BACKUP TO STAY IN AN INFINITE LOOP (see Table 1-8)

TABLE 1-8: RECURSIVE DEFINITION OF DIRECTORIES



- THE UIC WOULD BE OVERWRITTEN IF YOU ENTERED :

  Username: FINCH/CLI = xxx (more than 15 characters)

- THE PRIVILEGE MASK WOULD BE OVERWRITTEN IF YOU ENTERED :

  Username: FINCH/DISK = xxx (more than 137 characters)

- DISK SCAVENGING
- PASSWORD GRABBERS

1.3 SYSTEM ACCOUNTS

The passwords on SYSTEM accounts will need to be changed regularly, not written down, not obvious (i.e. MANAGER), and not known by a large number of people.

Access to a privileged system account is the easiest way to break into a VMS system. Once on the system in a privileged account, all security measures will be compromised.

TABLE 1-9: SYSTEM ACCOUNTS TO GUARD ON VMS

USERNAME	DEFAULT PASSWORD
o SYSTEM	MANAGER
o FIELD	SERVICE
o SYSTEST	UETP

ALLINI
MRMAN

NOTE

VMS V4 provides simple methods of maintaining the passwords of all accounts including these privileged system accounts. Methods of controlling these and other passwords are discussed in Chapter 4.

The installation and the upgrade procedures for a VMS V4 system will insist that you change the passwords for these accounts.

1.4 TYPES OF SECURITY CONTROLS

Internal security controls come in two major classes, discretionary and non-discretionary.

DISCRETIONARY CONTROLS

Discretionary controls are controls that can be changed at the discretion of the owner of a protected object. Some forms of these controls commonly appear on time sharing operating systems today. These discretionary controls can be further classified into four types.

Password On Files -

Many systems implement security by putting a password on every file. To gain access to a file, the user must enter the password of that file. Using this method has proven unwieldy. The password must be known by all users who must share a file. The user either must remember a large number of passwords, and write them down, or use the same password for all files and therefore share all files. VMS V4 does not implement this form of protection.

System/Owner/Group/World Protection -

Most Digital operating systems use some form of this discretionary protection on files. It is easy to use, but does not always provide the granularity required for sharing of files. VMS V4 still uses this method, but supplements it with access control lists.

Access Control Lists -

Access Control Lists (ACL's) provide the generality that is required in many sites. An ACL is a list of users who are allowed access to a particular file and what type of access that user is allowed. The owner of a file can specify exactly what access a particular user can have to that file. There is no limit on the size of a particular ACL.

ACL's are a feature VMS V4 and are discussed in Chapter 6.

Capability Based Controls -

In a capability based system of controls the user is given a particular capability. This is similar to a ticket to a file and permits a certain access to that file. The copying and granting of capabilities is done in a controlled manner, therefore permitting the controlled sharing of resources.

Capabilities must be inexpensive, non-forgeable, and context independent (i.e. system wide addresses). This implies that there must be special machine instructions to act upon capabilities, and most machines do not currently have this ability.

NON-DISCRETIONARY CONTROLS

Non-discretionary controls are such that the owner of a file may not arbitrarily grant access to anyone he/she wishes. The system is a fixed authorization system based on some predetermined policy such as the Department of Defense controls for national security information. These type of controls also have many commercial applications such as for use in large time-sharing systems.

VMS/VMS V4 does not support this approach, but there are plans to include discretionary controls in future releases of the operating system.

NONDISCRETIONARY controls and their future implementation are discussed in Chapter 12.

SE VMS - does allow Non discretionary controls

1.5 PHYSICAL SECURITY

Physical security of the computer hardware is an absolute essential. Physical security will not be discussed in this seminar. We will assume that the items listed in Table 1-10 concerning physical security have been implemented at your site.

TABLE 1-10: PHYSICAL SECURITY ASSUMPTIONS

- THE MACHINE ROOM IS PROPERLY GUARDED AGAINST ACCESS BY UNAUTHORIZED INDIVIDUALS
- THE MACHINE ROOM IS SHIELDED AGAINST COMPROMISING EMANATIONS (BOTH MICROWAVE AND RF)
- REMOTE TERMINALS ARE LOCATED IN AREAS PHYSICALLY SECURED
- BACKUP TAPES AND DISKS ARE NOT PUBLICALLY ACCESSIBLE
- THE LEVEL OF THE SECURITY ON A TERMINAL IS THE SAME AS THE SECURITY OF THE DATA THAT WILL BE PROCESSED ON THAT TERMINAL
- THE UTILITY SERVICES ARE SUITABLE, RELIABLE, AND CONSISTENT
- THE PHYSICAL SITE IS RELATIVELY SAFE FROM FLOODS AND EARTHQUAKES
- COMMUNICATION LINKS ARE ENCRYPTED

1.6 REPORTING SECURITY PROBLEMS

- ALL KNOWN SECURITY HOLES HAVE BEEN FILLED

- TO REPORT A SECURITY PROBLEM
  - SUBMIT A HIGH PRIORITY SPR
  - DO NOT ADVERTISE THE PROBLEM

1.7 CONCLUSION

Each site will have unique security requirements. Some sites may be satisfied with minimal measures while other sites cannot tolerate even the slightest probing. Many defense applications would be among such sites. Somewhere in between are many commercial sites.

TABLE 1-11: ACCESSING THE NEEDS OF A SECURITY SYSTEM

- DO YOU HAVE A LEGITIMATE SECURITY THREAT?
- ARE YOU CONTENT WITH YOUR PRESENT DEGREE OF SECURITY?
- WILL YOU TAKE THE TIME AND/OR ENERGY TO IMPLEMENT ADDITIONAL SECURITY MEASURES CORRECTLY?

TABLE 1-12: CATEGORIZING SECURITY REQUIREMENTS ACCORDING TO TOLERANCE TO EVENTS

COULD YOU TOLERATE THE FOLLOWING?	LEVEL OF SECURITY REQUIREMENTS LOW	MEDIUM	HIGH
A USER KNOWING THE IMAGES BEING EXECUTED ON YOUR SYSTEM	Y	Y	N
A USER ACCESSING THE FILES OF OTHER USERS IN THE GROUP, INCLUDING MAIL	Y	Y	N
AN OUTSIDER KNOWING THE NAME OF THE NODE JUST DIALED INTO	Y	Y	N
A USER COPYING FILES OF OTHER USERS	Y	N	N
A USER KNOWING THE NAMES OF OTHER USER'S FILES	Y	N	N
A USER BEING ABLE TO READ SECTIONS OF A DISK THAT MIGHT CONTAIN VARIOUS OLD FILES	Y	N	N
A USER CONSUMING MACHINE TIME AND RESOURCES TO PERFORM UNRELATED OR UNAUTHORIZED WORK. (I.E. PLAYING GAMES)	Y	N	N
A USER WRITING DATA INTO ANOTHER USER'S FILE	N	N	N
A USER DELETING ANOTHER USER'S FILE	N	N	N

Table 1-12 classifies your level of security needs by looking at your tolerance to certain events. Those sites that are most intolerant to the events generally have very high levels of security requirements.

Remember, all security improvements include a cost to the site. The cost may be difficult to measure in dollars and cents, but it will exist.

TABLE 1-13: COSTS OF SECURITY IMPROVEMENTS

- SLOW DOWN ACCESS TO DATA
- DELAY YOUR STAFF IN THEIR ROUTINE JOBS
- SLOW DOWN OVERALL SYSTEM PERFORMANCE
- REQUIRE ADDITIONAL STEPS BY PERSONNEL
- MORE SYSTEM MANAGEMENT/ADMINISTRATION TIME & RESOURCES

THERE IS NO SUCH THING AS A FREE LUNCH

TABLE 1-14: WHERE VMS V4 CAN ADD SECURITY SCREENING FUNCTIONS

- DURING LOGINS

  USE PASSWORDS TO KEEP OUT UNWANTED USERS

- DURING USER AUTHORIZATIONS -- FOLLOWING PASSWORD CLEARANCE

  ISSUE POWERS AND PRIVILEGES ON A HIGHLY DISCRETIONARY BASIS

- DURING DATA PROCESSING OR TRANSMITTAL

  ENCRYPT OR DECRYPT DATA

- DURING ACCESS TO FILES AND OTHER SYSTEM RESOURCES

  APPLY PROTECTION OR ACCESS CONTROL RESTRICTIONS

- DURING DEFINED ACTIVITIES

  MONITOR EVENTS AND REPORT OCCURRENCE VIA ALARMS

TABLE 1-15: SECURITY POSTULATES

* BE REALISTIC - BE RATIONAL
* OVERUSE OF SECURITY MAY BE EASY TO JUSTIFY
* ABSOLUTE SECURITY RESULTS IN ZERO PRODUCTIVITY . . . SELECT YOUR RISKS
* THE MOST SECURE SYSTEMS ARE THE HARDEST TO USE
* NOT ALL MEASURES ARE FOR EVERY SITE
* INCREASE YOUR IDEA OF THE SOPHISTICATION LEVEL OF YOUR POTENTIAL ADVERSARY
* DECREASE THE SOPHISTICATION LEVEL OF YOUR AVERAGE USER
* BEWARE OF SOMETHING FOR NOTHING - TAKE NO GIFTS
* KNOW YOUR SYSTEM
* EDUCATE THE USERS
* T A N S T A F L

CHAPTER 2
SECURITY REFERENCE MONITOR CONCEPTS

LIST OF TOPICS

- INTRODUCTION TO REFERENCE MONITOR MODEL
- VMS IMPLEMENTATION
- IDENTIFIERS AND THE REFERENCE MONITOR
- USING VAX/VMS SECURELY

2.1 INTRODUCTION TO THE REFERENCE MONITOR

The reference monitor concept is a model of a SECURE MULTI-USER COMPUTER SYSTEM. It is the result of extensive research concerning secure computer systems conducted over the last 15-20 years.

While not being a complete implementation of the reference monitor, VMS V4 attempts to follow many of the principles of the model. The reference monitor model did provide the foundations that guided the development of many security features implemented in VMS V4. It should also provide a basis for security additions to future versions of the operating system.

The REFERENCE MONITOR MODEL depicts a computer system in terms of SUBJECTS, OBJECTS, an AUTHORIZATION DATABASE, an AUDIT TRAIL, and a REFERENCE MONITOR MECHANISM. Table 2-1 lists the requirements of an ideal reference monitor mechanism. Section 2.1 discusses the VMS implementation of the reference monitor model.

TABLE 2-1: REFERENCE MONITOR MECHANISM REQUIREMENTS

- MEDIATE EVERY ATTEMPT BY A SUBJECT TO GAIN ACCESS TO AN OBJECT
- PROVIDE A TAMPERPROOF DATABASE AND AUDIT TRAIL THAT ARE THOROUGHLY PROTECTED FROM UNAUTHORIZED OBSERVATION AND MODIFICATION
- REMAIN SMALL, SIMPLE AND WELL-STRUCTURED, IN ORDER TO ASSURE EFFECTIVENESS IN ENFORCEMENT

2.2 VMS IMPLEMENTATION

- SUBJECT
  - ACTIVE AGENT THAT NEEDS ACCESS TO INFORMATION ON BEHALF OF PEOPLE
  - VMS PROCESS
  - NEED TO CONTROL ASSOCIATION OF USER WITH SUBJECT
  - ASSOCIATION IS MADE AT PROCESS CREATION
  - ASSOCIATION CONTROLLED WITH LOGIN CONTROLS & PASSWORD CONTROLS
  - NEED TO CONTROL ACCESS TO INFORMATION BY PROCESSES
  - ACCESS CONTROLLED WITH UIC's & PROCESS IDENTIFIERS

- OBJECT
  - ANY RESOURCE IN THE SYSTEM THAT NEEDS ACCESS PROTECTION
  - PASSIVE REPOSITORY OF INFORMATION
  - FILES AND DIRECTORIES ON FILES-11 ODS-2 DISK DEVICES
  - SECTIONS, MAILBOXES, LOGICAL NAMES, EVENT FLAG CLUSTERS, QUEUES

- AUTHORIZATION DATABASE
  - DESCRIBES ALL ACCESS AUTHORIZATIONS
  - RELATES ALL SUBJECTS TO ALL OBJECTS THROUGH IDENTIFIERS
  - DISTRIBUTED AND STORED WITH THE OBJECT TO BE PROTECTED
  - IMPLEMENTED BY UIC-BASED PROTECTION, ACL's, & OTHER IDENTIFIERS ASSOCIATED WITH A PROCESS
  - LEVEL OF FLEXIBILITY DEPENDS ON THE OBJECT BEING PROTECTED

SECURITY REFERENCE MONITOR CONCEPTS

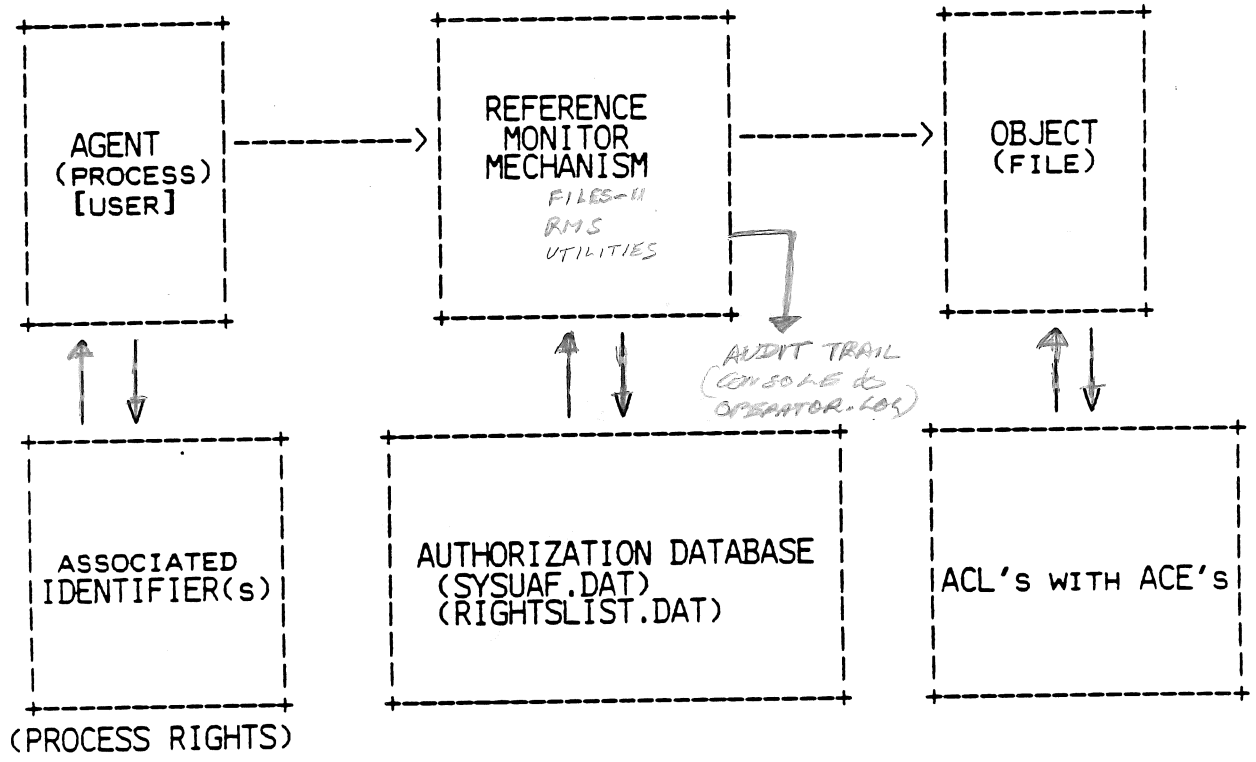
= ANY SECURITY ENABLED TERMINAL
= CONSOLE LOG
= OPERATOR.LOG

- AUDIT TRAIL
  - ACQUIRES A RECORD OF ALL ACCESS ATTEMPTS
  - ALLOWS STRAIGHTFORWARD AUDITING OF A VARIETY OF EVENTS
  - MUST BALANCE DESIRE TO AUDIT EVERYTHING AND REALITY OF LOOKING AT RECORDS
  - EFFECTIVENESS DEPENDS ON EXAMINING THE AUDIT TRAIL

- REFERENCE MONITOR
  - ENFORCES THE SECURITY RULES
  - IMPLEMENTED THROUGHOUT THE VMS EXECUTIVE
  - LARGE BUT DESIGNED SO AS NOT TO BYPASS SYSTEM SECURITY
  - SOME PROCESS PRIVILEGES CAN ALLOW A PROCESS TO BYPASS THE REFERENCE MONITOR
  - SOME PROCESS PRIVILEGES HELP A PROCESS MAINTAIN THE REFERENCE MONITOR

- IDENTIFIER
  - QUALITY ASSOCIATED WITH AN OBJECT OR AGENT
  - INDICATES SOME RELEVANT INFORMATION ABOUT THAT AGENT OR OBJECT
  - WITH AGENT -> PROCESS RIGHTS
  - WITH OBJECT -> FILE PROTECTION
  - WHEN A SUBJECT TRIES TO ACCESS AN OBJECT, THE TWO SETS OF IDENTIFIERS ARE COMPARED AND AN ACCESS DECISION IS MADE

FIGURE 2-1: SECURITY REFERENCE MONITOR FLOWCHART



2.3 USING VAX/VMS SECURELY

TABLE 2-2: ARE YOU USING YOUR SYSTEM SECURELY?

- HOW ARE USERS ASSOCIATED WITH SUBJECTS?
- WHAT IS THE RELIABILITY OF THE AUTHENTICATION MECHANISM?
- WHAT OBJECTS CONTAIN SENSITIVE INFORMATION IN THIS SYSTEM OR APPLICATION?
- IS ACCESS TO THOSE OBJECTS CONTROLLED?
- DOES THE AUTHORIZATION DATABASE REFLECT POLICY?
- WHO IS AUTHORIZED TO GAIN ACCESS TO SENSITIVE OBJECTS?
- ARE ADEQUATE RESTRICTIONS IN PLACE?
- IS THE AUDIT TRAIL RECORDING ENOUGH INFORMATION OR TOO MUCH?
- WHO WILL LOOK AT IT?
- WHAT PROGRAMS ARE FUNCTIONING AS PART OF THE REFERENCE MONITOR MECHANISM?
- WHAT USERS CAN MODIFY THE MECHANISM AND THE AUTHORIZATION DATABASE?
- IS THIS THE DESIRED CONFIGURATION?

CHAPTER 3
LOGIN CONTROLS

LIST OF TOPICS

- CLASSES AND TYPES OF LOGINS
  - Classes of Logins
  - Interactive Vs. Noninteractive Logins
- INFORMATIONAL MESSAGES
  - Types Of Informational Messages
  - Controlling The Display Of The Messages
- LOGIN Restrictions
  - SHIFT Restrictions
  - LOGIN CLASS Restrictions
  - Login Retries
- BREAKIN DETECTION
  - LGI SYSGEN PARAMETERS
  - Suspects/Intruders

3.1 CLASSES AND TYPES OF LOGINS

The sequence of actions a user follows to gain access to the system is known as logging in. This is the system's first opportunity to check users who request system access to insure they are desirable. Generally users must identify themselves with a user name and password to prove they are authorized to use the system.

This is also a time for the system to impose restrictions on the user regarding his or her access to the system.

A more secure site will have a more involved login sequence for the users. The users at such a site will also encounter more restrictions once they are logged on to the system.

There are seven identifiable classes of logins on a VMS system: **LOCAL, DIALUP, REMOTE, NETWORK, BATCH, DETACHED** and **SUBPROCESS**. These seven classes of logins each employ one of the two methods of logging in: **INTERACTIVE** or **NONINTERACTIVE**.

Table 3-1 lists the different login classes and types.

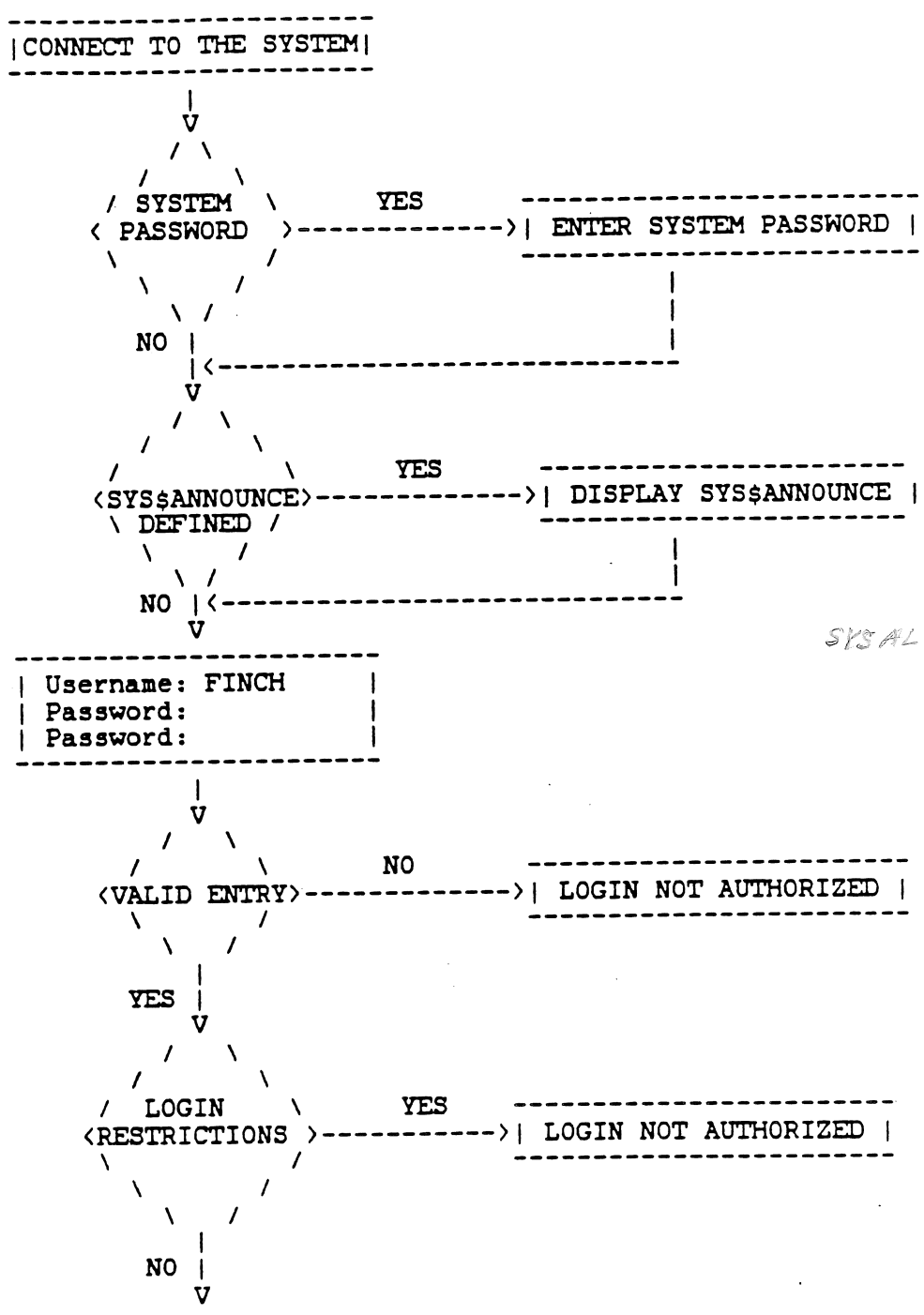
In **INTERACTIVE LOGINS** there is some communication between the system and the user. The user will provide the system with the answers to a series of questions and depending on the answers the system will permit or deny the user access to the system. Figure 3-1 pictures the steps an interactive login follows.

NONINTERACTIVE LOGINS involve no action from the user during the login.

NOTE

While interactive logins as described here are somewhat different from an interactive mode process, any form of interactive login will result in an interactive mode process. That is one that returns the string 'INTERACTIVE' from the lexical function F$MODE() or 'TRUE' from the lexical function F$ENVIRONMENT("INTERACTIVE").

FIGURE 3-1: SEQUENCE OF AN INTERACTIVE LOGIN



LOGIN CONTROLS

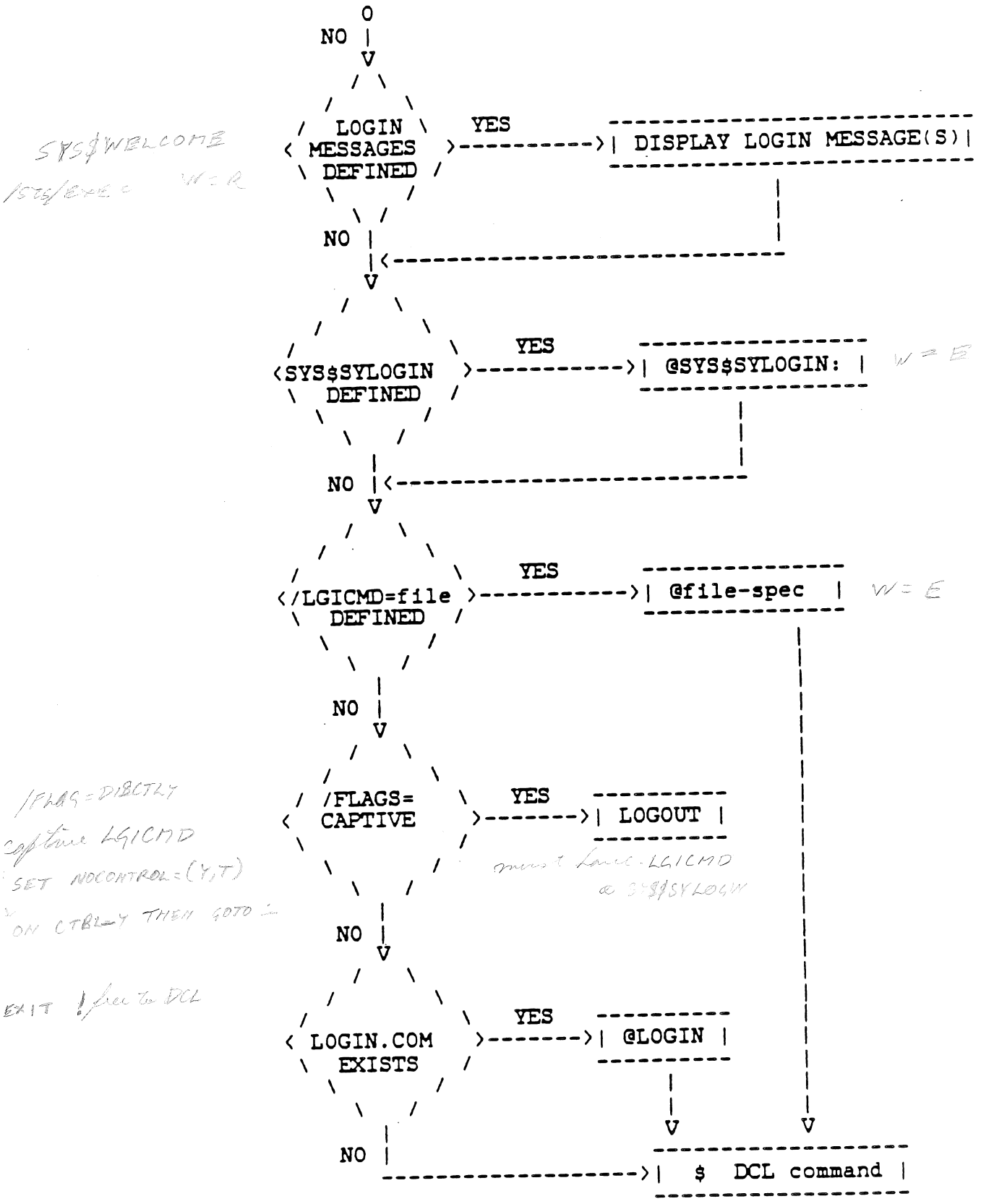


TABLE 3-1: CLASSES AND TYPES OF LOGINS

LOGIN CLASS	TYPE
LOCAL	INTERACTIVE
DIALUP	INTERACTIVE
REMOTE	INTERACTIVE
NETWORK	NONINTERACTIVE
BATCH	NONINTERACTIVE
DETACHED	DEPENDENT ON PARENT
SUBPROCESS	NONINTERACTIVE

LOCAL LOGINS

A local login is one that occurs from a terminal that is directly wired to the central processor. Local logins are always interactive. The sequence is as pictured in Figure 3-1. POSSIBLE TO CHECK FOR 'LTA' ... ALSO IN SYSLOGIN

DIALUP LOGINS

A dialup login is one that occurs from a terminal connected to a telephone line via a modem.

In order for a login to be classified dialup, the line that is connected to the modem or data switch must be set with the permanent terminal characteristic /DIALUP.

```
$ SET TERM/PERM/DIALUP/MODEM/HANGUP/DISCONNECT/AUTOBAUD TTC3:
```

The process involved in making the connection to the computer system will vary from system to system, but once you have connected to the VAX the sequence is the same as is pictured in Figure 3-1.

REMOTE LOGINS

A remote login is made by issuing the DCL command 'SET HOST node-name' which will allow you to connect to a node via DECnet. If the node is reachable, the login sequence will be similar to that in Figure 3-1.

NETWORK LOGINS

Network logins occur when you initiate a remote file access across DECnet or when you create some other type of network activity on a remote node.

Many DCL commands may specify a file or operation to be performed on a remote system.

Network logins are non-interactive although you may have to specify an access control string which will include your username and password for the remote node. Proxy logins are a special method of network logins that do not include an access control string.

Both your local and remote systems must be active nodes on the same network. Permitted access to your local system does not imply any access to the remote system.

Network logins have significant security considerations. This topic is discussed in more detail in Chapter 10.

BATCH LOGINS

A batch login is performed for you when a batch job that you submitted is ready to run. That is when it becomes current on the batch queue. If you submit a job to run after 7:00 P.M. with the following DCL command:

```
$ SUBMIT/AFTER=19:00 CHECKS.COM
```

The job will stay in the queue SYS$BATCH until sometime after 7:00 P.M. on the day the job is submitted. At that time the batch job controller will perform the batch login on behalf of the user and gain access to the procedure CHECKS.COM.

This would be a batch login, which is noninteractive. No password is provided although the batch process will have many of the attributes of the interactive process including UIC, privileges, and most quotas. The batch process will execute the same login command procedure(s) as those executed by the interactive process that submitted the batch job. Figure 3-2 pictures the execution sequence of a batch job which involves a batch login.

FIGURE 3-2: EXECUTION SEQUENCE OF A BATCH JOB

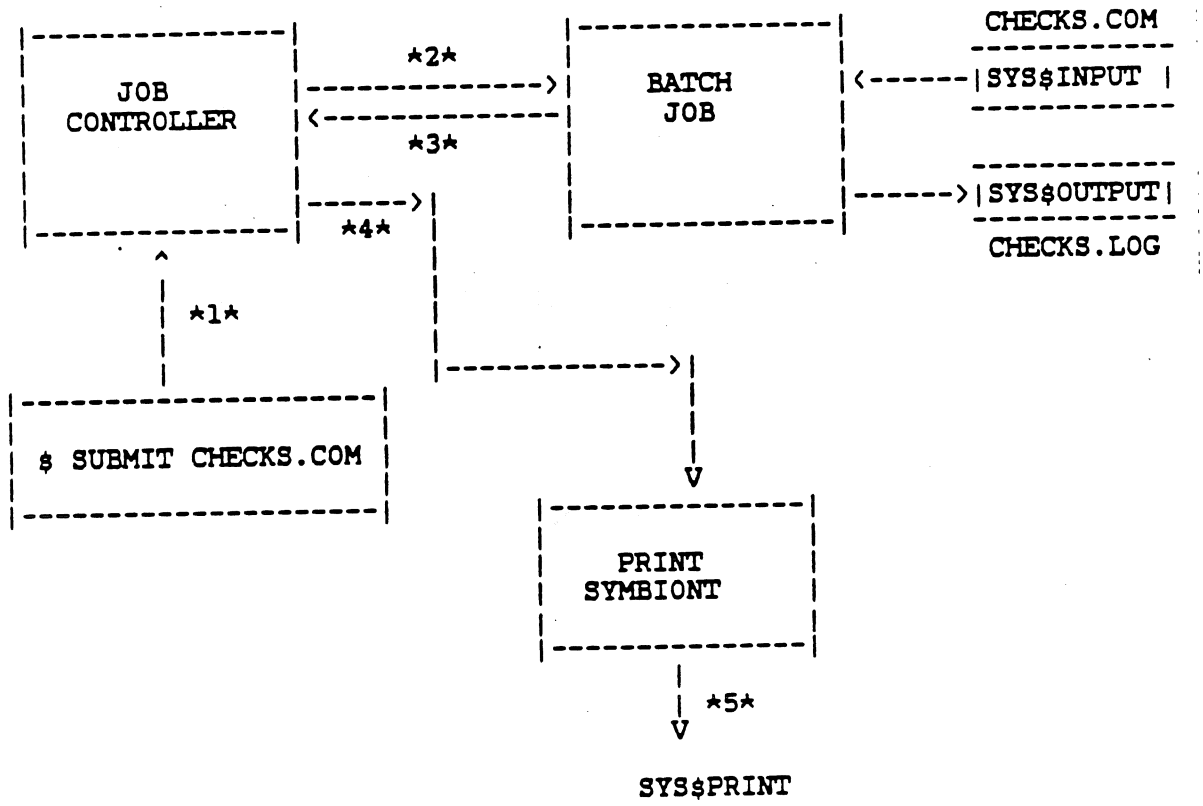


TABLE 3-2: CHARACTERISTICS OF DETACHED PROCESSES AND DETACHED PROCESS LOGIN

- INDEPENDENT OF CREATOR
- SHARE NO RESOURCES
- REQUIRES DETACH PRIVILEGE TO CREATE WITH DIFFERENT UIC
- REQUIRES NO PRIVILEGE TO CREATE WITH SAME UIC
- LIMITED BY PROCESS QUOTA MAX_DETACH

TABLE 3-3: METHODS OF CREATING DETACHED PROCESSES

- SYSTEM SERVICE $CREPRC UIC=[GROUP,MEMBER]
- DCL RUN/UIC=[GROUP,MEMBER] COMMAND
- DCL RUN/DETACH COMMAND

EXAMPLE 3-1: CREATING A DETACHED PROCESS WITH THE RUN COMMAND

```
$ RUN /UIC = [123,456]   SELECT.EXE
RUN-S-PROC_ID, identification of created process is 0000031B.

! requires DETACH privilege
```

```
$ RUN /DETACH   SELECT.EXE_
RUN-S-PROC_ID, identification of created process is 0000031C.

! same UIC and quotas as the current process
! no privilege required
! deducts from MAX_DETACH unless you have DETACH privilege
```

```
$ RUN /UIC=[123,456] /NOAUTHORIZE        -
      /OUTPUT=TTA6:  /INPUT=TTA6:     SYS$SYSTEM:LOGINOUT
RUN-S-PROC_ID, identification of created process is 0000031E.

! creates an interactive detached process
! requires DETACH privilege
```

TABLE 3-4: CHARACTERISTICS OF SUB PROCESSES AND SUBPROCESS LOGINS

- SUBPROCESS LOGIN IS ALWAYS NONINTERACTIVE
- DEPENDENT ON CREATOR
- RUN UNDER THE ACCOUNT OF THE CREATOR
- DELETED WHEN PARENT IS DELETED
- SHARES POOLED QUOTAS
- REQUIRES NO PRIVILEGE TO CREATE
- LIMITED BY PROCESS QUOTA PRCLM

TABLE 3-5: METHODS OF CREATING SUBPROCESSES

- SYSTEM SERVICE $CREPRC
- DCL COMMAND $ SPAWN
- RTL ROUTINE LIB$SPAWN
- DCL RUN/QUALIFIER COMMAND

NOTE

Any qualifier to the RUN command other than /DEBUG, /DETACH or /UIC=[group,member] will create a subprocess to run the image. See Table 3-6 for all run qualifiers.

Enabling certain characteristics for the subprocess that the parent does not posses will require privileges as noted in Table 3-6.

EXAMPLE 3-2: CREATING A SUB PROCESS

```
$ SPAWN   @SELECT
%DCL-S-SPAWNED, PROCESS FINCH_1 SPAWNED
%DCL-S-ATTACHED, TERMINAL NOW ATTACHED TO PROCESS FINCH_1
  OUTPUT FROM THE PROCEDURE
%DCL-S-RETURNED, CONTROL RETURNED TO PROCESS FINCH
$


$ SPAWN/NOWAIT   @SELECT
%DCL-S-SPAWNED, PROCESS FINCH_1 SPAWNED
$
  OUTPUT FROM THE PROCEDURE
$


$ RUN/PROCESS_NAME=SELECT_SUB   SELECT
%RUN-S-PROC_ID, IDENTIFICATION OF CREATED PROCESS IS 2080018B
$
   OUTPUT FROM SELECT.EXE
$
```

TABLE 3-6: QUALIFIERS TO THE $ RUN COMMAND

/AST_LIMIT=QUOTA	/BUFFER_LIMIT=QUOTA
/[NO]DEBUG	/DELAY=DELTA-TIME
/[NO]DUMP	/DETACH
/ENQUEUE_LIMIT=QUOTA	/ERROR=FILE-SPEC
/EXTENT=QUOTA	/FILE_LIMIT=QUOTA
/INPUT=FILE-SPEC	/INTERVAL=DELTA-TIME
/IO_BUFFERED=QUOTA	/IO_DIRECT=QUOTA
/JOB_TABLE_QUOTA=QUOTA	/MAILBOX=UNIT
/MAXIMUM_WORKING_SET=QUOTA	/OUTPUT=FILE-SPEC
/PAGE_FILE=QUOTA	
/PROCESS_NAME=PROCESS-NAME	/QUEUE_LIMIT=QUOTA
/[NO]RESOURCE_WAIT	/SCHEDULE=ABSOLUTE-TIME
/[NO]SERVICE_FAILURE	/SUBPROCESS_LIMIT=QUOTA
/TIME_LIMIT=LIMIT	/WORKING_SET=DEFAULT

/[NO]ACCOUNTING	(ACCNT PRIVILEGE)
/[NO]AUTHORIZE	(DETACH PRIVILEGE)
/PRIORITY=N	(ALTPRI PRIVILEGE)
/PRIVILEGES=(PRIVILEGE[,...]) =[NO]SAME	(SETPRV PRIVILEGE)
/[NO]SWAPPING	(PSWAPM PRIVILEGE)
/UIC=[G,M]	(DETACH PRIVILEGE)

3.2 INFORMATIONAL MESSAGES

All classes of interactive logins may receive a series of informational messages. Example 3-3 shows an interactive login session with all types of login messages displayed.

EXAMPLE 3-3: SUCCESSFUL INTERACTIVE LOGIN SESSION

```
                {ANNOUNCEMENT MSG}
      THIS IS NODE SEED ON VAXCLUSTER AVIARY

USERNAME: FINCH
PASSWORD:
PASSWORD:

                {DISCONNECTED JOB MSG}
 YOU HAVE THE FOLLOWING DISCONNECTED PROCESS:
 TERMINAL    PROCESS NAME     IMAGE NAME
 VTA52:      FINCH              (NONE)
 CONNECT TO ABOVE LISTED PROCESS [YES]:


                {WELCOME MSG}
 WELCOME TO VAX/VMS VERSION 4.0 ON NODE SEED

                {LAST LOGIN MSG 1}
 LAST INTERACTIVE LOGIN ON THURSDAY, 28-FEB-1985 10:20

                {LAST LOGIN MSG 2}
 LAST NON-INTERACTIVE LOGIN ON MONDAY, 25-FEB-1985 17:39

  2 FAILURES SINCE LAST SUCCESSFUL LOGIN  {LAST LOGIN MSG 3}

      YOU HAVE 1 NEW MAIL MESSAGE.          {NEW MAIL MSG}

$
```

ANNOUNCEMENT MESSAGE

The announcement message is displayed as soon as someone contacts the computer system (unless there is a system password). This message is used to indicate to the user trying to login what system they are accessing.

There are security implications in having this message enabled. Suppressing information about the system to which you are connected will make it a little more difficult for someone to break in to your system. The user will still receive the username/password prompt. However, it will also make it more confusing for authorized users particularly on a multi-node system.

This message is controlled by the logical name SYS$ANNOUNCE:. To have a message assign the logical name in one of the two methods shown in Example 3-4. If the logical name is not assigned then no announcement message will be displayed.

EXAMPLE 3-4: ASSIGNING SYS$ANNOUNCE LOGICAL NAME

```
$ ASSIGN/SYSTEM/EXEC "@SYS$MANAGER:ANNOUNCEMENTS.TXT"  SYS$ANNOUNCE:

                        or

$ ASSIGN/SYSTEM/TERM "This is NODE Seed On VAXCLUSTER Aviary"  SYS$ANNOUNCE:
```

NOTE

The logical name assignment should be placed in SYS$MANAGER:SYSTARTUP.COM.

The first method shown in Example 3-4 will type the contents of ANNOUNCEMENTS.TXT to the terminal. ANNOUNCEMENTS.TXT is a text file not a command procedure. The second method will display the quoted string on the terminal.

DISCONNECTED JOB MESSAGES

The disconnected job message will inform you after you login that your process was disconnected after your last successful login. It will prompt you and allow you to reconnect to that process. If you answer yes, your current process will be logged out and you will be connected to the disconnected process. This may also be accomplished by entering the DCL command $ CONNECT/CONTINUE.

The ability to reconnect to a process is a desirable feature and does not impose any special security considerations.

If your connection to the system is broken, you will be able to reconnect to that process for a time specified by the sysgen parameter TTY_TIMEOUT. This parameter sets the number of seconds before a process associated with a disconnected terminal is deleted.

If virtual terminals are enabled these messages can be disabled on a user basis with the UAF flag DISRECONNECT.

The following examples show the steps your must take in order to enable the use of virtual terminals.

Example 3-6 shows a terminal session where the process is disconnected and then reconnected.

EXAMPLE 3-5: ESTABLISHING THE USE OF VIRTUAL TERMINALS

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>CONNECT  VTA0: /NOADAPTOR /DRIVER=TTDRIVER
SYSGEN>EXIT

! for each terminal that will be used in this manner
! place the following command in SYS$MANAGER:SYSTARUP.COM
!
$ SET TERMINAL/PERMANENT/DISCONNECT   TTcu:
```

EXAMPLE 3-6: USING DISCONNECT/RECONNECT

```
$ EDIT NEW_WINGS.LIST
*I
        this is line 1 in the file
        now i will be disconnected ^Y
[EOB]
$ DISCONNECT

      This is Node SEED on VAXCLUSTER AVIARY

Username: FINCH
Password:

 You have the following disconnected process:
 Terminal    Process name     Image name
 VTA562:     FINCH        DUA0:[SYS0.SYSEXE]EDT.EXE
 Connect to above listed process [YES]:
 Connected to terminal VTA562:

 $ CONTINUE

         this is the last line of the file^Z
 [EOB]
 *EXIT
 WORK1:[FINCH]NEW_WINGS.LIST;1 3 lines


 $  TYPE NEW_WINGS.LIST
 this is line 1 in the file
 now i will be interrupted
 this is the last line of the file
 $
```

WELCOME MESSAGES

The welcome message is displayed once you have successfully logged on to the system. This message will display information about the system and optionally about other activities to all users.

This message may be suppressed for similar reasons as for suppressing the announcement message.

The welcome message is associated with the logical name SYS$WELCOME. This logical name is assigned in the same manner as SYS$ANNOUNCE.

To disable the printing of a message you must assign the logical name to string of blanks, as is done in Example 3-7

EXAMPLE 3-7: DISABLING SYS$WELCOME

```
$ ASSIGN/SYSTEM/EXEC" "  SYS$WELCOME
```

NOTE

If the logical name SYS$WELCOME is not assigned then a standard welcome message will be displayed. Example 3-3 is an example of the standard welcome message.

Individual users may have this message suppressed by setting the UAF flag DISWELCOME.

LAST LOGIN MESSAGES

There are three types of last login messages, and all or them are enabled or disabled as a group. Table 3-7 lists the types of last login messages.

TABLE 3-7: TYPES OF LAST LOGIN MESSAGES

- TIME OF LAST SUCCESSFUL INTERACTIVE LOGIN
- TIME OF LAST SUCCESSFUL NONINTERACTIVE LOGIN
- NUMBER OF LOGIN FAILURES TO THE USERNAME

The time of the last successful interactive login will always be displayed if these messages are enabled. After displaying the messages their values are reset.

From a security standpoint, these messages are valuable aids to increasing security. This is a chance for the user to note if someone is trying to break in or is actually using his/her account without the user's knowledge. In order for this method to be most effective, the user must note the messages and give thought as to whether he/she can account for the login failures, and the last login times. These messages also will concern unauthorized users since the intruder will know that logins are being monitored.

These messages can be disabled by the UAF flag DISREPORT.

NEW MAIL MESSAGES

The new mail message will inform the user of unread mail.

These message do not have security implications. You might want to disable these messages when mail is also disabled. This might be done for captive accounts.

The delivery of mail and the displaying of the messages is done by the UAF flags DISMAIL and DISNEWMAIL.

If an account is set DISMAIL, as well as denying the use of mail for that account, other users will not be able to send mail messages to that account.

EXAMPLE 3-8: DISABLING MESSAGES WITH AUTHORIZE

```
UAF> MODIFY FINCH -
     /FLAGS = (DISMAIL, DISNEWMAIL, DISRECONNECT, DISREPORT, DISWELCOME)
```

NOTE

Any of these flags may be removed (thus enabling the messages) by adding NO in front of the value. The DEFAULT account in AUTHORIZE has all messages and mail enabled.

3.3 LOGIN RESTRICTIONS

All accounts may be restricted from all classes of logins over different times and days.

TABLE 3-8: REASONS TO RESTRICT MODES & HOURS OF OPERATION

- DIALUP AND NETWORK ACCESS ARE MORE ANONYMOUS
- DIALUP ACCESS OPENS THE SYSTEM TO ANYONE WITH A TELEPHONE AND MODEM
- DIALUP AND NETWORK ACCESS ARE MORE TEMPTING FOR INTRUDERS
- CERTAIN DATA SHOULD BE ACCESSED AT YOUR SITE
- WORK/WORKERS NEED TO BE SUPERVISED
- BALANCE THE LOAD ON YOUR SYSTEM

SHIFT RESTRICTIONS

The times at which an account may be active can be restricted by several qualifiers in the UAF. Access to the system may be restricted by type of day and time of day.

There are two types of days, PRIMARY and SECONDARY. These are set with the UAF qualifier /PRIMEDAYS. The setting of the DEFAULT record in the UAF has Monday through Friday as primary days and Saturday and Sunday as secondary days.

For any hour on each type of day you may restrict or permit all access or certain classes of access. General access is permitted or denied with the /ACCESS qualifier to AUTHORIZE.

EXAMPLE 3-9: SETTING UP SHIFT RESTRICTIONS IN AUTHORIZE

```
/ACCESS                    ALLOWS UNRESTRICTED ACCESS

/NOACCESS=SECONDARY        ALLOWS ACCESS ON PRIMARY DAYS ONLY

/ACCESS=(9-17)             ALLOWS ACCESS FROM 9 THROUGH 6 ON ALL DAYS

/NOACCESS=(PRIMARY, 18-8, SECONDARY)

                           ALLOWS ACCESS FROM 9 THROUGH 6 ON PRIMARY
                           DAYS AND NO ACCESS ON SECONDARY DAYS

UAF>MODIFY FINCH /PRIMEDAYS = (NOSUN,NOMON,TUE,WED,THU,FRI,SAT) -
                 /ACCESS    = (PRIMARY, 9-11,14-18)              -
                 /NOACCESS  =  SECONDARY

                 PRIMARY DAYS ARE TUESDAY THROUGH SATURDAY AND
                 SECONDARY DAYS ARE SUNDAY AND MONDAY
                 ALL TYPES OF LOGINS ARE PERMITTED ON PRIMARY DAYS
                 BETWEEN 9 AM AND 12 NOON AND BETWEEN 2 PM AND 7 PM
                 NO ACCESS IS ALLOWED ON SECONDARY DAYS
```

LOGIN CLASS RESTRICTIONS

Class restrictions will limit classes of logins which serves to restrict the mode of operation. As with all access restrictions, the class restrictions may be done by hours of primary and secondary days. All interactive logins, batch logins or network logins may be restricted or dialup, remote, or local interactive logins may be indivdually allowed or restricted.

The qualifiers to AUTHORIZE which restrict classes of logins and their format are listed in Table 3-9. The interpretation of their values is the same as for the /ACCESS qualifier.

TABLE 3-9: AUTHORIZE QUALIFIER FORMAT FOR RESTRICTING CLASSES OF LOGINS

```
/[NO]LOCAL=
/[NO]NETWORK=
/[NO]REMOTE=
/[NO]DIALUP=
/[NO]BATCH=
/[NO]INTERACTIVE=
= ([PRIMARY], [N-M], [N] [,...] [SECONDARY], [N-M], [N] [,...])

/FLAGS = [NO]DISUSER     !RESTRICTS ALL ACCESS TO THE ACCOUNT
```

EXAMPLE 3-10: UAF RECORD WITH HOURLY AND CLASS RESTRICTIONS

```
UAF>ADD WARBLER /UIC=[350,15] /OWNER="LUCY WARBLER" /ACCOUNT=SONGS -
_               /DEVICE = WORK1: /DIRECTORY = [WARBLER]        -
_               /LOCAL =  (PRIMARY, 8-16, SECONDARY, 9-17) /BATCH -
_               /DIALUP = (PRIMARY ,17-7) /NOREMOTE /NONETWORK
_

%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier WARBLER value: [000350,000015]
added to RIGHTSLIST.DAT


UAF> SHOW WARBLER

Username: WARBLER                       Owner:  LUCY WARBLER
Account:  SONGS                         UIC:    [350,15] ([SONGS,WARBLER])
CLI:      DCL                           Tables: DCLTABLES
Default:  WORK1:[WARBLER]
Login Flags:
Primary days:    Mon Tue Wed Thu Fri     Sat Sun
Secondary days:                         Secondary 000000000011111111112222
Primary   000000000011111111112222      Day Hours 012345678901234567890123
Day Hours 012345678901234567890123                ----- No access  ------
Network:  ----- No access  ------                 ##### Full access ######
Batch:    ##### Full access ######                ---------#########------
Local:    --------#########-------                ----- No access  ------
Dialup:   ########---------#######                ----- No access  ------
Remote:   ----- No access  ------
Expiration:                (none)   Pwdminimum:  6   Login Fails:     0
Pwdlifetime:            180 00:00   Pwdchange:       (pre-expired)
Last Login:  (none) (interactive),                   (none) (non-interactive)
Maxjobs:          0  Fillm:          20  Bytlm:          8192
Maxacctjobs:      0  Shrfillm:        0  Pbytlm:            0
Maxdetach:        0  BIOlm:          18  JTquota:        1024
Prclm:            2  DIOlm:          18  WSdef:           150
Prio:             4  ASTlm:          24  WSquo:           200
Queprio:          0  TQElm:          10  WSextent:        500
CPU:         (none)  Enqlm:          30  Pgflquo:       10000
Authorized Privileges:
  TMPMBX NETMBX
Default Privileges:
  TMPMBX NETMBX
```

Example 3-10 shows how to add a user with various hourly and class restrictions and the resulting UAF record. User WARBLER will have local interactive access from 8-5 on primary days, 9-6 on secondary days and batch access all day long on both primary and secondary days. She will have dialup access from midnight till 8 am and 6 pm till midnight on primary days and no remote, nor network access at any time.

EXAMPLE 3-11: FAILED LOGIN MESSAGES

```
! CAPTIVE ACCOUNT WITH NO ACCESS TO CAPTIVE COMMAND PROCEDURE

USERNAME: WREN
PASSWORD:
%DCL-E-NOCMDPROC, ERROR OPENING CAPTIVE COMMAND PROCEDURE - ACCESS DENIED


! ACCOUNT SET /FLAGS=DISUSER

USERNAME: SPARROW
PASSWORD:
USER AUTHORIZATION FAILURE


! LOGIN ACCESS DENIED

USERNAME: ORIOLE
PASSWORD:
YOU ARE NOT AUTHORIZED TO LOGIN FROM THIS SOURCE
```

EXAMPLE 3-12: FAILED LOGINS = INVALID PASSWORD/USERNAME

```
USERNAME: FINCH                          {GIVE INCORRECT PRIMARY PASSWORD}
PASSWORD:
PASSWORD:
USER AUTHORIZATION FAILURE

USERNAME: FINCH
PASSWORD:                                {GIVE INCORRECT SECONDARY PASSWORD}
PASSWORD:
USER AUTHORIZATION FAILURE

USERNAME: FINCHES                        {GIVE INCORRECT USERNAME}
PASSWORD:
USER AUTHORIZATION FAILURE

USERNAME: FINCH                          {GIVE CORRECT INFORMATION}
PASSWORD:
PASSWORD:

  WELCOME TO VAX/VMS VERSION 4.0 ON NODE SEED
   LAST INTERACTIVE LOGIN ON TUESDAY, 26-FEB-1985 15:25
   2 FAILURES SINCE LAST SUCCESSFUL LOGIN
```

DIALUP RESTRICTIONS

Dialup logins impose a special security risk because they give the user an anonymity which tends to open up your system to more attacks. It also exposes you to anyone with a telephone, modem, a little knowledge and persistence aided by a microcomputer.

The most secure method is not to have any dial-in lines. This may be appropriate for high security environments. This does however severely restrict the use of authorized users and takes away a possibly desirable working environment.

VMS V4 provides you with some security tools which both counter someone trying to guess a password on your system over a dialup line and at the same time make dialing in easier for authorized users.

LOGIN RETRIES

The first thing you can control is the number of retry attempts the user is allowed for a login via dialup line. In this way, if the user mistypes part of the username or password the carrier will not automatically be lost. Instead, the user receives a grace period and a limited number of attempts to succeed. This is an important aid to unskillful and/or careless typists while at the same time it is a deterrent to the troublesome user who dials in with a known username and attempts to achieve a login by using a small computer programmed to enter every word in the dictionary as a password.

Two SYSGEN parameters LGI_RETRY_TMO and LGI_RETRY_LIM implement this form of control for dialups. The default values afford the users 3 retries with a 20 second interval between each. This means that users will only lose the carrier if they either fail to specify a valid password within three tries, or they spend more than 20 seconds between two of their tries.

NOTE

These values are system wide parameters and affect every user who tries to access the system via a dialup line. They are dynamic parameters so that they can be changed temporarily on a running system before making the change permanent.

Example 3-13 illustrates how you might set the total number of retry attempts to 6, allowing a half-minute interval between tries.

EXAMPLE 3-13: SETTING LGI PARAMETERS WITH SYSGEN

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  USE ACTIVE
SYSGEN>  SET LGI_RETRY_LIM 6
SYSGEN>  SET LGI_RETRY_TMO 30
SYSGEN>  WRITE ACTIVE      !WRITE CURRENT to make the change permanent
 !OPCOM message indicating current parameters were modified !
SYSGEN>  EXIT
$
```

NOTE

This method will not completely stop someone trying to break in. The user attempting to try all words in the dictionary as passwords or a series of common names as usernames is relentless, and redialing alone is not going to prove an effective deterrent.

3.4 BREAKIN DETECTION

LGI SYSGEN PARAMETERS

VMS offers additional methods of discouraging breakin attempts. These methods also use SYSGEN parameters in the LGI (Login Security Parameter) category.

Table 3-10 lists the LGI parameters, their effects and the unit in which they are measured.

TABLE 3-10: LGI PARAMETERS

PARAMETER	USE	UNITS
LGI_BRK_DISUSER	SETS UAF FLAG DISUSER WHEN AN ATTEMPTED BREAKIN IS NOTED	BOOLEAN
LGI_BRK_LIM	LIMIT ON LOGIN FAILURES BEFORE EVASIVE ACTION IS TAKEN	COUNT
LGI_BRK_TERM	CAUSES TERMINAL NAMES TO BE PART OF BREAKIN SOURCE	BOOLEAN
LGI_BRK_TMO	DELTA TIME INTERVAL FOR ACCUMULATING FAILURES	SECONDS
LGI_HID_TIM	INTERVAL FOR EVASIVE ACTION	SECONDS
LGI_PWD_TMO	TIME LIMIT TO ENTER SYSTEM PWD	SECONDS
LGI_RETRY_LIM	NUMBER OF RETRIES FOR DIALUPS	COUNT
LGI_RETRY_TMO	INTERVAL FOR DIALUP RETRIES	SECONDS

CIA BLOCK

VMS uses a data structure called the Compound Intrusion Analysis (CIA) block to contain information about suspected and known intruders.

TABLE 3-11: CIA BLOCK FIELDS

- CIA$K_TERMINAL	- Unknown user at terminal
- CIA$K_TERM_USER	- Known username at terminal
- CIA$K_NETWORK	- Network source
- CIA$K_USERNAME	- Username of parent process
- CIA$M_INTRUDER	
- CIA$W_COUNT	- Count of attempts
- CIA$K_LENGTH	- Length of CIA block
- CIA$L_FLINK	- Forward link to next block
- CIA$L_BLINK	- Backward link to previous block
- CIA$Q_TIME	- Expiration time of entry
- CIA$B_TYPE	- Structure type
- CIA$S_DATA	- Data area

SUSPECTS/INTRUDERS

LGI_BRK_LIM defines a threshold count for login failures that triggers suspicion that a breakin attempt is in progress. That is, when the count of login failures exceeds the LGI_BRK_LIM value, within a reasonable time interval, the system assumes a breakin is in progress. These failures must be from a specific source.

TABLE 3-12: SOURCES OF LOGIN FAILURES THAT EFFECT LGI_BRK_LIM

- A SPECIFIC TERMINAL AND A SPECIFIC VALID USERNAME
- A SPECIFIC REMOTE NODE AND A SPECIFIC REMOTE USERNAME
- THE USERNAME OF THE CREATOR OF A DETACHED PROCESS
- BY DEFAULT, USERNAMES AND TERMINALS ARE TRACKED TOGETHER
- INVALID USER NAMES ARE ALL COUNTED TOGETHER AS A SINGLE CLASS PER TERMINAL (USED TO TRIGGER ALARMS)
- ASSOCIATION OF TERMINAL TO USERNAME MADE BY THE SYSGEN PARAMETER LGI_BRK_TERM BEING SET TO 1
- VALUE OF 0 SAYS TO TREAT USERNAMES AS THE SOURCE OF A FAILURE REGARDLESS OF THE TERMINAL
- USEFUL WITH A DATA SWITCH, LAT-11, OR SIMILAR FACILITY

NOTE

Counting terminals and usernames in pairs raises the system's exposure but greatly reduces the problems that could arise regarding denial of service. Setting LGI_BRK_TERM to 0 allows you to put more emphasis on quickly detecting breakin attempts. Taking this action will however allow a determined malicious user to at least temporarily disable all known accounts.

TABLE 3-13: WHEN A SUSPECT BECOMES AN INTRUDER

- THE SPECIFIED TIME FOR WHICH VMS WILL REMEMBER LOGIN FAILURES IS AFFECTED BY SYSGEN PARAMETER LGI_BRK_TMO
- THE ACTUAL TIME INTERVAL WILL CHANGE UPWARDS AS MORE BREAKIN ATTEMPTS OCCUR FROM A SINGLE SOURCE
- DURING THE INTERVAL ALL FAILED ATTEMPTS FROM A SOURCE WILL BE SUMMED AND COMPARED TO LGI_BRK_LIM
- IF THE VALUE GOES OVER THE LIMIT THE SUSPECT BECOMES AN INTRUDER (EVASIVE ACTION TAKEN AND SECURITY ALARMS SET)
- COUNT AND THE INTERVAL ARE RESET WHEN THE INTERVAL EXPIRES
- LGI_BRK_TMO IS A DELTA TIME (SECONDS) WHICH IS ADDED TO THE INTERVAL FOR WHICH FAILURES ARE COUNTED FROM EACH SOURCE
- THE CUMULATIVE EFFECT IS :
  MORE FAILURES ⟹ INCREASE THE INTERVAL TO COUNT FAILURES
- DEFAULT VALUE OF LGI_BRK_LIM = 5
- DEFAULT VALUE OF LGI_BRK_TMO = 300 SEC (5 MIN)
- IMPLIES THAT THE SYSTEM TOLERATES AN AVERAGE RATE OF 1 LOGIN FAILURE EVERY 5 MINUTES
- EVASIVE ACTION TAKEN WHEN :
  - THE RATE EXCEEDS THAT RATE
  - THE 5 FAILURES LIMIT IS EXCEEDED

Table 3-14 pictures successive login failures from a terminal and the effects on the time interval and count of failures. Assume that LGI_BRK_LIM = 5 and LGI_BRK_TMO = 300 (5 minutes), their default values.

TABLE 3-14: WHEN BREAKIN ACTION IS TAKEN

LOGIN FAILURE	FAILURE COUNT	TIME INTERVAL	EVASIVE ACTION TAKEN
10:00:00	1	5 MIN	NO
10:00:30	2	9.5 MIN	NO
10:01:00	3	14 MIN	NO
10:15:00	COUNT AND TIME RESET NO ACTION TAKEN		
11:00:00	1	5 MIN	NO
11:01:00	2	9 MIN	NO
11:02:00	3	13 MIN	NO
11:03:00	4	17 MIN	NO
11:04:00	5	21 MIN	NO
11:05:00	6		YES!!!

NOTE

After evasive action is taken the failure count is set to BRK_LIM and the timeout interval is set to BRK_LIM * BRK_TMO. This serves to cause another failure to quickly restart evasive action.

EVASIVE SYSTEM ACTIONS

This action consists of not accepting any logins from the offending source for a period of time or disabling that account from logging in. If the parameter LGI_BRK_DISUSER is set to 1 then the evasive action will be to set the DISUSER flag in UAF record for that user. This must be manually reset before the user can log on again.

If the parameter is set to 0 then the user under suspicion is prohibited from logging in at that terminal or from a node for a period of time.

NOTE

Setting LGI_BRK_DISUSER does make the effects of breakin detection more severe. Its effects can be very severe and a determined malicious user can disable access to all known accounts.

Duration Of Evasive Action -

The duration of the evasive action is effected by the sysgen parameter LGI_HID_TIM.

The formula for computing the duration of the evasive action is:

EVASION TIME = LGI_HID_TIM * (RANDOM NUMBER)

1.0 <= RANDOM NUMBER <= 1.5

NOTE

Loopholes still exist here for the determined outsider. The outsider can modify his/her technique of attempting to guess passwords to perform the guessing over sufficiently long intervals, from different terminals, nodes, and/or usernames so as not to be detected.

Do not develop a false sense of security that simply because breakin detection had been implemented, there is no chance of an outsider breaking in to the system.

CHAPTER 4
CONTROLLING PASSWORDS

LIST OF TOPICS

- User Passwords
- Protecting Your Passwords
- Storage of Passwords
- Password Size
- Selecting Secure Passwords
- Changing Passwords
- Automatic Password Generator
- Primary and Secondary Passwords
- SYSTEM Passwords
- Preventing PASSWORD GRABBERS
  - Password Grabber Programs
  - Terminal Protection
  - Using the 'BREAK' Key
- Summary Of Secure Password Management

4.1 USER PASSWORDS

Passwords are strings of characters that a user must enter in order to validate his or her identity to VMS. This is a basic protection on the system, and good password control is a necessity in managing a secure system.

All non-open user accounts on a system have passwords. This password is required for logging in to the system. After you are prompted for a user name, the system will ask you for one and possibly a second password. The password you enter is not echoed on the terminal.

Passwords can contain from 1 to 31 characters. The valid characters are :

A through Z
a through z
0 through 9
$ (dollar sign)
_ (underscore)

NOTE

All lowercase characters are converted to uppercase before the password is encrypted.

The initial password is specified when an account is added with AUTHORIZE. The following examples ADD, COPY and RENAME an account and specify new passwords.

EXAMPLE 4-1: SPECIFYING PASSWORDS TO AUTHORIZE

```
UAF> ADD  FINCH /UIC=[350,35] /DIRECTORY=[FINCH]  -
                / PASSWORD = BRD345SEED

%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier FINCH value: [000350,000035]
added to RIGHTSLIST.DAT
```

```
UAF> ADD  ROBIN /UIC=[350,36] /DIRECTORY=[ROBIN] /FLAGS=GENPWD

%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier ROBIN value: [000350,000036]
added to RIGHTSLIST.DAT
```

```
UAF> COPY  FINCH  WREN /UIC=[350,37] /DIRECTORY=[WREN] -
                       /PASSWORD=GRAY987BILL /PWDEXPIRED

%UAF-I-COPMSG, user record copied
%UAF-I-RDBADDMSGU, identifier WREN value: [000350,000037]
added to RIGHTSLIST.DAT
```

```
UAF> COPY FINCH REDFINCH /UIC=[350,40] /DIRECTORY=[REDFINCH] -
                         /EXPIRATION=4-JUL-1986

%UAF-I-COPMSG, user record copied
%UAF-W-DEFPWD, Warning: copied or renamed records
must receive new password
%UAF-I-RDBADDMSGU, identifier REDFINCH value: [000350,000040]
added to RIGHTSLIST.DAT
```

NOTE

If you omit the /PASSWORD qualifier in the ADD command, the password defaults to USER. However you must specify a password when creating a new UAF record with the COPY or RENAME command.

4.2 PROTECTING YOUR PASSWORDS

Good maintenance of passwords by the user and security manager are essential for good system security. VMS provides control over several items relating to passwords that help you protect passwords. Any of these methods can be circumvented by sloppy practices of the users. Therefore education of the user is especially essential in this security measure. Items you must consider are selection of a password, both in size and type, and how often the password is changed.

A system manager can establish a minimum acceptable password length, and the maximum period of time that a password may remain unchanged. He or she can also fix the password so that the user cannot change it or must use a randomly generated password.

PASSWORDS ARE YOUR SYSTEMS BIGGEST EXPOSURE

USE DIFFERENT PASSWORDS ON DIFFERENT ACCOUNTS

4.3 STORAGE OF PASSWORDS

- ACTUAL PASSWORD IS NOT STORED ON THE SYSTEM
- SENT THROUGH A ONE-WAY ENCRYPTION ALGORITHM
  - DOES NOT USE A KEY
  - IS NOT REVERSIBLE
  - EVEN IF A USER WERE TO KNOW THE ENCRYPTED PASSWORD AND THE ALGORITHM USED, HE/SHE COULD NOT PRODUCE THE ACTUAL PASSWORD
- THE ENCRYPTED FORM IS STORED IN THE UAF (UAF$QPWD)
- ALL PASSWORDS ENTERED ARE ENCRYPTED AND COMPARED TO THE STORED VALUE
- PASSWORDS ARE NEVER DISPLAYED

See RECALL
& RECALL/ERASE

- USERS NEED TO REMEMBER THEIR PASSWORDS
- THIS IS INFLUENCED BY THE TYPES OF PASSWORDS THEY USE
- REMEMBERING IS ALSO INFLUENCED BY HOW THEY ARE EDUCATED REGARDING SECURITY

4.4 PASSWORD SIZE

The size and type of password will make guessing it more difficult. There are 38 different characters for a password and it may be up to 31 characters in size. Table 4-1 show the number of different possible passwords for a given password size. Using numbers as well as letters greatly increases the number of combinations for a password of a given size. A password of 6 characters has 309 million combinations with letters only, while there are over 3 billion combinations when numbers and special characters are added.

TABLE 4-1: PASSWORD COMBINATIONS VS. PASSWORD SIZE

PASSWORD SIZE	COMBINATIONS (LETTERS ONLY)	COMBINATIONS (ALL CHARACTERS)
1	26	38
2	676	1,444
3	17,576	54,872
4	456,976	2,085,136
5	12 million	79 million
6	309 million	3 billion
7	8 billion	114 billion
8	209 billion	4.3 trillion
9	5 trillion	165 trillion
10	141 trillion	6,278 trillion
15	1.7 E 21	5.0 E 30
20	2.0 E 27	4.0 E 32
25	2.4 E 35	3.1 E 39
30	2.8 E 42	2.5 E 47
31	7.3 E 43	9.4 E 48

AUTHORIZE provides you with methods of enforcing a minimum password length. The qualifier /PWDMINIMUM=value specifies minimum password length in characters (default is 6 for nonprivileged and 8 for privileged accounts). This value is enforced only at the user level by the DCL command SET PASSWORD. When you are creating a password in AUTHORIZE no minimum length requirement is enforced.

4.5 SELECTING SECURE PASSWORDS

It is never a good idea to use a password that can easily be guessed. A larger password will be harder to guess by brute force than a smaller one, but there are many passwords that can be guessed by someone who has a little knowledge of the particular user.

Table 4-2 lists very basic common passwords that should always be avoided.

TABLE 4-2: COMMON PASSWORDS TO AVOID

- YOUR NAME OR INITIALS
- THE NAME OF A FAMILY MEMBER, BOYFRIEND, GIRLFRIEND, ETC.
- THE NAME OR INITIALS OF YOUR COMPANY
- THE NAME OF WORK GROUPS OR PROJECTS
- THE NAME OF YOUR PET
- THE NAME OF YOU HOME TOWN, STREET ADDRESS, ZIP CODE
- OTHER ITEMS THAT HAVE A STRONG PERSONAL ASSOCIATION TO YOU SUCH AS AN ITEM FROM YOUR HOBBIES
- COMMON ENGLISH WORDS

NOTE

There are fewer than 100,000 English words in an average size English dictionary and fewer than 10% of those are in common use. Trying 10,000 or 100,000 combinations would not take someone with a microcomputer very long. Therefore in a medium to high risk environment, English words should not be chosen for passwords.

TABLE 4-3: CONCERNS WITH INITIAL PASSWORDS

- NEW ACCOUNTS ARE A SPECIAL CONCERN TO ANY SECURITY MANAGER
- MUST SOMEHOW TRANSMIT THE SPECIFICS OF THE ACCOUNT TO THE NEW USER
- MUST INCLUDE THE INITIAL PASSWORD
- COMMON PRACTICE IS TO USE THE USERS FIRST NAME
- PROVIDES A WELL KNOWN AND EASILY GUESSABLE PASSWORD
- TRANSMISSION OF ANY PASSWORD IS A WEAK SECURITY LINK
  - CAN INSIST THAT THE USER CHANGE HIS/HER PASSWORD AT FIRST LOGIN
  - AUTHORIZE QUALIFIER /PWDEXPIRED
  - DEFAULT IS /NOPWDEXPIRED
  - /PWD2_EXPIRED MARKS A SECONDARY PASSWORD AS EXPIRED
- CHECK ACCOUNTS AFTER CREATING THEM TO ASSURE THAT THEY HAVE BEEN USED AT LEAST ONCE

NOTE

Secondary passwords are discussed in section 4.8.

4.6 CHANGING PASSWORDS

To maintain secrecy, users should change their passwords from time to time. The SET PASSWORD command offers a means of making this change.

A system manager can control which users have the right to change their passwords. By adding the /LOCKPWD modifier to the UAF record the user will not be allowed to change his or her own password. This gives the system manager complete control over passwords but also puts a lot of possibly unnecessary burden on him or her. The default is /NOLOCKPWD which will be used at most sites. Many sites will insist that users change their passwords periodically.

TABLE 4-4: FEATURES OF PASSWORD EXPIRATION TIME

- PASSWORDS MAY HAVE A PASSWORD LIFETIME
- EXPIRE AT END OF LIFETIME
- EXPRESSED AS A DELTA TIME
- UAF QUALIFIER /PWDLIFETIME = DELTA-TIME

  DEFAULT VALUE = 180 00:00 (180 DAYS)

  (30 DAYS FOR PRIVILEGED ACCOUNTS)
- UAF> MODIFY FINCH/PWDLIFETIME = "60-"
- DISABLE IT WITH UAF QUALIFIER /NOPWDLIFETIME
- WHEN THE USER CHANGES HIS/HER PASSWORD THE EXPIRATION TIME IS RESET TO THE CURRENT TIME PLUS PWDLIFETIME

- WARNING MESSAGES AS YOU APPROACH EXPIRATION TIME
- IF THE SPECIFIED TIME HAS ELAPSED
  - A FINAL WARNING MESSAGE IS DISPLAYED
  - THE PASSWORD IS MARKED AS EXPIRED
  - ONE FINAL LOGIN PERMITTED AFTER EXPIRATION TIME
  - USER MUST CHANGE PASSWORD WITH $ SET PASSWORD

Lockin in VMS V5.0

- CHANGING PASSWORDS REGULARLY PROMOTES SECURITY
- PRIVILEGED ACCOUNTS - EVERY 1-2 MONTHS
- NONPRIVILEGED ACCOUNTS - EVERY 3-6 MONTHS
- STILL REQUIRES SOME COOPERATION OF THE USER

PASSWORDS ARE YOUR SYSTEMS BIGGEST EXPOSURE

EXAMPLE 4-2: WARNING MESSAGES CONCERNING PASSWORD EXPIRATION

```
Username: FINCH
Password:
 Welcome to VAX/VMS version V4.0 on node NEST
    Last interactive login on Thursday,  7-NOV-1985 8:54
WARNING - Your password expires on Friday,  8-NOV-1985 15:00
$ LOGOUT
  FINCH        logged out at  8-NOV-1985 13:52:20.64


Username: FINCH
Password:
 Welcome to VAX/VMS version V4.0 on node NEST                    -
    Last interactive login on Friday,  8-NOV-1985 13:51
WARNING - Your password has expired; update immediately with SET PASSWORI

$ SET PASSWORD
Old password:
New password:
Verification:
%SET-E-PWDNOTDIF, new password must be different from current password
$ SET PASSWORD
Old password:
New password:
Verification:
$ LOGOUT
  FINCH        logged out at  9-NOV-1984 15:05:28.23
```

EXAMPLE 4-3: MESSAGE AFTER PASSWORD HAS EXPIRED

```
Username: FINCH
Password:
Your password has expired - contact your system manager
```

NOTE

Both PRIMARY and SECONDARY passwords are subject to the same expiration times.

4.7 AUTOMATIC PASSWORD GENERATOR

- PASSWORDS MAY BE AUTOMATICALLY GENERATED
- THE PASSWORDS THAT ARE GENERATED WILL BE :
  - RANDOM
  - PRONOUNCABLE
  - NON-DUPLICATING (EXCEPT OVER A VERY LONG TIME)
- SYSTEM MANAGER CAN FORCE USE OF GENERATED PASSWORDS BY SETTING /FLAGS=GENPWD IN THE UAF
- USE $ SET PASSWORD/GENERATE=N COMMAND
- MAY SPECIFY A VALUE N (1<=N<=10)
- SET PASSWORD WILL GENERATE PASSWORDS OF FROM N TO (N+2) CHARACTERS IN LENGTH
- GREATER OF N AND THAT SPECIFIED BY PWDMINIMUM IN THE UAF WILL BE USED
- DEFAULT SIZE IS GREATER OF 6 OR /PWDMINIMUM
- UAF> /GENERATE_PASSWORD=(CURRENT|BOTH|PRIMARY|SECONDARY)

EXAMPLE 4-4: USING THE AUTOMATIC PASSWORD GENERATOR

```
$ SET PASSWORD/GENERATE=8
OLD PASSWORD:
EFWYKUAMOI          EF-WY-KU-A-MOI
GUZAWPVI            GU-ZAWP-VI
OORSOULT            OOR-SOULT
MUODTUMJO           MU-OD-TUM-JO
PENETDABZO          PE-NET-DAB-ZO
CHOOSE A PASSWORD FROM THIS LIST OR PRESS
RETURN TO GET A NEW LIST
NEW PASSWORD:
VERIFICATION:
$
```

Example 4-4 shows how a user would invoke the automatic password generator. Five random passwords between 8 and 10 characters each are displayed. One of them is selected and entered. If you do not like any of the passwords displayed, pressing RETURN will cause five new passwords to be displayed.

EXAMPLE 4-5: INVOKING THE AUTOMATIC PASSWORD GENERATOR IN AUTHORIZE

```
UAF> MODIFY FINCH/GENERATE_PASSWORD

PAUXLER             PAUX-LER
OVSNAWIHO           OV-SNA-WI-HO
BLUZZA              BLUZ-ZA
SABABBIN            SA-BAB-BIN
AMVONEXEPP          AM-VO-NEX-EPP

ENTER PRIMARY PASSWORD:
%UAF-I-MDFYMSG, USER RECORD(S) UPDATED
```

4.8 PRIMARY AND SECONDARY PASSWORDS

A secondary password provides a method of extra security that can be added at sites that have medium to high security needs. Once a secondary password has been established, it is prompted for after the primary password is entered at login time.

TABLE 4-5: SECONDARY PASSWORDS CHARACTERISTICS

- MAY BE ESTABLISHED FOR ANY ACCOUNT
- WILL RECEIVE TWO PASSWORD: PROMPTS WHEN LOGGING IN
- SET UP AN ACCOUNT, SUCH THAT TWO DIFFERENT PEOPLE WOULD BE REQUIRED TO ACCESS THAT ACCOUNT

  (EACH PERSON MUST KNOW ONE OF THE TWO PASSWORDS)
- CAN ASSURE AN ACCOUNT IS ACCESSED UNDER SUPERVISION
- FOR HIGH SECURITY SITES AND/OR SPECIALLY PRIVILEGED ACCOUNTS
- USE DCL COMMAND $ SET PASSWORD /SECONDARY
- MAY BE LOCKED WITH UAF /FLAGS = LOCKPWD

NOTE

Secondary passwords may not be specified in access control strings for file access over DECnet. Accounts that are using access control strings for remote file access may not require secondary password.

EXAMPLE 4-6: SPECIFYING SECONDARY PASSWORDS TO AUTHORIZE

To set both passwords :

```
UAF> MODIFY FINCH  /PASSWORD=(PASSWORD1, PASSWORD2)
```

To set the first password and nullify the second :

```
UAF> MODIFY FINCH  /PASSWORD=PASSWORD
```

To change only the first password :

```
UAF> MODIFY THRUSH /PASSWORD=(PASSWORD, "")
```

To change only the second password :

```
UAF> MODIFY WREN   /PASSWORD=("", PASSWORD)
```

To set both passwords to null, specify

```
UAF> MODIFY ORIOLE /NOPASSWORD
```

4.9 SYSTEM PASSWORDS

- OPTIONALLY CONTROL ACCESS TO SPECIFIC TERMINALS
  - DIALUP LINES
  - TERMINAL LINES USING PUBLIC DATA NETWORKS
  - LINES IN REMOTE TERMINAL ROOMS
  - TERMINALS RESERVED FOR PRIVILEGED OPERATION

- PROVIDES ADDED LEVEL OF PROTECTION ON AREAS SUBJECT TO ATTACKS
- DELIBERATELY CONFUSING
- EXTREMELY UNFRIENDLY
- ENTER BEFORE SYS$ANNOUNCE: AND PASSWORD PROMPT APPEARS
- NO PROMPT GIVEN / NO FAILURE MESSAGE GIVEN
- PASSWORD NOT ECHOED
- ENABLE OR DISABLE TERMINAL WITH DCL : REQUIRES LOG_IO OR PHY_IO PRIVILEGE

  $ SET TERMINAL /SYSPWD /PERMANENT

  $ SET TERMINAL /NOSYSPWD /PERMANENT

- SET PASSWORD WITH DCL OR AUTHORIZE:

  $ SET PASSWORD /SYSTEM REQUIRES SECURITY PRIVILEGE @ CMKRNL

  UAF> MODIFY /SYSTEM_PASSWORD=123BLUETEAL456 REQUIRES SYSPRV PRIVILEGE

- REMOTE (RTAn:), LAT-11 (LTAn:) AND X.25 TERMINAL CHARACTERISTICS SET WITH SYSGEN PARAMETER TTY_DEFCHAR2 (SEE TABLE 4-6)

TABLE 4-6: HOW TO SET TTY_DEFCHAR2 TO ENABLE SYSTEM PASSWORDS

1. ADD THE FOLLOWING LINES TO SYS$SYSTEM:MODPARAMS.DAT

```
!ENABLE SYSTEM PASSWORDS ON ALL TERMINALS BY DEFAULT
TTY_DEFCHAR2 = CHAR1 + CHAR2 + CHAR3 + %X80000
```

2. ADD THE FOLLOWING COMMAND TO SYS$MANAGER:SYSTARTUP.COM FOR ALL TERMINALS THAT WILL NOT HAVE A SYSTEM PASSWORD :

```
$ SET TERMINAL /NOSYSPWD /PERMANENT  TTA1:
```

PASSWORD OPA0:

3. MODIFY THE SYSTEM PARAMETERS AND REBOOT TO HAVE THE CHANGE PUT IN PLACE :

```
$ @SYS$UPDATE:AUTOGEN  SAVEPARAMS SETPARAMS
```

NOTE

The system password still must be set with DCL or AUTHORIZE

POSSUM
WOMBAT TTY_DEFCHAR2 = 4106. = %1002A plus %X80000 = ~~36874.~~ 36874.
PERCY 4106. 36866.
EAGLE 4098. = %1002

4106 = %1002A
%16100A = 528394

4.10 PREVENTING PASSWORD GRABBERS

TABLE 4-7: PASSWORD GRABBER PROGRAMS

- PROGRAM SHOWS EMPTY VIDEO SCREEN, SYSTEM JUST INITIALIZED, OR USER JUST LOGGED OUT
- READS FROM THE 'IDLE' TERMINAL WHEN USER PRESSES 'RETURN'
- GIVES USERNAME:/PASSWORD: PROMPT
- WRITES PASSWORD TO PERPETRATOR
- WRITES FAILURE MESSAGE AND GOES AWAY
- SEARCH PACK FOR "USERNAME:"
- ENCOURAGE USERS NOT TO LEAVE TERMINALS WHILE LOGGED ON
- ENCOURAGE USERS TO USE THE BREAK KEY

Table 4-7 lists the characteristics of a PASSWORD GRABBER program. It also lists some solutions. VMS's best method of guarding against these type of programs is the use of the Secure Terminal Server and the BREAK key and to set the appropriate terminal protection.

TABLE 4-8: HOW TO SET TERMINAL PROTECTION

- SETTING THE TERMINAL PROTECTION SO THAT THE WORLD HAS NO ACCESS
- TERMINAL OWNER SHOULD BE SYSTEM UIC
- CHANGE WITH DCL :

  $ SET PROTECTION = (S:R,O,G,W) /DEVICE /OWNER_UIC=[1,4] TTA1:
- DEFAULTS SET WITH SYSGEN PARAMETERS TTY_PROT AND TTY_OWNER
- PREVENTS USERS FROM ALLOCATING THE TERMINAL
- CAN STILL LOGIN
- ALLOCATED FROM PRIVILEGED PROCESS BY LOGINOUT

USING THE SECURE TERMINAL SERVER

VMS provides a secure terminal server that the security manager can invoke. The purpose of the secure server is to stop any executing process prior to the start of a login.

TABLE 4-9: SECURE TERMINAL SERVER

- INVOKE ON A TERMINAL-BY-TERMINAL BASIS, WITH :

  $ SET TERMINAL /PERMANENT /SECURE /DISCONNECT

- FORCES USER TO PRESS THE BREAK KEY FOLLOWED BY THE RETURN KEY TO INITIATE A LOGIN
- DELETES ANY ACTIVE PROCESS ON THAT TERMINAL
- THE LOGIN PROCEEDS AS USUAL
- YOU MAY WANT TO MAKE THE LOGIN PROCEDURE CONSISTENT THROUGHOUT THE SITE
- CERTAIN APPLICATIONS MAY NEED TO USE THE BREAK KEY FOR THEIR OWN PURPOSES
- TERMINAL SERVERS MAY USE THE BREAK KEY TO GET ATTENTION OF SYSTEM
- INCOMPATIBLE WITH AUTOBAUD

  $ SET TERM/PERM/NOSECURE/AUTOBAUD/MODEM/DISCONNECT/DIALUP . . .

NOTE

The SET TERMINAL commands should go in SYS$MANAGER:SYSTARTUP.COM.
This cannot be used with OPA0:.

4.11 SUMMARY OF SECURE PASSWORD MANAGEMENT

- CONTROL PASSWORDS ON SYSTEM ACCOUNTS
- DISABLE ACCOUNTS NO LONGER IN USE
- DO NOT LET SERVICE ORGANIZATIONS DICTATE THEIR PASSWORD SELECTION - KEEP ACCOUNTS /FLAGS=DISUSER
- USE DIFFERENT PASSWORD ON PRIVILEGED AND NONPRIVILEGED ACCOUNTS
- DO NOT USE THE SAME PASSWORDS ACROSS SYSTEM
- USE GENERATED PASSWORDS WHENEVER POSSIBLE
- ENFORCE A PASSWORD EXPIRATION TIME
- ENFORCE A MINIMUM PASSWORD SIZE
- DO NOT LEAVE ACCOUNT LISTINGS OF THE UAF IN AN ACCESSIBLE PLACE
- PROTECT THE UAF
- AVOID SHARING OF ONE ACCOUNT
- DO NOT LEAVE THE DIALUP TELEPHONE NUMBERS IN AN ACCESSIBLE PLACE
- MAKE ALL OPEN ACCOUNTS CAPTIVE

CHAPTER 5
MANAGING DISKS FOR SECURITY

LIST OF TOPICS

- PREVENTING DISK SCAVENGING
  - Erase on Delete - /ERASE Qualifier
  - Erase on Allocate - 'Highwater Marking'
  - Erase $QIO and $ERAPAT
- PHYSICAL SECURITY FOR DISKS AND BACKUPS
  - Security With BACKUP

5.1 PREVENTING DISK SCAVENGING

Disk scavenging is a method of obtaining information from deleted files on the disk. When a file is deleted the file header is changed to indicate that those blocks are free, and the header is freed up. The information on the disk is not actually erased or changed at all. All file protection schemes are gone once the file is deleted because the header has been released. Another user can come along and through very legitimate means have those blocks allocated to his/her file. This happens all the time, and normally the blocks are overwritten with information for the new file.

Disk scavenging can occur from insiders or outsiders. Most users will not look at blocks that are not being used by the current file, but the determined disk scavenger can glean information from those discarded blocks.

VMS V4 provides methods to effectiely prevent disk scavenging. These methods are erase on delete and erase on allocate. They are discussed in the next sections

TABLE 5-1: ERASE ON DELETE/PURGE

- ERASURE PATTERN WRITTEN TO BLOCKS OF DELETED FILES
- USERS CAN REQUEST IT SELECTIVELY
  - $ DELETE /ERASE FILE-SPEC
  - $ PURGE /ERASE FILE-SPEC
  - $ SET FILE /ERASE_ON_DELETE FILE-NAME
  - DEFAULT IS /NOERASE
- ENCOURAGE USERS TO USE IT ON KEY FILES
- CAN FORCE IT FOR ENTIRE VOLUME
  - $ SET VOLUME /ERASE_ON_DELETE DEVICE-NAME
  - FORCES ALL FILES DELETED OR PURGED TO BE ERASED
- $ INITIALIZE/ERASE DEVICE-NAME LABEL
  - ERASES ENTIRE DISK
  - ENABLES ERASE_ON_DELETE FOR VOLUME
  - VALID FOR FILES-11 ODS-2 DISKS AND ANSI TAPES
- SUBSTANTIAL OVERHEAD / BE SELECTIVE
- DEFAULT DATA SECURITY ERASE (DSE) PATTERN
  - PATTERN OF ZEROS
  - WRITES PATTERN ONCE
  - STILL LEAVES FAINT RESIDUAL THAT CAN BE READ WITH SPECIAL EQUIPMENT
  - CAN WRITE YOUR OWN PATTERN MORE THAN ONCE

TABLE 5-2: ERASE ON ALLOCATE = 'HIGHWATER MARKING'

- ATTEMPTS TO TRACK THE FURTHEST EXTENT WHERE FILE HAD BEEN WRITTEN AND PREVENT USERS FROM REACHING BEYOND THAT POINT
- IMPLEMENTED BY APPLYING SECURITY ERASURE PATTERN TO BLOCKS ALLOCATED FOR WRITING
- FILE IS PROTECTED TO ITS HIGH WATER MARK
- ENABLED BY DEFAULT
- REQUIRE ADDITIONAL I/O OPERATIONS
- SIGNIFICANT PERFORMANCE IMPACT (GREATEST ON FRAGMENTED DISKS)
- CAN TURN IT OFF
  - $ SET VOLUME/NOHIGHWATER DEVICE-NAME
  - $ INITIALIZE/NOHIGHWATER DEVICE-NAME LABEL

ERASE $QIO AND $ERAPAT

There are two system services that can be used to get and use an erasure pattern. The $ERAPAT system service (GET SECURITY ERASE PATTERN) will generate a security erasure pattern. User-written erase routines can then write this pattern into areas of memory, of the disk, or to a tape to erase sensitive data that is there. The writing will be accomplished by the $QIO system service with the IO$M_ERASE modifier.

The $ERAPAT service provides a consistent mechanism for performing security erase operations. This service is used primarily by VAX/VMS, but it may also be used by users.

TABLE 5-3: $ERAPAT SYSTEM SERVICE

SYS$ERAPAT TYPE ,COUNT ,PATADR

- TYPE
  - LONGWORD CONTAINING THE TYPE OF STORAGE TO BE WRITTEN OVER WITH THE ERASE PATTERN
  - THREE STORAGE TYPES ARE DEFINED BY THE $ERADEF MACRO

STORAGE TYPE	SYMBOLIC NAME
MAIN MEMORY	ERA$K_MEMORY
DISK	ERA$K_DISK
TAPE	ERA$K_TAPE

- COUNT
  - LONGWORD THAT CONTAINS THE NUMBER OF TIMES $ERAPAT HAS BEEN CALLED IN A SINGLE SECURITY ERASE OPERATION
  - SHOULD BE INITIALLY SET TO 1, THEN 2, AND SO ON, UNTIL THE STATUS CODE SS$_NOTRAN IS RETURNED
- PATADR
  - ADDRESS OF A LONGWORD INTO WHICH THE SECURITY ERASE PATTERN IS TO BE WRITTEN

TABLE 5-4: $QIO SYSTEM SERVICE

SYS$QIO[W] [EFN], CHAN, FUNC [,IOSB] [,ASTADR] [,ASTPRM] [,P1] [,P2] [,P3] [,P4] [,P5] [,P6]

- CHAN - I/O CHANNEL ASSIGNED TO THE DEVICE WHERE THE IO IS DIRECTED
- FUNC - DEVICE SPECIFIC FUNCTION CODE AND MODIFIERS SPECIFYING THE OPERATION TO BE PERFORMED
- WRITE FUNCTIONS
  - IO$_WRITEVBLK - WRITE VIRTUAL BLOCK
  - IO$_WRITELBLK - WRITE LOGICAL BLOCK
    LOG_IO PRIVILEGE REQUIRED
  - IO$_WRITEPBLK - WRITE PHYSICAL BLOCK
    PHY_IO PRIVILEGE REQUIRED
    (NON-DSA DISKS ONLY)
- IO$M_ERASE MODIFIER
  - WORKS FOR DISKS AND TAPES
  - PROPAGATES AN ERASE PATTERN THROUGH THE SPECIFIED RANGE
  - P1 - USER SPECIFIED ERASURE PATTERN
    IF P1=0 THEN ALL ZEROS ARE WRITTEN - THIS IS MORE EFFICIENT
  - P2 - NUMBER OF BYTES TO WRITE
    (ROUNDED UP TO MULTIPLES OF 512)
  - P3 - STARTING BLOCK NUMBER OF DATA TO BE ERASED

EXAMPLE 5-1: USING $ERAPAT AND $QIO

```
; Code fragment that erases 2000 blocks
; (blocks 1000 through 2999) on a disk
;
DISK_NAME:      .ASCID /GOOSE$DUA1:/
DISK_CHAN:      .WORD
PATTERN:        .LONG   0               ; write erasure pattern here
;
        $ERADEF                         ; Macro to define names
                                        ; used by $ERAPAT
        .ENTRY ERAPAT,^M<>
;
        $ASSIGN_S DEVNAM = DISK_NAME, - ; get a channel
                  CHAN   = DISK_CHAN
;
        MOVL    #1, R2                  ; Set initial count
;
ERAPAT_CALL:
        $ERAPAT_S    -                  ; call the $ERAPAT service
                COUNT=R2,-
                TYPE=#ERA$K_DISK,-
                PATADR=PATTERN

        BLBC    R0, EXIT                ; Branch if error

        CMPL    #SS$_NOTRAN, R0         ; Are we done?

        BEQL    EXIT                    ; Branch if so


QIO_CALL:
; call the $QIO service to write the erase pattern onto the disk

        $QIO_S  CHAN = DISK_CHA , -
                FUNC=#IO$_WRITELBLK!IO$M_ERASE,-
                P1=PATTERN,-            ; pattern to write
                P2=#<2000*512>,-        ; number of blocks to write
                P3=#1000                ; starting block_number

        INCL    R2                      ; Increase count

        BRB     ERAPAT_CALL             ; repeat the process

EXIT:   .
        .
        .
```

NOTES ON EXAMPLE 5-1

1. after each call to $ERAPAT, a test for the status SS$_NOTRAN is made

2. if SS$_NOTRAN has not been returned, $QIO is called to write the pattern returned by $ERAPAT onto the disk

3. each call to the $QIO writes the pattern one time

4. after each write, increment the count argument and call $ERAPAT again, until the code SS$_NOTRAN is returned

5. when SS$_NOTRAN is returned the security erase procedure is complete

6. provides a consistent mechanism for erasing disks, tapes or memory

7. see SYS$EXAMPLES:DOD_ERAPAT.MAR for information a routine to change the erasure pattern

5.2 PHYSICAL SECURITY FOR DISKS AND BACKUPS

Physical security of disk packs and tapes is essential. Access to the data will be much easier if a media can be removed from your site.

Disk scavenging may still be possible if a disk is removed from your site and the correct methods are applied to it.

Disk scavening may not be necessary if a media is taken from your site. Your protection scheme will be meaningless on another system.

The assumption is that your site is physically secure. You must also insure that your backup tapes are not available to users.

TABLE 5-5: SECURITY WITH BACKUP

- FILE SYSTEM TREATS A BACKUP SAVE SET AS A SINGLE FILE
- APPLIES TO BOTH DISK AND TAPE SAVE SETS
- BACKUP DOES NOT CHECK PROTECTION CODES OF INDIVIDUAL FILES WHEN READING A SAVESET
- ASSIGN A SAVE SET A RESTRICTIVE PROTECTION CODE

  $ BACKUP/QUALIFIERS

  FROM: FILE-SPEC(S)

  TO: SAVE-SET /OWNER_UIC=[UIC] /PROTECTION=(CODE)

- DO NOT GIVE USERS BACKUP TAPES
- RESTORE THE FILE(S) YOURSELF

  $ BACKUP/LOG

  FROM: MTA0:SAVESET/SELECT=[ROBIN.FEATHERS]*.DAT

  TO: WORK1:[*...] /OWNER_UIC=ORIGINAL

NOTE

This command will restore the file with its original directory, ownership, and protection. The file system will determine whether or not the user can access the file.

CHAPTER 6
FILE AND OBJECT PROTECTION METHOD

LIST OF TOPICS

- FILE PROTECTION AND THE SECURITY REFERENCE MONITOR
- HOW THE SYSTEM DETERMINES ACCESS
- OVERVIEW OF FILES-11 ODS-2
- UIC-BASED PROTECTION SYSTEM
  - Specifying UIC's
  - System Usage of UIC's
  - Determining UIC-based Protection
  - VOLUME, DIRECTORY, FILE Protection
- VMS PRIVILEGES AFFECTING PROTECTION
- ACCESS CONTROL LIST (ACL) - BASED PROTECTION
  - SYSTEM RIGHTS DATABASE
  - IDENTIFIERS
  - PROCESS RIGHTS LIST
- ACCESS CONTROL LISTS AND ACCESS CONTROL ENTRIES
  - Types of ACE's
  - Creating and Maintaining ACL's and ACE's
- DEFAULT PROTECTION AND OWNERSHIP
- MANAGING FILE PROTECTION

6.1 FILE PROTECTION METHODS AND THE SECURITY REFERENCE MONITOR

VMS V4 provides an expanded collection of file protection methods. These methods provide an important tool for enhancing system security. The basic method of SYSTEM, OWNER, GROUP,and WORLD (SOGW) protection from earlier versions of VMS still applies and will be sufficient for many sites. Access Control Lists (ACL's) have been added to the protection scheme to provide more capability and more flexibility in sharing and protecting files.

ACL's provide you with a greater amount of granularity in sharing files and other objects. Granularity refers to how fine the specification of who can and who can't access an object may be. VMS V4 enables you to allow or deny access to specific users from specific modes of operation.

Learning these new methods takes a little time but will be well worth the time spent for many users.

The new method of controlling access to files is based on a REFERENCE MONITOR MODEL. The reference monitor model uses OBJECTS, SUBJECTS, and IDENTIFIERS. The Reference Monitor Model was discussed in Chapter 2.

6.2 HOW THE SYSTEM DETERMINES ACCESS

The SECURITY REFERENCE MONITOR is invoked to make all access decisions. The steps taken to make an access decision are listed in Table 6-1.

NOTE

A user may be granted access at any step. Once granted, the user will receive access and will process the object. It is possible however to bypass the denial of access at a step by possessing a privilege or being the owner of the object.

TABLE 6-1: STEPS TAKEN IN MAKING ALL ACCESS REQUESTS

- IF THE OBJECT HAS AN ASSOCIATED ACL

  THEN

  - IF THE ACL GRANTS ACCESS
    THEN
    ACCESS IS GIVEN

  - IF THE ACL DENIES ACCESS
    THEN
    USE THE SYSTEM & OWNER FIELDS
    OF THE UIC-BASED PROTECTION

  - IF THE ACL DOES NOT EXPLICITLY DENY OR GRANT ACCESS
    THEN
    USE THE UIC-BASED PROTECTION

- IF THE OBJECT HAS NO ACL
  THEN
  USE THE UIC-BASED PROTECTION MASK

- IF THE AGENT HAS BYPASS, READALL, SYSPRV OR GRPPRV PRIVILEGE

  THEN

  THE AGENT MAY GAIN EXTRA ACCESS

6.3 OVERVIEW OF FILES-11 ODS-2

Disk files reside on FILES-11 disks. Files-11 refers to the logical structure given to the disk. It is a hierarchical organization of files, their data, and the directories needed to gain access to them. The VAX/VMS file system implements the disk structure and provides access control to the files located on the disk. RMS provides the internal organization of the files and methods of access to the data within the files.

TABLE 6-2: FILES-11 OVERVIEW

- ON DISK STRUCTURE LEVEL 1 (ODS-1) - RSX AND VMS
- ON DISK STRUCTURE LEVEL 2 (ODS-2) - VMS ONLY
- BLOCK
  - SMALLEST ADDRESSABLE UNIT OF INFORMATION
  - 512 BYTES
  - VIRTUAL BLOCK - FILE RELATIVE
  - LOGICAL BLOCK - DISK RELATIVE (ALWAYS 512 BYTES)
  - PHYSICAL BLOCK - DISK RELATIVE (MAY BE <512 BYTES)
- CLUSTERS
  - ONE OR MORE CONTIGUOUS BLOCKS
  - SMALLEST ALLOCATION UNIT FOR FILES
  - SIZE SET PER VOLUME WITH $ INIT/CLUSTER_SIZE = N
- EXTENTS
  - CONTIGUOUS CLUSTERS ALLOCATED TO A FILE

FILE AND OBJECT PROTECTION METHOD

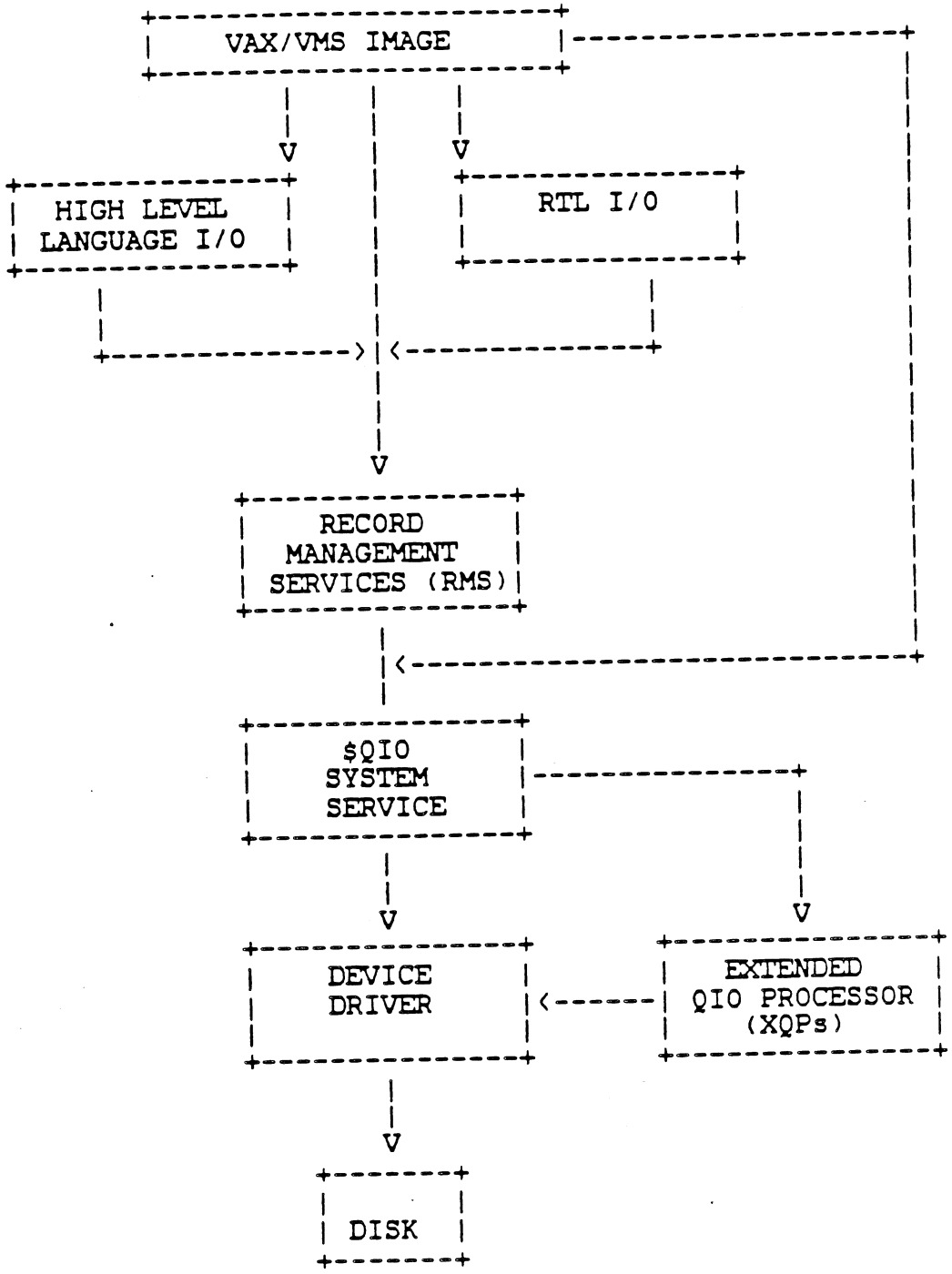
- FILE HEADER
  - CONTAINS DESCRIPTIVE INFORMATION (OWNER,PROTECTION,SIZE,ACL,DATES, . . .)
  - RETRIEVAL POINTERS TO EXTENTS

- INDEX FILE
  - CONTAINS ALL FILE HEADERS
  - [000000]INDEXF.SYS

- FILE DIRECTORY RECORD CONTAINS
  - FILE NAME
  - TYPE
  - VERSION NUMBER
  - FILE ID (POINTER TO FILE HEADER IN INDEX FILE)

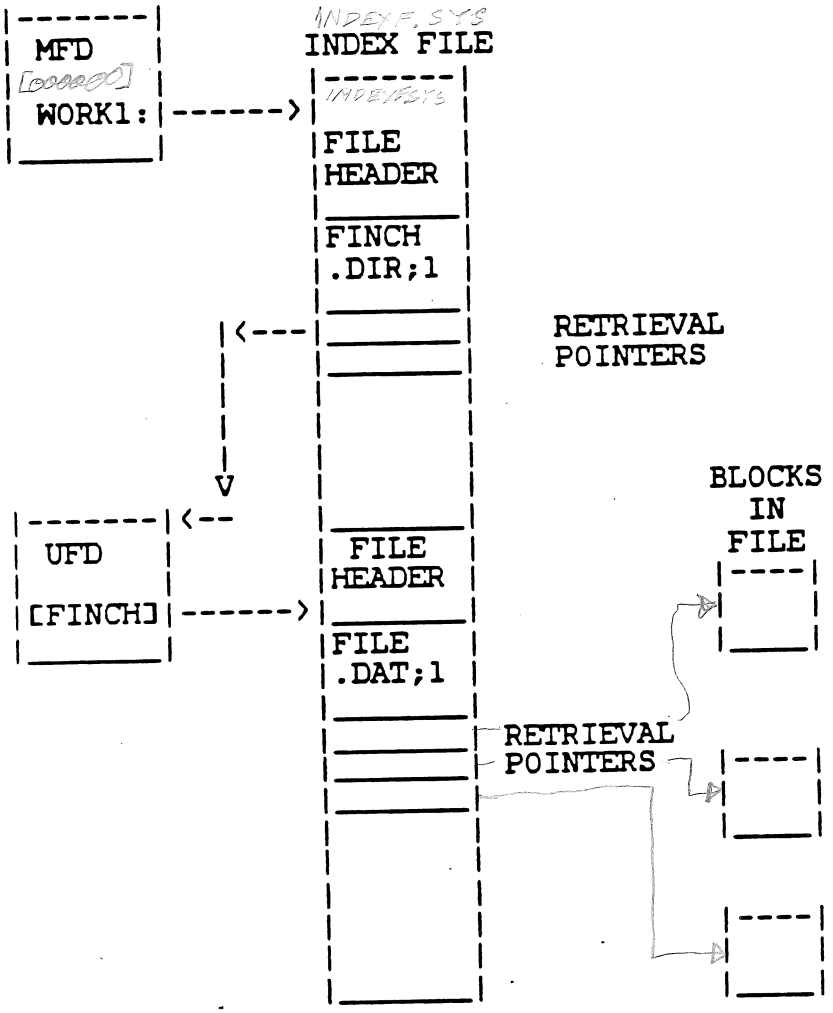
- FILES MAY BE ACCESSED VIA FILE ID
  - WILL BYPASS DIRECTORY PROTECTION
  - MCR PIP COMMAND OR THROUGH CALLING RMS ROUTINES

FILE AND OBJECT PROTECTION METHOD



FILE AND OBJECT PROTECTION METHOD

SEARCH FOR WORK1:[FINCH]FILE.DAT;1



EXAMPLE 6-1: DUMP/HEADER COMMAND

```
$ DUMP/HEADER FEATHERS.DAT / BLOCKS=(START=0,END=0)
Dump of file WORK1:[FINCH]FEATHERS.DAT;1 on 15-JAN-1985 13:21:19.65
File ID (12596,3,0)    End of file block 2 / Allocated 3

                              File Header

Header area
     Identification area offset:              40
     Map area offset:                         100
     Access control area offset:              215
     Reserved area offset:                    255
     Extension segment number:                0
     Structure level and version:             2, 1
     File identification:                     (12596,3,0)
     Extension file identification:           (0,0,0)
     VAX-11 RMS attributes

                    . . . .

     Access mode:                             0
     File owner UIC:                          [BIRDS,FINCH]
     File protection:                         S:RWED, O:RWED, G:RE, W:R
DIRECTOR pointer → Back link file identification:  (12594,4,0)
     Journal control flags:                   <none specified>
     Highest block written:                   3

Identification area
     File name:                               FEATHERS.DAT;1
     Revision number: # opened for write      3
     Creation date:                           15-JAN-1985 12:24:37.49
     Revision date:                           15-JAN-1985 13:18:47.06
     Expiration date:                         <none specified>
     Backup date:  /RECORD on BACKUP          <none specified>

Map area
     Retrieval pointers
          Count:          3          LBN:       46464

Access Control List
   (IDENTIFIER=SONG_BIRDS,ACCESS=READ+WRITE+EXECUTE+DELETE)
   (IDENTIFIER=PAYROLL_PROGRAMMERS+LOCAL,ACCESS=READ+WRITE)
   (IDENTIFIER=DIALUP,OPTIONS=PROTECTED,ACCESS=NONE)
   (IDENTIFIER=[300,*]+NETWORK,ACCESS=READ)
   (IDENTIFIER=WREN,OPTIONS=PROTECTED,ACCESS=NONE)

Checksum:                                     26727

Dump of file WORK1:[FINCH]FEATHERS.DAT;1 on 15-JAN-1985 13:21:19.65
File ID (12596,3,0)    End of file block 2 / Allocated 3

Virtual block number 1 (00000001), 512 (0200) bytes
```

6.4 UIC-BASED PROTECTION SYSTEM

When an account is created, two key factors affect the UIC-based protection scheme. Those factors which the system manager will establish for an account are the DEFAULT PROTECTION CODE FOR NEW FILES and the USER IDENTIFICATION CODE (UIC) given to the user.

This UIC-based protection is used in most cases. It will apply to protection for files, directories, volumes, and devices.

SPECIFYING UIC'S

UIC's can be specified in two manners, NUMERIC and ALPHANUMERIC format. Numeric format UIC's are described in Table 6-3 and alphanumeric format UIC's are described in Table 6-4. The corresponding AUTHORIZE commands to add the UIC to an account and to create an alphanumeric UIC are demonstrated in Example 6-5.

TABLE 6-3: NUMERIC FORMAT UIC'S

- [GROUP,MEMBER]
- BRACKETS ARE REQUIRED
- ASSIGNED WITH AUTHORIZE
- 1 <= GROUP <= 37776
- 0 <= MEMBER <= 177776

TABLE 6-4: ALPHANUMERIC FORMAT UIC'S

- [GROUP,MEMBER]
- [MEMBER]
- BRACKETS ARE REQUIRED
- ASSIGNED WITH AUTHORIZE
- GROUP NAME MAY BE TAKEN FROM ACCOUNT NAME
- GROUP NAME CAN BE CREATED WITH:

  UAF>ADD/IDENTIFIER/VALUE=UIC:[GROUP,*] IDENTIFIER

  UAF>ADD/IDENTIFIER/USER=[GROUP,*]
- MEMBER NAME WILL BE TAKEN FROM USERNAME
- GROUP AND MEMBER NAME <= 31 CHARACTERS (ALPHANUMERIC, '$', OR '_')
- MUST CONTAIN AT LEAST ONE LETTER
- CONVERT TO NUMBER WITH F$IDENTIFIER() LEXICAL FUNCTION

EXAMPLE 6-2: SAMPLE UICS

VALID	INVALID
[350,36]	[350,38]
[FINCH]	[350,177777]
[BIRDS,FINCH]	[WINGS,FINCH]
[WREN]	[C&P]
[2000,1]	[200000,1]

EXAMPLE 6-3: USING F$IDENTIFIER

```
! CONVERT A NAMED UIC IDENTIFIER TO A NUMBER

$ UIC  = F$IDENTIFIER ("FINCH","NAME_TO_NUMBER")
$ SHOW SYMBOL  UIC
  UIC = 15204381   HEX = 00E8001D  OCTAL = 00072000035

$ UIC = F$FAO ("!%U",UIC)
$ SHOW SYMBOL  UIC
  UIC = "[350,35]"



! CONVERT A NUMBERED UIC TO A NAME

$ UIC  = F$IDENTIFIER (%O4 + (%O1 *  %X10000),"NUMBER_TO_NAME")
$ SHOW SYMBOL  UIC
  UIC  = "SYSTEM"
```

SYSTEM USAGE OF UIC

TABLE 6-5: UIC TRANSLATION AND STORAGE

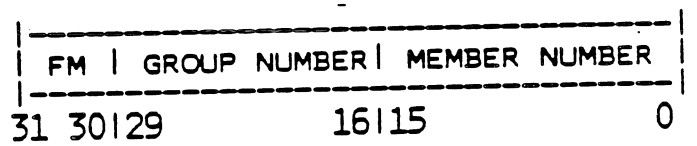
- TRANSLATED TO 32 BIT VALUE LONGWORD

  14 BIT GROUP NUMBER

  16 BIT MEMBER NUMBER

  2 BIT FORMAT IDENTIFIER

EXAMPLE 6-4: UIC FORMAT



- UIC's ARE CONTAINED IN THE UAF RECORD UAF$L_UIC
- ARE ALSO PART OF THE SOFTWARE CONTEXT OF THE PROCESS (PCB$L_UIC)
- ALPHANUMERIC UIC's EQUATED TO GROUP AND MEMBER PART OF A NUMERIC UIC
- EQUATED IN SYSTEM RIGHTS DATABASE AND PROCESS RIGHTS LIST
- IF AN ALPHANUMERIC UIC CONTAINS BOTH A GROUP AND MEMBER PART SYSTEM TRANSLATES MEMBER PORTION ONLY
- NO MEMBER CAN BE IN MORE THAN ONE GROUP
- EACH MEMBER NAME MUST BE UNIQUE FOR THE SYSTEM

EXAMPLE 6-5: ASSIGNING UIC'S TO NEW ACCOUNTS

```
UAF> ADD WREN/UIC=[350,37]
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier WREN value: [000350,000037]
added to RIGHTSLIST.DAT

UAF> SHOW WREN
Username: WREN                  Owner:
Account:                        UIC:    [350,37] ([WREN])
                        . . .


UAF> ADD/IDENTIFIER/VALUE=UIC:[350,*] BIRDS
%UAF-I-RDBADDMSGU, identifier BIRDS value: [000350,177777]

UAF> SHOW WREN
Username: WREN                  Owner:
Account:                        UIC:    [350,37] ([BIRDS,WREN])
                        . . .


UAF> ADD FINCH/UIC=[350,35]

UAF> SHOW FINCH
Username: FINCH                 Owner:
Account:                        UIC:    [350,35] ([BIRDS,FINCH])
                        . . .


UAF> SHOW/IDENTIFIER BIRDS
  Name                          Value           Attributes
  BIRDS                         [000350,177777] NORESOURCE

UAF> SHOW/IDENTIFIER/USER=[BIRDS,*]
  Name                          Value           Attributes
  FINCH                         [000350,000035] NORESOURCE
  WREN                          [000350,000037] NORESOURCE
                        . . .


UAF> ADD WARBLER/ACCOUNT=SONGS/UIC=[150,15]
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier WARBLER value: [000150,000015]
%UAF-I-RDBADDMSGU, identifier SONGS value: [000150,177777]

UAF> SHOW WARBLER
Username: WARBLER               Owner:
Account:  SONGS                 UIC:    [150,15] ([SONGS,WARBLER])
                        . . .
```

DETERMINING UIC-BASED PROTECTION

The system will often use the UIC-based protections, even when an ACL exists. The only exception to this is when an ACL immediately grants access to a particular user.

In determining whether or not to grant access via UIC-based protection, the system must first place the user in one or more of four categories (SYSTEM,OWNER,GROUP,WORLD) in relation to the object the agent is trying to access. Placement in one of these categories is made by comparing the UIC of the agent with the UIC of the object.

DETERMINING A USER'S CATEGORY FOR FILE ACCESS

- SYSTEM
  - USER'S UIC GROUP NUMBER <= MAXSYSGROUP (10 OCTAL)
  - USER WITH SYSPRV PRIVILEGE
  - USER WITH GRPPRV PRIVILEGE AND UIC GROUP NUMBER = GROUP NUMBER OF OBJECT
  - USER'S UIC = THE VOLUME'S OWNER UIC
  - INTENDED FOR SYSTEM MANAGERS, SECURITY MANAGERS, SYSTEM PROGRAMMERS, AND/OR OPERATORS

- OWNER
  - USER'S UIC GROUP AND MEMBER NUMBER = OBJECT'S UIC GROUP AND MEMBER NUMBER

- GROUP
  - USER'S UIC GROUP NUMBER = OBJECT'S UIC GROUP NUMBER (INCLUDES THE OWNER)

- WORLD
  - ALL USERS, REGARDLESS OF UIC

EXAMPLE 6-6: ILLUSTRATING PROTECTION CATEGORIES FOR A FILE

WORK1:[BIRDS]FEATHERS.DAT [350,35]

USER	UIC	USER CATEGORIES FOR FEATHERS.DAT
FINCH	[350,35]	OWNER, GROUP, WORLD
WREN	[350,36]	GROUP, WORLD
ROBIN	[350,3700]	GROUP, WORLD
PARROT	[35,35]	WORLD
PANDA	[6,200]	SYSTEM, WORLD
SYSTEM	[1,4]	SYSTEM, WORLD

ACCESS RIGHTS -

For each category of user, four access rights may be granted or denied under UIC-based protection. These are READ, WRITE, EXECUTE, and DELETE.

A fifth access right CONTROL is never specified by UIC-based protection but is always available to users in the OWNER and SYSTEM categories. CONTROL access gives the agent the ability to change the file protection and file characteristics.

EXAMPLE 6-7: UIC PROTECTION SPECIFICATION

SYSTEM:RWED, OWNER:RWED, GROUP:RE, WORLD:No Access

SYSTEM:RWE, OWNER:RWE, GROUP:No Access, WORLD:RWED

CHECKING USER CATEGORIES -

User categories are checked in the order OWNER >> WORLD >> GROUP >> SYSTEM. The system continues checking until the access is granted or all categories have been checked. That means that to deny a user category access to a resource you must deny that level and all outermost level (O >> G >> W).

In the second protection code in Example 6-7 all categories of users have READ, WRITE, EXECUTE, and DELETE access, since the WORLD is granted all access.

NOTE

The actual meaning of the access rights will vary depending on the type of object to which they are applied. This is displayed in Table 6-6

FILE AND OBJECT PROTECTION METHOD

VOLUME, DIRECTORY, FILE PROTECTION

TABLE 6-6: ACCESS TYPES MEANINGS

- DISK VOLUME

  - READ (R)

    READ, PRINT, COPY, OR EXECUTE FILES ON THE VOLUME

  - WRITE (W)

    MODIFY OR WRITE EXISTING FILES ON THE VOLUME

  - EXECUTE (E)

    CREATE FILES AND WRITE TO THOSE FILES ON THE VOLUME

  - DELETE (D)

    DELETE FILES ON THE VOLUME

  - CONTROL (C)

    CHANGE OWNERSHIP AND PROTECTION OF THE VOLUME

    I.E. $ SET VOLUME/OWNER=[UIC]/PROTECTION=CODE DDCU:

- DISK DIRECTORY

  - READ (R)

    EXAMINE OR LIST THE CONTENTS OF THE DIRECTORY

    I.E. $ DIRECTORY [BIRDS]

  - WRITE (W)

    WRITE TO OR MODIFY THE DIRECTORY FILE

  - READ AND WRITE

    RENAME OR CREATE FILES IN THE DIRECTORY

    V5.0 → DELETE FILES IN THE DIRECTORY
    (MUST ALSO HAVE DELETE ACCESS TO THE FILE)

  - EXECUTE (E)

    LIST FILES IN THE DIRECTORY BY SPECIFIC NAME

    I.E. $ DIRECTORY [BIRDS]RED_FEATHERS.DAT

  - DELETE (D)

    DELETE THE DIRECTORY FILE (IF IT IS EMPTY)

  - CONTROL (C)

    CHANGE THE PROTECTION AND FILE CHARACTERISTICS OF THE DIRECTORY FILE

FILE AND OBJECT PROTECTION METHOD

- DISK FILE

  - READ (R)

    READ, PRINT, COPY, OR EXECUTE THE FILE

  - WRITE (W)

    WRITE TO OR MODIFY THE FILE

    MUST ALSO HAVE READ ACCESS TO THE FILE

    REQUIRED TO CREATE NEW VERSION OF FILE

    (ALONG WITH WRITE TO .DIR FILE)

  - EXECUTE (E)

    EXECUTE EXECUTABLE IMAGES (.EXE) OR COMMAND PROCEDURES (.COM)

    I.E. $ RUN SELECT.EXE
    $ @SELECT.COM

  - DELETE (D)

    DELETE THE FILE, REMOVE entries, RENAME file
    V5.x

  - CONTROL (C)

    CHANGE THE PROTECTION AND FILE CHARACTERISTICS OF THE FILE

- TAPE VOLUMES
  - APPLIES TO ALL FILES ON THE VOLUME
  - ESTABLISHED/CHANGED AT VOLUME INITIALIZATION
  - SYSTEM AND OWNER ALWAYS HAVE READ AND WRITE ACCESS
  - READ (R)

    READ, PRINT, COPY, OR EXECUTE FILES ON THE TAPE VOLUME
  - WRITE (W)

    CREATE FILES ON THE TAPE VOLUME
  - EXECUTE (E), DELETE (D), CONTROL (C)

    DOES NOT APPLY

- NON-RECORD STRUCTURED DEVICES
  - READ (R)

    ALLOCATE THE DEVICE — Do Not allow W=R to Terminals (Password grabbers)
  - WRITE (W)

    WRITE TO THE DEVICE
  - LOGICAL (L), PHYSICAL (P)

    PERFORM A LOGICAL OR PHYCIAL I/O OPERATION TO THE DEVICE

- PRINT OR BATCH QUEUES

  - STANDARD UIC BASED PROTECTION ON QUEUES

  - READ (R)

    SEE ATTRIBUTES OF THE JOB

    (USER CAN ALWAYS SEE HIS/HER OWN JOBS)

  - WRITE (W)

    CAN SUBMIT JOBS TO THE QUEUE

  - EXECUTE (E)

    CAN ACT AS THE OPERATOR FOR THAT QUEUE
    (START, STOP, MODIFY ENTRIES)

    USERS WITH OPER PRIVILEGE HAVE EXECUTE ACCESS TO ALL QUEUES

  - DELETE (D)

    CAN DELETE JOBS FROM THE QUEUE
    (USER CAN ALWAYS DELETE HIS/HER OWN JOBS)

  - DEFAULT CODE GIVES WORLD:WRITE

```
$ INITIALIZE /QUEUE /PROTECTION=(SOGW CODE) /OWNER=UIC
$ START /QUEUE /PROTECTION=(SOGW CODE) /OWNER=UIC
```

TABLE 6-7: INITIALIZE DEVICE QUALIFIERS AFFECTING VOLUME PROTECTION AND OWNERSHIP

INITIALIZE /QUALIFIERS DEVICE-NAME[:] VOLUME-LABEL

/PROTECTION=CODE	
/OWNER_UIC=UIC	
/SYSTEM	OWNER UIC = [1,1] VOLUME PROTECTION = (S:RWED, O:RWED, G:RWED, W:RWED) MFD PROTECTION = (S:RWED, O:RWED, G:RWED, W) NEED SYSTEM UICs TO CREATE DIRECTORIES
/GROUP	OWNER UIC = [GROUP-NUMBER,0] VOLUME PROTECTION = (S:RWED, O:RWED, G:RWED, W:RWED) MFD PROTECTION = (S:RWED, O:RWED, G:RWED, W)
/GROUP/NOSHARE	OWNER UIC = [GROUP-NUMBER,0] PROTECTION = (S:RWED, O:RWED, G:RWED, W:No Access) MFD PROTECTION = (S:RWED, O:RWED, G:RWED, W)

NOTE

NO PRIVILEGE IS REQUIRED TO INITIALIZE A DISK THAT IS BLANK OR THAT YOU OWN, REGARDLESS OF THE QUALIFIER USED. FOR NON-BLANK DISKS THAT YOU DO NOT OWN VOLPRO PRIVILEGE IS REQUIRED.

TABLE 6-8: MOUNT QUALIFIERS AFFECTING VOLUME PROTECTION AND OWNERSHIP

MOUNT/QUALIFIERS DEVICE-NAME[:] [VOLUME-LABEL] [LOGICAL-NAME[:]]

/PROTECTION=CODE	SPECIFIES THE PROTECTION CODE USED WHILE THE VOLUME IS MOUNTED (FOR TAPES, ONLY APPLIES TO CONTINUATION VOLUMES OF MULTI-VOLUME SETS)
/OWNER_UIC=UIC	SPECIFIES THE OWNER UIC OF VOLUME TO USE WHILE THE VOLUME IS MOUNTED
/FOREIGN	INDICATES THAT THE VOLUME IS NOT IN STANDARD FORMAT PROVIDES NO FILE STRUCTURE OR FILE PROTECTION
/OVERRIDE=IDENTIFICATION	WILL MOUNT A VOLUME WITHOUT KNOWING THE VOLUME LABEL
/NOWRITE	PROVIDES READ-ONLY ACCESS TO ALL FILES ON THE VOLUME

NOTE

ANY QUALIFIER WHICH CHANGES CHARACTERISTICS OF THE VOLUME WHILE IT IS MOUNTED REQUIRES THAT THE PROCESS ISSUING THE MOUNT COMMAND BE THE OWNER OF THE VOLUME OR HAVE VOLPRO PRIVILEGE.

EXAMPLE 6-8: SEEING TAPE & DISK VOLUME PROTECTION

```
$ SHOW DEVICE/FULL DRA0:

Disk DRA0:, device type RM03, is online, mounted,
file-oriented device, shareable, error logging is enabled.

Error count                    0    Operations completed          19535
Owner process                 ""    Owner UIC                     [1,1]
Owner process ID        00000000    Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
Reference count               24    Default buffer size             512
Total blocks              131680    Sectors per track                32
Total cylinders              823    Tracks per cylinder               5

Volume label       "VAXVMSRL4"      Relative volume number            0
Cluster size                  1     Transaction count                68
Free blocks                9576     Maximum files allowed         32920
Extend quantity               5     Mount count                       1
Mount status             System     Cache name          "_DRA0:XQPCACHE"
Extent cache size             64    Maximum blocks in extent cache  957
File ID cache size            64    Blocks currently in extent cache 324
Quota cache size               0    Maximum buffers in FCP cache    152

Volume status:  subject to mount verification, file high-water marking,
    write-through caching enabled.

$ SHOW DEVICES/FULL MTA0:

Magtape MTA0:, device type TU77, is online, allocated, deallocate on
dismount, mounted, file-oriented device, error logging is enabled.

Error count                    0    Operations completed             45
Owner process          "FINCH"      Owner UIC                   [350,35]
Owner process ID        000000F5    Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
Reference count                2    Default buffer size           2048

Volume label           "FLYWAY"     Relative volume no.               0
Record size                   0     Transaction count                 1
Mount status            Process     Mount count                       1
ACP process name     "MTA0CACP"
Density                    1600     Format                    Normal-11

Volume status:  beginning-of-tape, odd parity.
```

EXAMPLE 6-9: SEEING TERMINAL DEVICE PROTECTION AND CHARACTERISTICS

```
$ SHOW DEVICE/FULL TTA4:

Terminal TTA4:, device type VT200, is online,
    record-oriented device, carriage control.

Error count                  0    Operations completed          1408
Owner process            "FINCH"  Owner UIC                [VMS,SYSTEM]
Owner process ID        0000011B  Dev Prot               S:RWLP,O:,G:,W:
Reference count              2    Default buffer size             80


$ SHOW DEVICE/FULL TTA6:

Terminal TTA6:, device type VT100, is online,
    record-oriented device, carriage control.

Error count                  0    Operations completed             5
Owner process                ""   Owner UIC                [VMS,SYSTEM]
Owner process ID        00000000  Dev Prot               S:RWLP,O:,G:,W:
Reference count              0    Default buffer size             80


$ SHOW TERMINAL TTA6:
Terminal: _TTA6:      Device_Type: VT200_Series  Owner: No Owner
   Input:   9600      LFfill:  0      Width:  80      Parity: None
   Output:  9600      CRfill:  0      Page:   24
Terminal Characteristics:
   Interactive         Echo                 Type_ahead        No Escape
   No Hostsync         TTsync               Lowercase         Tab
   Wrap                Scope                No Remote         Eightbit
   Broadcast           No Readsync          No Form           Fulldup
   No Modem            No Local_echo        No Autobaud       No Hangup
   No Brdcstmbx        No DMA               No Altypeahd      Set_speed
   Line Editing        Overstrike editing   No Fallback       No Dialup
   No Secure server    Disconnect           No Pasthru        No Syspassword
   No SIXEL Graphics   No Soft Characters   No Printer Port   Numeric Keypad
   ANSI_CRT            No Regis             No Block_mode     Advanced_video
   Edit_mode           DEC_CRT              DEC_CRT2
```

6.5 VMS PRIVILEGES AFFECTING PROTECTION

If a user holds one of certain privileges, the outcome of the protection check may be quite different from the scheme outlined above. Users with privileges are entitled to special access to objects throughout the system. For example, there is no way you could stop a user with the BYPASS privilege from accessing your files. The same applies to a user with SYSPRV or possessing a system UIC.

Most privileges are designed for trusted users who are involved in setting up the security monitor. There is no protection against a duly privileged user. The protection of your system depends on the judgment used in granting these privileges.

The privileges that most directly affect system security are SYSPRV, GRPPRV, BYPASS, and READALL. These are explained in the following sections.

SYSPRV

A user with SYSPRV privilege will receive the access accorded to users in the SYSTEM category. This means they will have control access to files, and even if the SYSTEM is denied access in the protection code, they can change both the protection code, and ownership of a file or directory.

You should exercise extreme caution in granting this privilege. A user with SYSPRV (or a system UIC) will have complete access to all objects on your system.

GRPPRV

A user with GRPPRV privilege, whose UIC group matches the group of the owner of the object, receives the same access accorded to users in the SYSTEM category. This means that a user with GRPPRV can change the ownership or protection of objects within the process's group. Thus, the user with GRPPRV privilege is able to manage a group's files.

This privilege should be granted only to users who function as group manager.

BYPASS

A user with BYPASS privilege receives all types of access to all files, regardless of its protection. He/she can bypass the the security reference monitor for that object.

You should use EXTREME CAUTION in exercising this privilege. It should be reserved for well tested command procedures or system backup operations. SYSPRV will be sufficient for all interactive work by system users.

still audited.

READALL

A user with READALL privilege receives READ and CONTROL access to the object, even if that access is denied by the ACL or UIC-based protection. In addition, the user may receive any other access that is granted through the protection code.

This privilege is useful for operators performing backups. It permits the operator to perform a BACKUP/RECORD without having WRITE access to the files.

You should exercise extreme caution in granting this privilege. A user with CONTROL access to objects can gain any type of access to that object. Therefore, a user with READALL privilege can gain access to any object on the system.

6.6 ACCESS CONTROL LIST (ACL) - BASED PROTECTION

VMS V4 provides an object protection method in addition to UIC-based protection. That is the use of access control lists or ACL's. ACL's provide a way to match the specific access you want to grant or deny to specific users of each object. It provides a much finer granularity of access control.

The key objects that VMS uses to institute the ACL-based protection are a RIGHTS DATABASE, IDENTIFIERS, and HOLDERS of the identifiers.

SYSTEM RIGHTS DATABASE

The reference monitor model specifies an authorization database, which describes all access authorizations in the system for all subjects and all objects. This database is often represented as an ACCESS MATRIX with subjects on one axis and objects on the other. The matrix indicates what access each subject is granted or denied for particular objects. VMS implements this access matrix by using the system rights database as the rows of the matrix and the ACL's as the columns.

The system rights database is a file that associates users of the system with identifiers those subjects can hold. Identifiers are special names the users hold. They are also associated with objects to indicate that holder of particular identifiers can or cannot access that object.

Characteristics of the system rights database are described in Table 6-9.

TABLE 6-9: SYSTEM RIGHTS DATABASE CHARACTERISTICS

- PERMANENT SYSTEM REGISTRY OF PROTECTION INFORMATION
- CONTAINS ALL IDENTIFIERS AND ASSOCIATED HOLDERS OF THOSE IDENTIFIERS
- CONTAINS INFORMATION ABOUT ATTRIBUTES OF IDENTIFIERS
- CONTAINS USERNAMES AND ASSOCIATED UIC'S
- CONTAINS ALPHANUMERIC UIC'S AND NUMERIC VALUE
- AN IDENTIFIER MUST EXISTS IN DATABASE BEFORE IT CAN BE USED ON THE SYSTEM
- STORED IN SYS$SYSTEM:RIGHTSLIST.DAT
- MAINTAINED WITH AUTHORIZE UTILITY
- PERMITS ESTABLISHMENT OF GROUPS THAT CROSS UIC BOUNDS

IDENTIFIERS

Identifiers are names established by VMS or by the security manager to be held by subject and by objects. Identifiers provide the means of specifying the users in an ACL. Possession of an identifier is what is used to grant or deny access via an ACL.

TABLE 6-10: IDENTIFIER CHARACTERISTICS

- IDENTIFIERS CONSIST OF A 32-BIT NUMBER AND A ALPHANUMERIC NAME

- 32-BIT NUMBER USED BY VMS

- ALPHANUMERIC NAME USED BY USER AND SYSTEM MANAGER

- THREE TYPES OF IDENTIFIERS
  - UIC IDENTIFIERS

    UNIQUELY IDENTIFIES USER TO THE SYSTEM
    NUMERIC OR ALPHANUMERIC FORMAT
    AUTOMATICALLY ADDED WHEN A USER IS ADDED TO THE UAF

  - GENERAL IDENTIFIERS

    DEFINED IN THE SYSTEM RIGHTS DATABASE
    <= 31 CHARACTERS (ALPHANUMERIC, '$', OR '_')
    CREATED AND ASSIGNED WITH AUTHORIZE
    ALLOWS ESTABLISHMENT OF USER GROUPS ACROSS UIC'S

    I.E. SONG_BIRDS, PROJECT23_MANAGERS, PAYROLL_PROGRAMMERS

  - SYSTEM-DEFINED IDENTIFIERS

    CREATED AND ASSIGNED BY VMS
    DESCRIBE CERTAIN TYPES OF USERS BASED ON THEIR USE OF THE SYSTEM

TABLE 6-11: SYSTEM-DEFINED IDENTIFIERS CORRESPONDING TO LOGIN CLASSES

INTERACTIVE

BATCH

NETWORK

LOCAL

DIALUP

REMOTE

NOTE

All users automatically become the holder of one of these identifiers when they access the system by that method. These identifiers are used for all access attempts by a process that was created via the particular type of login.

When using these identifiers in Access Control Lists, they should generally be seen as mutually exclusive. The INTERACTIVE identifier will appear with the identifier LOCAL, REMOTE or DIALUP when you are logged in interactively locally, through a remote terminal, or through a dial-up line. No others will appear in pairs.

All of these identifiers may be used in conjunction with general or UIC identifiers. For example, BATCH access may be denied for all members of UIC group [350,*].

EXAMPLE 6-10: USING AUTHORIZE TO CREATE AND GRANT GENERAL IDENTIFIERS

```
UAF> ADD/IDENTIFIER/ATTRIBUTE=RESOURCE  SONG_BIRDS
%UAF-I-RDBADDMSG, identifier SONG_BIRDS value: %X8001000B
added to RIGHTSLIST.DAT

UAF> ADD/IDENTIFIER  PAYROLL_PROGRAMMERS


UAF> SHOW/IDENTIFIER  PAYROLL_PROGRAMMERS
  Name                          Value       Attributes
  PAYROLL_PROGRAMMERS           %X8001000C  NORESOURCE


UAF> SHOW/IDENTIFIER/FULL  SONG_BIRDS
  Name                          Value       Attributes
  SONG_BIRDS                    %X8001000B  RESOURCE


UAF> GRANT/IDENTIFIER  SONG_BIRDS  FINCH
%UAF-I-GRANTMSG, identifier SONG_BIRDS granted to FINCH

UAF> GRANT/IDENTIFIER  SONG_BIRDS  [350,37]

UAF> GRANT/IDENTIFIER  SONG_BIRDS  ROBIN

UAF> GRANT/IDENTIFIER  SONG_BIRDS  WARBLER


UAF> SHOW/IDENTIFIER/FULL  SONG_BIRDS
  Name                          Value       Attributes
  SONG_BIRDS                    %X8001000B  RESOURCE
    Holder                        Attributes
    FINCH                         RESOURCE
    WREN                          RESOURCE
    ROBIN                         RESOURCE
    WARBLER                       RESOURCE


UAF> REVOKE/IDENTIFIER  SONG_BIRDS  WREN
%UAF-I-REVOKEMSG, identifier SONG_BIRDS revoked from WREN


UAF> SHOW/IDENTIFIER/FULL  SONG_BIRDS
  Name                          Value       Attributes
  SONG_BIRDS                    %X8001000B  RESOURCE
    Holder                        Attributes
    FINCH                         RESOURCE
    ROBIN                         RESOURCE
    WARBLER                       RESOURCE
```

EXAMPLE 6-11: USING AUTHORIZE TO MAINTAIN GENERAL IDENTIFIERS

```
UAF> GRANT/IDENTIFIER  PAYROLL_PROGRAMMERS  ROBIN

UAF> GRANT/IDENTIFIER  PAYROLL_PROGRAMMERS  [150,15]

UAF> GRANT/IDENTIFIER  PAYROLL_PROGRAMMERS  WREN

UAF> GRANT/IDENTIFIER  SONG_BIRDS  WREN

UAF> SHOW/RIGHTS/USER=[150,*]

Identifier                          Value          Attributes
Identifiers held by ROBIN :
  SONG_BIRDS                        %X8001000B     RESOURCE
  PAYROLL_PROGRAMMERS               %X8001000C     NORESOURCE

Identifiers held by WARBLER :
  SONG_BIRDS                        %X8001000B     RESOURCE
  PAYROLL_PROGRAMMERS               %X8001000C     NORESOURCE


UAF> SHOW/RIGHTS  WREN

Identifier                          Value          Attributes
  PAYROLL_PROGRAMMERS               %X8001000C     NORESOURCE
  SONG_BIRDS                        %X8001000B     RESOURCE


UAF> RENAME/IDENTIFIER  PAYROLL_PROGRAMMERS  PERSONNEL_PROGRAMMERS
%UAF-I-RDBMDFYMSG, identifier PAYROLL_PROGRAMMERS modified

UAF> SHOW/RIGHTS  WREN

Identifier                          Value          Attributes
  PERSONNEL_PROGRAMMERS             %X8001000C     NORESOURCE
  SONG_BIRDS                        %X8001000B     RESOURCE
```

NOTE

You cannot use wildcards to grant general identifiers.

PROCESS RIGHTS LIST

The Process Rights List is the active list of all identifiers a process holds. It is maintained in memory and used to make all protection checks. Table 6-12 lists the characteristics of the Process Rights List.

TABLE 6-12: PROCESS RIGHTS LIST CHARACTERISTICS

- ONE PER PROCESS
- HOLDS PROCESS'S IDENTIFIERS
- USED FOR ALL PROTECTION CHECKS
- CREATED AT PROCESS CREATION TIME
- STORED IN NONPAGED POOL
- POINTED TO BY A FIELD OF THE PROCESS CONTROL BLOCK (PCB$L_ARB)
- CALLED ACCESS RIGHTS BLOCK (ARB)
- ACCESSED BY MULTIPLE DATA STRUCTURES : IO REQUEST PACKETS, UNIT CONTROL BLOCKS, GLOBAL SECTION DESCRIPTORS, KNOWN FILE ENTRY LISTS
- COPIES IDENTIFIERS FROM SYSTEM RIGHTS DATABASE INCLUDING YOUR UIC
- ADDITIONAL IDENTIFIERS PUT THERE BY LOGIN SOFTWARE OR SITE SPECIFIC SOFTWARE
- MAY SEE WITH DCL COMMAND : $ SHOW PROCESS/PRIVILEGES

TABLE 6-13: ACCESS RIGHTS BLOCK (ARB) PICTURE

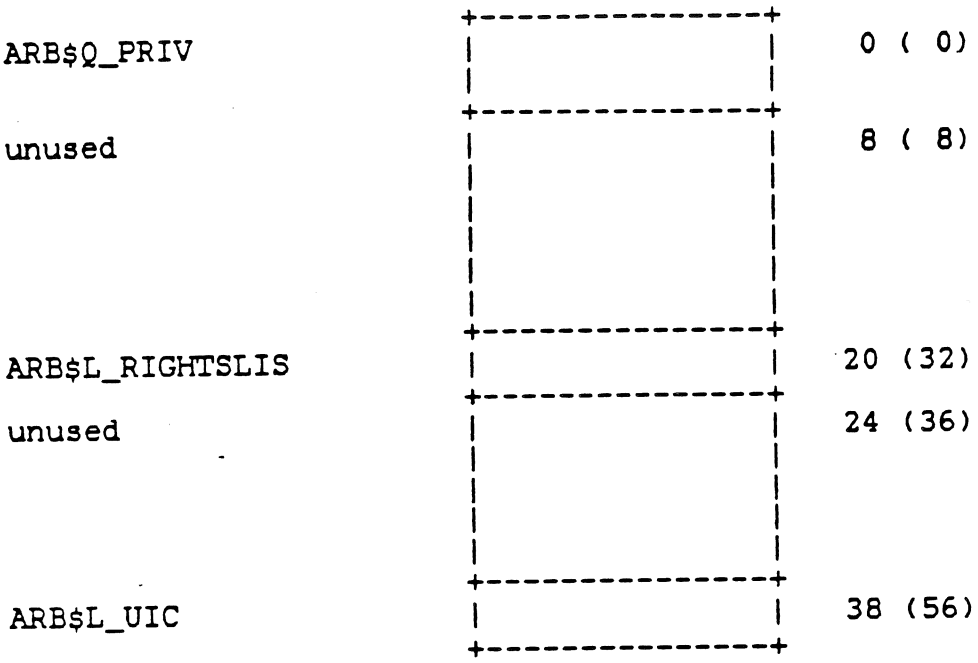


TABLE 6-14: DESCRIPTION OF ARB FIELD

Fields of ARB	OFFSET	DESCRIPTION
ARB$Q_PRIV	0	Privilege mask
ARB$R_CLASS	12	Security classification mask *
ARB$L_RIGHTSLIST	32	Pointer to rights list descriptor
ARB$R_RIGHTSDESC	48	Descriptor for local rights list*
ARB$R_LOCALRIGHTS	56	Process local rights list*
ARB$L_UIC	56	User identification code

*for Secure VMS SEVM
Non discretionary control

EXAMPLE 6-12: SEEING PROCESS RIGHTS

```
$ SHOW PROCESS /PRIVILEGES

15-JAN-1985 10:58:23.42   VTA202:                 User: FINCH

Process privileges:
  GRPNAM                  may insert in group logical name table
  TMPMBX                  may create temporary mailbox
  EXQUOTA                 may exceed quota
  NETMBX                  may create network device

Process rights identifiers:
  INTERACTIVE
  LOCAL
  SONG_BIRDS
  PAYROLL_PROGRAMMERS
```

EXAMPLE 6-13: SEEING PROCESS UIC

```
$ SHOW PROCESS

15-JAN-1985 10:58:42.04   VTA202:           User: FINCH
Pid: 2180013F    Proc. name: FINCH          UIC: [BIRDS,FINCH]
Priority:    4   Default file spec: WORK1:[FINCH]
Devices allocated: VTA202:
```

6.7 ACCESS CONTROL LISTS AND ACCESS CONTROL ENTRIES

Access Control Lists (ACL's) are composed of lists of one or more Access Control list Entries (ACE's).

TABLE 6-15: ACL'S AND ACE'S

- ACL'S ARE LISTS OF ACE'S
- EACH ACL IS ASSOCIATED WITH A SPECIFIC OBJECT
- EACH ACE CONTAINS IDENTIFIER(S) AND ACCESS GRANTED OR DENIED TO HOLDERS OF THOSE IDENTIFIERS
- CREATE WITH DCL COMMANDS OR ACL EDITOR
  - $ SET ACL
  - $ SET DIRECTORY/ACL
  - $ SET DEVICE/ACL
  - $ SET FILE/ACL
  - $ EDIT/ACL FILE-NAME
    EDIT/ACL /OBJECT= type
- SEE WITH DCL COMMANDS OR ACL EDITOR
  - $ SHOW ACL
  - $ DIRECTORY/ACL
  - $ DIRECTORY/SECURITY
  - $ DIRECTORY/FULL
  - $ EDIT/ACL FILE-NAME
- THREE TYPES OF ACE'S
  - IDENTIFIER ACE
  - DEFAULT PROTECTION ACE
  - SECURITY ALARM ACE
  - information ACE

IDENTIFIER ACE

An Identifier Ace controls the type of access a particular user or groups of users is allowed to a particular object.

FORMAT :
(IDENTIFIER=ID[,OPTIONS=OPTION],ACCESS=ACCESS-TYPE)

OPTIONS

- DEFAULT *FOR ANY NEW FILES CREATED IN DIRECTORY*
  - VALID ONLY ON DIRECTORIES
  - SPECIFIES THE ACE THAT IS APPLIED TO ALL FILES CREATED WITHIN THE DIRECTORY
  - DEFAULT OPTION IS REMOVED WHEN THE ACE IS PROPAGATED
  - HAS NO EFFECT ON CONTROLLING ACCESS TO THE DIRECTORY ITSELF
- PROTECTED
  - CAN ONLY DELETE ENTIRE ACL WITH ACL EDITOR OR $ SET ACL/DELETE/ACL=(ACE)
- NOPROPAGATE *To New Versions of this file.*
  - ACE IS NOT PROPAGATED WHEN COPYING THE FILE
  - DEFAULT IS TO PROPAGATE ACE TO NEW VERSIONS
  - CANNOT SPECIFY PROPAGATE
- NONE
  - NO OPTIONS ARE APPLIED
  - NOT DISPLAYED WITH ACL
  - IMPLIES PROPAGATE

TABLE 6-16: IDENTIFIER ACE ACCESS TYPES

- READ
  - CAN READ A FILE
  - CAN READ FROM A DISK
  - CAN ALLOCATE A DEVICE
- WRITE
  - CAN READ OR WRITE TO A FILE
- EXECUTE
  - CAN EXECUTE AN IMAGE FILE *OR COMMAND PROCEDURE*
  - CAN LOOKUP FILES IN A DIRECTORY WITHOUT USING WILDCARDS
- DELETE
  - CAN DELETE A FILE
- CONTROL
  - HAS THE SAME ACCESS/PRIVILEGES AS THE OWNER OF THE OBJECT
- NONE
  - HAD NO ACCESS TO THE OBJECT – *unless object has SYSTEM or OWNER UIC access.*

EXAMPLE 6-14: EFFECTS OF IDENTIFIER ACE OPTIONS

```
$ SHOW ACL BIRDSEED.FORMULA
Object type: file,  Object name: WORK1:[FINCH]BIRDSEED.FORMULA;1,
on 15-JAN-1985 13:14:08.96
     (IDENTIFIER=[BIRDS,WREN]+DIALUP,ACCESS=NONE)
     (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)
     (IDENTIFIER=PAYROLL_PROGRAMMERS,OPTIONS=NOPROPAGATE,ACCESS=NONE)


$ COPY  BIRDSEED.FORMULA;1    *.*;


$ COPY  BIRDSEED.FORMULA;1    BIRDSEED.FORMULA;


$ DIRECTORY/ACL BIRDSEED.FORMULA

Directory WORK1:[FINCH]

BIRDSEED.FORMULA;3
     (IDENTIFIER=[BIRDS,WREN]+DIALUP,ACCESS=NONE)
     (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)
BIRDSEED.FORMULA;2
     (IDENTIFIER=[BIRDS,WREN]+DIALUP,ACCESS=NONE)
     (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)
BIRDSEED.FORMULA;1
     (IDENTIFIER=[BIRDS,WREN]+DIALUP,ACCESS=NONE)
     (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)
     (IDENTIFIER=PAYROLL_PROGRAMMERS,OPTIONS=NOPROPAGATE,ACCESS=NONE)

Total of 3 files.


$ COPY  BIRDSEED.FORMULA;1    NEW_BIRDSEED.FORMULA


$ SHOW  ACL  NEW_BIRDSEED.FORMULA;*
%RMS-F-WLD, invalid wildcard operation


$ SHOW  ACL  NEW_BIRDSEED.FORMULA
%SYSTEM-W-ACLEMPTY, access control list is empty
```

FILE AND OBJECT PROTECTION METHOD

```
$ SET ACL/LOG/LIKE=BIRDSEED.FORMULA;1       -
                                  NEW_BIRDSEED.FORMULA
%SET-I-MODIFIED, WORK1:[FINCH]NEW_BIRDSEED.FORMULA;1 modified


$ SHOW ACL  NEW_BIRDSEED.FORMULA
Object type: file,  Object name: WORK1:[FINCH]NEW_BIRDSEED.FORMULA;1,
on 15-JAN-1985 13:15:21.35
    (IDENTIFIER=[BIRDS,WREN]+DIALUP,ACCESS=NONE)
    (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)
    (IDENTIFIER=PAYROLL_PROGRAMMERS,OPTIONS=NOPROPAGATE,ACCESS=NONE)


$ SET ACL/DELETE  BIRDSEED.FORMULA;*


$ DIRECTORY/ACL BIRDSEED.FORMULA

Directory WORK1:[FINCH]

BIRDSEED.FORMULA;3
    (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)
BIRDSEED.FORMULA;2
    (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)
BIRDSEED.FORMULA;1
    (IDENTIFIER=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)

Total of 3 files.


$ SET ACL /DELETE                                                    -
         /ACL=(ID=[BIRDS,*],OPTIONS=PROTECTED,ACCESS=READ+WRITE)  -
       BIRDSEED.FORMULA;*


$ DIRECTORY/ACL BIRDSEED

Directory WORK1:[FINCH]

BIRDSEED.FORMULA;3
BIRDSEED.FORMULA;2
BIRDSEED.FORMULA;1
```

EXAMPLE 6-15: EFFECTS OF DEFAULT OPTION ON IDENTIFIER ACE

```
$ SHOW DEFAULT
  WORK1:[FINCH.PATHWAYS]

$ DIR/ACL  [-]PATHWAYS.DIR

Directory WORK1:[FINCH]

PATHWAYS.DIR;1
     (IDENTIFIER=SONG_BIRDS,OPTIONS=DEFAULT,ACCESS=READ+WRITE+EXECUTE)
     (IDENTIFIER=PAYROLL_PROGRAMMERS+BATCH,
     OPTIONS=DEFAULT+NOPROPAGATE,ACCESS=READ+EXECUTE)
     (IDENTIFIER=SONG_BIRDS,ACCESS=READ+WRITE+EXECUTE)
     (IDENTIFIER=PAYROLL_PROGRAMMERS+BATCH,ACCESS=READ+EXECUTE)
     (IDENTIFIER=NETWORK,ACCESS=NONE)

Total of 1 file.

$ CREATE   EASTERN_MAPS.DATA
     this is map2 routes ^Z

$ COPY  EASTERN_MAPS.DATA   *.*

$ COPY  EASTERN_MAPS.DATA   NEW_EASTERN_MAPS.DATA

$ DIRECTORY/ACL

Directory WORK1:[FINCH.PATHWAYS]

EASTERN_MAPS.DATA;2
     (IDENTIFIER=SONG_BIRDS,ACCESS=READ+WRITE+EXECUTE)
EASTERN_MAPS.DATA;1
     (IDENTIFIER=SONG_BIRDS,ACCESS=READ+WRITE+EXECUTE)
     (IDENTIFIER=PAYROLL_PROGRAMMERS+BATCH,
     OPTIONS=NOPROPAGATE,ACCESS=READ+EXECUTE)
NEW_EASTERN_MAPS.DATA;1
     (IDENTIFIER=SONG_BIRDS,ACCESS=READ+WRITE+EXECUTE)
     (IDENTIFIER=PAYROLL_PROGRAMMERS+BATCH,
     OPTIONS=NOPROPAGATE,ACCESS=READ+EXECUTE)

Total of 3 files.

$ SET FILE/ACL/DEFAULT EASTERN_MAPS.DATA;2      ! copy default from parent directory.

$ DIR/ACL EASTERN_MAPS.DATA;2
EASTERN_MAPS.DATA;2
     (IDENTIFIER=SONG_BIRDS,ACCESS=READ+WRITE+EXECUTE)
     (IDENTIFIER=PAYROLL_PROGRAMMERS+BATCH,
     OPTIONS=NOPROPAGATE,ACCESS=READ+EXECUTE)

Total of 1 file.
```

EXAMPLE 6-16: SAMPLE IDENTIFIER ACL

```
(IDENTIFIER=SONG_BIRDS,
ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)

(IDENTIFIER=PAYROLL_PROGRAMMERS+LOCAL,
ACCESS=READ+WRITE+EXECUTE+DELETE)

(IDENTIFIER=DIALUP, ACCESS=NONE)

(IDENTIFIER=PAYROLL_PROGRAMMERS, ACCESS=READ+EXECUTE)

(IDENTIFIER=[300,*]+NETWORK, ACCESS=NONE)

(IDENTIFIER=PERCH, ACCESS=NONE)
```

NOTES ON EXAMPLE 6-16

- PROVIDES GREATEST ACCESS FIRST
- A SUBJECT WITH BOTH IDENTIFIERS SONG_BIRDS AND PAYROLL_PROGRAMMERS WILL GET ALL ACCESS (through SONG_BIRDS)
- MEMBERS OF UIC GROUP 300 WILL BE DENIED NETWORK ACCESS UNLESS THEY ARE ALSO THE HOLDER OF THE IDENTIFIER SONG_BIRDS OR PAYROLL_PROGRAMMERS
- USER PERCH WILL BE DENIED ALL ACCESS UNLESS SHE IS ALSO THE HOLDER OF THE IDENTIFIER SONG_BIRDS OR PAYROLL_PROGRAMMERS
- DIALUP USERS MAY ONLY GAIN ACCESS IF THEY ARE ALSO THE HOLDER OF THE IDENTIFIER SONG_BIRDS

EXAMPLE 6-17: DIRECTORY/FULL COMMAND

```
$ DIRECTORY/FULL  FEATHERS.DAT

Directory WORK1:[FINCH]

FEATHERS.DAT;1                File ID:  (12596,3,0)
Size:            2/3          Owner:    [BIRDS,FINCH]
Created:  15-JAN-1985 12:24   Revised:  15-JAN-1985 13:18 (3)
Expires:   <None specified>   Backup:    <No backup done>
File organization:  Sequential
File attributes:    Allocation: 3, Extend: 0, Global buffer count: 0,
                    No version limit
Record format:      Variable length, maximum 13 bytes
Record attributes:  Carriage return carriage control
File protection:    System:RWED, Owner:RWED, Group:RE, World:R
Access Cntrl List:  (IDENTIFIER=SONG_BIRDS,OPTIONS=PROTECTED,
                    ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)
                    (IDENTIFIER=DIALUP,OPTIONS=PROTECTED,ACCESS=NONE)
                    (IDENTIFIER=[300,*]+NETWORK,ACCESS=READ)

Total of 1 file, 2/3 blocks.
```

EXAMPLE 6-18: DIRECTORY/SECURITY COMMAND

```
$ DIRECTORY/SECURITY FEATHERS.DAT

Directory WORK1:[FINCH]

FEATHERS.DAT;1        [BIRDS,FINCH]          (RWED,RWED,RE,R)
    (IDENTIFIER=SONG_BIRDS,OPTIONS=PROTECTED,
    ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)
    (IDENTIFIER=DIALUP,OPTIONS=PROTECTED,ACCESS=NONE)
    (IDENTIFIER=[300,*]+NETWORK,ACCESS=READ)

Total of 1 file.
```

IDENTIFIER ACE FOR A DEVICE -

Devices may be protected by ACL's as well as by standard UIC-based protection. An example of such a use might be a special graphics printer that is used for the latest design plans for a special project (SHERLOCK). Only holders of the identifier PROJECT_SHERLOCK will be allowed to send any output to this printer. Example 6-19 shows the commands that will establish an ACL on the device so as lock anyone out from allocating, or sending output to the device.

EXAMPLE 6-19: SETTING DEVICE PROTECTION

```
$ SET PROTECTION=(S,O,G,W)/DEVICE  TTC3:


$ SET DEVICE TTC3: /ACL=(IDENTIFIER=PROJECT_SHERLOCK,ACCESS=READ+WRITE)


$ SHOW DEVICE/FULL TTC3:

Terminal TTC3:, device type VT200 Series, is online,
    record-oriented device, carriage control.

Error count                    0    Operations completed          7
Owner process                  ""   Owner UIC            [VMS,SYSTEM]
Owner process ID        00000000    Dev Prot              S:,O:,G:,W:
Reference count                0    Default buffer size            80

Device access control list:
    (IDENTIFIER=PROJECT_SHERLOCK,ACCESS=READ+WRITE)
```

DEFAULT PROTECTION ACE

A Default Protection ACE defines UIC-based protection that will be propagated to all files in a directory and its subdirectories.

(DEFAULT_PROTECTION [,OPTIONS=PROTECTED] ,PROTECTION MASK)

(DEFAULT_PROTECTION ,OPTIONS=PROTECTED ,S:RWE,O:RWED,G,W)

- CAN ONLY BE APPLIED TO DIRECTORY FILES
- DEFINES UIC PROTECTION CODE THAT IS APPLIED TO ALL NEW FILES IN THAT DIRECTORY AND ALL SUBDIRECTORIES
- USE OPTIONS=DEFAULT ON IDENTIFIER ACE TO PROPAGATE ACE'S TO FILES
- HAS NO EFFECT ON EXISTING FILES
- USE $ SET PROTECTION FILE-SPEC COMMAND TO CHANGE PROTECTION ON EXISTING FILES
- PROTECTION MASK IS LIKE ANY UIC-BASED CODE

EXAMPLE 6-20: USING DEFAULT PROTECTION ACE'S

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


$ DIRECTORY/ACL WORK1:[000000]FINCH.DIR

Directory WORK1:[000000]

FINCH.DIR;1
          (DEFAULT_PROTECTION,SYSTEM:RWE,OWNER:RWE,GROUP:,WORLD:)

Total of 1 file.


$ DIRECTORY/PROTECTION/ACL  FLYWAYS.MAP
Directory WORK1:[FINCH]

FLYWAYS.MAP;1          (RWED,RWED,RE,R)
          (IDENTIFIER=BIRDS,ACCESS=READ+WRITE)

Total of 1 files.


$ COPY FLYWAYS.MAP *.*

$ COPY FLYWAYS.MAP  NEW_ROUTES.DAT


$ CREATE NEW_ROUTES2.DAT
    this is new_routes2 data ^Z


$ DIR/PROTECTION/ACL  FLYWAYS.MAP, NEW_ROUTES.DAT, NEW_ROUTES2.DAT

Directory WORK1:[FINCH]

FLYWAYS.MAP;2          (RWED,RWED,RE,R)
          (IDENTIFIER=BIRDS,ACCESS=READ+WRITE)
FLYWAYS.MAP;1          (RWED,RWED,RE,R)
          (IDENTIFIER=BIRDS,ACCESS=READ+WRITE)
NEW_ROUTES.DAT;1       (RWE,RWE,,)
NEW_ROUTES2.DAT;1      (RWE,RWE,,)

Total of 4 files.
```

Protection propagate to new versions of itself.

New file protection comes from Directory.

SECURITY ALARM ACE

A **SECURITY ALARM ACE** provides an alarm to the SECURITY operator(s) when an object is accessed.

FORMAT : (ALARM_JOURNAL=SECURITY [,OPTIONS] [,ACCESS])

- SENDS ALARM WHEN AN OBJECT IS ACCESSED IN THE PRESCRIBED MANNER
- OPTIONS ARE THE SAME AS FOR AN IDENTIFIER ACE
  SEE TABLE 6-15
- ACCESS CODE SPECIFIES WHAT ACTIONS WILL SEND ALARM
  MEANINGS ARE THE SAME AS FOR AN IDENTIFIER ACE
  SEE TABLE 6-16
  - READ, WRITE, EXECUTE, DELETE, CONTROL
  - SUCCESS, FAILURE

NOTE

SUCCESS, FAILURE (OR BOTH) MUST BE INCLUDED ON EVERY SECURITY ALARM ACE

- ALARM GOES TO SECURITY OPERATORS AND OPERATOR'S LOG FILE
- SECURITY AUDITING MUST ALSO BE ENABLED

  DISCUSSED IN CHAPTER 8

EXAMPLE 6-21: SAMPLE SECURITY ALARM ACE

```
(ALARM_JOURNAL=SECURITY,OPTIONS=NOPROPAGATE,ACCESS=EXECUTE+SUCCESS)

(ALARM_JOURNAL=SECURITY,OPTIONS=PROTECTED,ACCESS=READ+WRITE+FAILURE)

(ALARM_JOURNAL=SECURITY,ACCESS=CONTROL+SUCCESS+FAILURE)
```

change ACL, protection, backup date

NOTES ON EXAMPLE 6-21

- The first example will send an alarm at every successful execution of the file or command procedure.
- It will not be propagated to new versions of the file.
- The second example will send an alarm when unsuccessful reads or writes are attempted to the file or directory.
- The third ACE will send an alarm when any control operation is performed on the file or directory.

USING THE ACL EDITOR

TABLE 6-17: ACL EDITOR CHARACTERISTICS

- USED TO CREATE OR MODIFY AN ACL
- CAN WORK WITH ANY TYPE OF ACE
- WORKS ON EXISTING FILES-11 STRUCTURE LEVEL 2 FILES, DIRECTORIES, OR DEVICES (queues in V5)
  $ EDIT/ACL/OBJECT=([FILE][DEVICE])

logical name table
sys global sect
grp global sect

- SCREEN ORIENTED EDITOR (VT200, VT100, VT52)
- KEYPAD FUNCTIONS SIMILAR TO EDT
- ONLINE HELP
- IMMEDIATE ACL SYNTAX CHECKING
- CTRL Z EXITS AND WRITES ACL
- DEFINABLE KEYS
- STARTUP FILE (ACLEDIT$INIT)
- MUST HAVE CONTROL ACCESS TO THE OBJECT
- PROMPTING AND NONPROMPTING MODE

  $ EDIT/ACL/MODE=[NO]PROMPTING

- EDIT JOURNAL AND RECOVERY CAPABILITY

  $ EDIT/ACL/[NO]JOURNAL[=FILE-SPEC]

  $ EDIT/ACL/KEEP=([JOURNAL][,RECOVER])

  $ EDIT/ACL/RECOVER[=FILE-SPEC]

TABLE 6-18: GETTING STARTED WITH THE ACL EDITOR

1. If not already at the "(IDENTIFIER=" prompt, use the INSERT key to indicate you wish to add an ACE (you will then get the prompt).

2. Enter the UIC (and any identifiers such as BATCH, REMOTE, etc) that will identify to whom the ACE applies (or use the ITEM key to select a different ACE type).

3. Use the FIELD key to advance to the next portion of the ACE.

4. Use the ITEM key to select the desired option or access.

5. Repeat 3 and 4 until all options and access modes have been selected.

6. Use the ENTER key to indicate the you are satisfied with the ACE, and that it should be added to the file's ACL.

7. If the ACE is not entered because of a syntax error, use the arrow keys to position to the portion of the ACE causing the error. Correct the error, and again press the ENTER key to add the ACE.

8. Repeat 2 through 7 until all ACEs have been added. Then type Control-Z to finish the editing session, and return to DCL.

6.8 DEFAULT PROTECTION AND OWNERSHIP

Ownership of an object is critical to the outcome of any protection check. Obtaining the ownership rights to an object will allow complete access to the object, since ownership implies control, and control allows the subject to change the file protection and/or ACL.

OBJECT OWNERSHIP

Ownership of an object may come in several forms. The subject may be the recorded owner, or may have obtained the OWNERSHIP PRIVILEGE to that object. Table 6-19 lists the conditions that are required to have the ownership privilege. Possession of any one of the conditions is sufficient to receive the ownership privilege.

TABLE 6-19: CONDITIONS NECESSARY TO RECEIVE OWNERSHIP PRIVILEGE

- HAVE A PROCESS UIC THAT MATCHES EXACTLY THE UIC OF THE OBJECT
- QUALIFY AS A MEMBER OF THE SYSTEM CATEGORY OF USERS
- HOLD SYSPRV OR BYPASS PRIVILEGE
- HOLD THE GRPPRV PRIVILEGE AND ALSO HAVE A UIC IN THE SAME GROUP AS THE OWNER OF THE OBJECT
- POSSESS OWNER IDENTIFIERS IN THE RIGHTS LIST WITH THE RESOURCE ATTRIBUTE THAT CORRESPOND TO THE OWNER OF THE OBJECT

TABLE 6-20: RESOURCE ATTRIBUTE FOR IDENTIFIERS

- GENERAL IDENTIFIERS MAY HAVE THE RESOURCE ATTRIBUTE
- UAF> ADD/IDENTIFIER IDENTIFIER/ATTRIBUTE=RESOURCE
- UAF> GRANT/ID IDENTIFIER USERNAME/ATTRIBUTE=RESOURCE
- ALLOWS OBJECTS TO BE OWNED BY IDENTIFIERS RATHER THAN INDIVIDUAL USERS
- DISK SPACE CAN BE CHARGED TO THE IDENTIFIER
- DISK QUOTA CAN BE DONE BY DEPARTMENT OR PROJECT
- SOME HOLDERS OF THE IDENTIFIER MAY NOT HAVE THE RESOURCE ATTRIBUTE
- GIVEN TO PROJECT MANAGERS, GROUP LEADERS, SENIOR MEMBERS

TABLE 6-21: DEFAULT FILE AND DIRECTORY OWNERSHIP

- THE OWNER OF A PREVIOUSLY EXISTING VERSION OF THE FILE
- THE OWNER OF THE PARENT DIRECTORY
- THE UIC OF THE FILE CREATOR
- USES FIRST MATCHING CONDITION
- USERS WITH THE OWNERSHIP PRIVILEGE CAN CHANGE THE DEFAULT OWNERSHIP RULES
  - $ CREATE/OWNER=OWNER
  - $ SET FILE/OWNER=OWNER FILE-SPEC
  - $ SET DIRECTORY/OWNER=OWNER [DIRECTORY-NAME]
- SEE OWNERSHIP WITH $ DIRECTORY/OWNER OR $ DIRECTORY/SECURITY

TABLE 6-22: DEFAULT DIRECTORY PROTECTION

- UIC PROTECTION CODE OF NEWLY CREATED DIRECTORY FILE DEFAULTS TO PROTECTION OF NEXT HIGHER DIRECTORY
- MAY CHANGE WITH $ CREATE/DIRECTORY/PROTECTION=(CODE)
- SUBDIRECTORY INHERITS ACL's OF ITS PARENT NOT SET OPTIONS=NOPROPAGATE

TABLE 6-23: DEFAULT FILE PROTECTION

- NEW VERSIONS OF FILES WILL INHERIT UIC PROTECTION FROM
  1. EXISTING VERSIONS (DIRECTORY/PROTECTION)
  2. DEFAULT_PROTECTION ACE ON DIRECTORY
  3. PROCESS DEFAULT PROTECTION

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

     $ SET PROTECTION=(CODE)/DEFAULT

     SYSGEN PARAMETER RMS_FILEPROT

- ACL's ARE PROPAGATED FROM OLD FILES TO NEW VERSIONS
- ACL's ARE CREATED FROM OPTIONS=DEFAULT ACE ON DIRECTORY FILES

EXAMPLE 6-22: USING A RESOURCE ATTRIBUTE IDENTIFIER

```
UAF> ADD/ID/ATTRIBUTE=RESOURCE SEED_FORMULAS

UAF> GRANT/ID/ATTRIBUTE=RESOURCE  SEED_FORMULAS FINCH

UAF> GRANT/ID/ATTRIBUTE=NORESOURCE  SEED_FORMULAS WREN

UAF> GRANT/ID/ATTRIBUTE=NORESOURCE  SEED_FORMULAS ROBIN

UAF> SHOW/ID/FULL SEED_FORMULAS

  Name       -                    Value           Attributes
  SEED_FORMULAS                   %X8001000E      RESOURCE
    Holder                          Attributes
    FINCH                           RESOURCE
    ROBIN                           NORESOURCE
    WREN                            NORESOURCE


$ RUN SYS$SYSTEM:DISKQUOTA
DISKQ>USE WORK1:
DISKQ>ADD  SEED_FORMULAS  /PERMQUOTA=10000/OVERDRAFT=500


$ CREATE/DIRECTORY/OWNER=SEED_FORMULAS  WORK1:[SEED_FORMULAS]


$ SET DIRECTORY  WORK1:[SEED_FORMULAS]                                  -
      /ACL=((ID=SEED_FORMULAS,ACCESS=READ+WRITE+EXECUTE),               -
      (ID=SEED_FORMULAS,OPTIONS=DEFAULT,ACCESS=READ+WRITE+EXECUTE))



$ CREATE  WORK1:[SEED_FORMULAS]SUNFLOWER.MIX1
      formula1^Z
```

FILE AND OBJECT PROTECTION METHOD

```
$ DIR/SECURITY   WORK1:[000000]SEED_FORMULAS.DIR

Directory WORK1:[000000]

SEED_FORMULAS.DIR;1      SEED_FORMULAS          (RWE,RWE,RE,E)
    (IDENTIFIER=SEED_FORMULAS,ACCESS=READ+WRITE+EXECUTE)
    (IDENTIFIER=SEED_FORMULAS,OPTIONS=DEFAULT,
    ACCESS=READ+WRITE+EXECUTE)

Total of 1 file.


$ DIR/SECURITY  WORK1:[SEED_FORMULAS]

Directory WORK1:[SEED_FORMULAS]

SUNFLOWER.MIX1;1         SEED_FORMULAS          (RWED,RWED,RE,)
    (IDENTIFIER=SEED_FORMULAS,ACCESS=READ+WRITE+EXECUTE)

Total of 1 file.


$ DIR/SECURITY FLYWAYS.DATA

Directory WORK1:[FINCH]

FLYWAYS.DATA;2           [BIRDS,FINCH]          (RWED,RWED,RE,)

Total of 1 file.


$ COPY FLYWAYS.DATA  WORK1:[SEED_FORMULAS]


$ DIR/SECURITY  WORK1:[SEED_FORMULAS]

Directory WORK1:[SEED_FORMULAS]

FLYWAYS.DATA;2           SEED_FORMULAS          (RWED,RWED,RE,)
    (IDENTIFIER=SEED_FORMULAS,ACCESS=READ+WRITE+EXECUTE)

SUNFLOWER.MIX1;1         SEED_FORMULAS          (RWED,RWED,RE,)
    (IDENTIFIER=SEED_FORMULAS,ACCESS=READ+WRITE+EXECUTE)

Total of 2 files.
```

NOTES ON EXAMPLE 6-22

1. FINCH HOLDS THE IDENTIFIER SEED_FORMULAS WITH THE RESOURCE ATTRIBUTE

2. WREN AND ROBIN HOLD THE SAME IDENTIFIER WITH THE NORESOURCE ATTRIBUTE

3. FILES CREATED BY FINCH IN THE DIRECTORY [SEED_FORMULAS] WILL BE OWNED BY SEED_FORMULAS

4. FILES CREATED BY FINCH IN HER OWN DIRECTORY WILL BE OWNED BY FINCH

5. ACL's ARE NECESSARY FOR THE DIRECTORY AND FILES SINCE NO PROCESS WILL HAVE SEED_FORMULAS AS THEIR PROCESS UIC

6. THE DEFAULT ACE ALLOWS ALL FILES CREATED IN THE DIRECTORY TO HAVE THE SAME ACE

7. DELETE AND CONTROL ACCESS ARE DENIED NON-OWNERS

8. FILES CREATED BY WREN OR ROBIN WILL BE OWNED BY THEM

6.9 MANAGING FILE PROTECTION

A system of file protection management will include the use of UIC-based protection as well as ACL's. Try to use UIC based protection whenever possible. Only go to ACL's when the UIC groups do not provide a sufficient amount of control.

Good file protection will help to avoid WORM HOLES and keep out WORMS. A worm is a user (insider or intruder) who will wind through the system picking up privileges as he/she goes. Worms will access another command procedure or executable image and modify the file so that it carries a copy of the worm to the next executor of the file. The worm attempts to propagate itself from one user to another through a worm hole until he/she arrives at a privileged account. Worm holes are gaps in file protection that will allow this to happen. ACL's provide a great opportunity for such holes. Favorite targets are login command procedures. These need to be protected as do the directories in which they reside. This is especially true of those for privileged accounts.

Table 6-24 lists some general considerations for good design of ACL's. Table 6-25 lists some considerations and guidelines for overall management of the file system. Good file protection design and management will lead to a more secure system.

TABLE 6-24: ACL DESIGN CONSIDERATIONS

- THE ORDER IS ESSENTIAL
  - USERS MAY HOLD MULTIPLE IDENTIFIERS
  - USES FIRST MATCH
  - PLACE ACE's THAT DENY ACCESS TO SPECIFIC USERS AT THE TOP
  - PLACE ACE's THAT DENY ACCESS TO LARGE GROUPS NEAR THE BOTTOM
  - PLACE ACE's THAT GRANT THE WIDEST ACCESS IMMEDIATELY AFTER THE MOST RESTRICTIVE ONES
- DO NOT ASSUME THAT ACCESS=NONE WILL ABSOLUTELY PROHIBIT THE HOLDERS OF THE IDENTIFIER TO ACCESS THE OBJECT
  - MAY BE SYSTEM OR OWNER OR HOLD A PRIVILEGE
- DO NOT PLACE ACL's ON EVERYTHING
- CONSIDER THE PERFORMANCE - ACL's ARE USED IF THEY EXIST
- A SMALL NUMBER OF DEFAULT ACE's CAN PROPAGATE TO MANY FILES
- KEEP ACL's AS SHORT AS POSSIBLE
- USE GENERAL IDENTIFIERS TO CREATE LOGICAL AND PRACTICAL GROUPS
- PLAN BEFORE IMPLEMENTING
- UPDATE ACE's WHEN USERS LEAVE

TABLE 6-25: GENERAL FILE PROTECTION CONSIDERATIONS

- CONSIDER THREE LEVELS OF PROTECTION

  DEVICE >>> DIRECTORY >>> FILE

- WRITE ACCESS TO A DIRECTORY ALLOWS MANIPULATION OF PROTECTED FILES

  $ RENAME [FINCH]FLYWAYS.DATA [MARAUDER.HIDDEN]JUNK.XYZ

- TO PROTECT A FILE COMPLETELY YOU MUST PROTECT THE DIRECTORY AND THE FILE

- CONSIDER THE USE OF EXECUTE ACCESS TO DIRECTORY FILES

- DISCOURAGE USERS FROM HAVING WORLD READABLE DIRECTORIES AND FILES

- PERIODICALLY SEARCH THE DISK FOR SUCH FILES AND DIRECTORIES

- PAY SPECIAL ATTENTION TO COMMAND PROCEDURES AND IMAGE FILES ACCESSED BY PRIVILEGED USER

- DO NOT ALLOW WORLD:WRITE TO SYSTEM DIRECTORIES

- DO NOT RUN USER PROGRAMS OR COMMAND PROCEDURES FROM YOUR ACCOUNT

- LET USERS KNOW ~~WHO HOLDS IDENTIFIERS~~ WHAT IDENTIFIERS THEY HOLD.

FILE AND OBJECT PROTECTION METHOD

- EDUCATE THE USERS TO

  - USE NONOBVIOUS AND UNINTERESTING NAMES FOR FILES AND DIRECTORIES

  - BE CONSISTENT IN NAMING CONVENTIONS

  - KEEP DIRECTORIES CLEAN - PURGE/DELETE UNNECESSARY FILES

  - USE $ DIRECTORY/SECURITY COMMAND TO MONITOR FILE PROTECTION, OWNERSHIP, AND ACL's

  - USE $ DIRECTORY/FULL TO CHECK REVISION DATES

  - PROTECT LOGIN COMMAND PROCEDURES

  - CORRECTLY CREATE ACL'S BASED ON LOGICAL GROUPINGS

CHAPTER 7
SECURITY IMPLEMENTATION METHODS

LIST OF TOPICS

- Security Related System Services
  - Overview of Security System Services
  - Using the Security Related Services
  - $CHKPRO SYSTEM SERVICE
  - KGB
- Granting User Privileges
  - Classes of Privileges
  - Secure Management of Privileges
  - Management of Privileged Accounts
  - Installing Privileged Images
- Captive Command Procedures
  - Security Problems with Command Procedures
  - GUEST and MAIL Accounts
  - AUTOMATIC LOGIN FACILITY
- Removing DCL Commands

7.1 SECURITY RELATED SYSTEM SERVICES

OVERVIEW OF SECURITY RELATED SYSTEM SERVICES

The VAX/VMS security system services provide various mechanisms that you can use to enhance the security of VAX/VMS systems. These services are called by the DCL and UAF commands to manipulate the rights database and are used by the file system to make all protection checks. The reference monitor is invoked by a series of calls to these system services. They may also be called by application programs to perform security checks.

TABLE 7-1: SECURITY FUNCTIONS AVAILABLE THROUGH SYSTEM SERVICES

- CREATE AND MAINTAIN A RIGHTS DATABASE
- CREATE AND TRANSLATE ACCESS CONTROL LIST ENTRIES
- MODIFY A PROCESS RIGHTS LIST
- CHECK ACCESS PROTECTION
- PROVIDE A SECURITY ERASE PATTERN FOR DISKS
- CONTROL MAGNETIC TAPE ACCESS

TABLE 7-2: SECURITY SERVICES OVERVIEW

SERVICE NAME	FUNCTION(S)
$ADD_HOLDER	ADD HOLDER RECORD TO RIGHTS DATABASE
$ADD_IDENT	ADD IDENTIFIER TO RIGHTS DATABASE
$ASCTOID	TRANSLATE IDENTIFIER NAME TO BINARY VALUE
$CHANGE_ACL	CREATE OR MODIFY AN ACL
$CHKPRO	INVOKE SYSTEM ACCESS PROTECTION CHECK
$CREATE_RDB	INITIALIZE A RIGHTS DATABASE
$ERAPAT	GENERATE A SECURITY ERASE PATTERN
$FIND_HELD	RETURN IDENTIFIER(S) HELD BY A HOLDER IN RIGHTS DATABASE
$FIND_HOLDER	RETURN HOLDER(S) OF AN IDENTIFIER IN RIGHTS DATABASE
$FINISH_RDB	DEALLOCATE RECORD STREAM AND CLEAR CONTEXT VALUE WHEN SEARCHING THE RIGHTS DATABASE
$FORMAT_ACL	FORMAT ACE INTO A TEXT STRING
$GRANTID	ADD IDENTIFIER TO RIGHTS LIST
$IDTOASC	TRANSLATE IDENTIFIER VALUE TO ITS IDENTIFIER NAME
$MOD_HOLDER	MODIFIY HOLDER RECORD IN RIGHTS DATABASE
$MOD_IDENT	MODIFIY IDENTIFIER RECORD IN RIGHTS DATABASE
$MTACCESS	CONTROL MAGNETIC TAPE ACCESS
$PARSE_ACL	CONVERT TEXT ACE INTO BINARY FORMAT
$REM_HOLDER	DELETE HOLDER RECORD FROM IDENTIFIER'S LIST OF HOLDERS IN RIGHTS DATABASE
$REM_IDENT	DELETE IDENTIFIER AND ALL HOLDERS OF THAT IDENTIFIER FROM RIGHTS DATABASE
$REVOKID	REMOVE IDENTIFIER FROM RIGHTS LIST

USING SECURITY RELATED SERVICES

EXAMPLE 7-1: USING $IDTOASC TO LIST ALL IDENTIFIERS

```
        Program ID_LIST
*
* Produce a list of all the identifiers
*
        integer SYS$IDTOASC
        external SS$_NORMAL, SS$_NOSUCHID

        character*31 NAME
        integer IDENTIFIER, ATTRIBUTES

        integer ID/-1/, LENGTH, CONTEXT/0/
        integer NAME_DSC(2)/31, 0/

        integer STATUS

*
* Initialization
*
        NAME_DSC(2) = %loc(NAME)
        STATUS = %loc(SS$_NORMAL)

*
* Scan through the entire RDB ...
*
        do while (STATUS .and. (STATUS .ne. %loc(SS$_NOSUCHID)))

         STATUS = SYS$IDTOASC(%val(ID), LENGTH, NAME_DSC,
     +   IDENTIFIER, ATTRIBITUES, CONTEXT)

         if (STATUS .and. (STATUS .ne. %loc(SS$_NOSUCHID))) then
            NAME(LENGTH+1:LENGTH+1) = ','
            print 1, NAME, IDENTIFIER, ATTRIBUTES
1           format(1X,'Name: ',A31,' Id: ',Z8,', Attributes: ',Z8)
         end if

        end do

*
* Do we need to finish the RDB ???
*
       if (STATUS .ne. %loc(SS$_NOSUCHID)) then
          call SYS$FINISH_RDB(CONTEXT)
       end if

       end
```

Terminal Session :

```
$ RUN ID_LIST

Name: BIRDS,                Id:   E8FFFF, Attributes:          0
Name: DECNET,               Id:   FF00FF, Attributes:          0
Name: DIALUP,               Id: 80000002, Attributes:          0
Name: DOE,                  Id:   D20006, Attributes:          0
Name: FIELD,                Id:    10008, Attributes:          0
Name: FINCH,                Id:   E8001D, Attributes:          0
Name: FLYERS,               Id: 80010008, Attributes:          0
Name: INTERACTIVE,          Id: 80000003, Attributes:          0
Name: INVENTORY,            Id:  100000E, Attributes:          0
Name: LOCAL,                Id: 80000004, Attributes:          0
Name: OPERATOR,             Id:   C7FFFF, Attributes:          0
Name: ROBIN,                Id:   680008, Attributes:          0
Name: SEED_FORMULAS,        Id: 8001000B, Attributes:          0
Name: SONG_BIRDS,           Id: 80010006, Attributes:          0
Name: VMS,                  Id:    1FFFF, Attributes:          0
Name: WARBLER,              Id:   68000D, Attributes:          0
```

NOTES ON EXAMPLE 7-1

1. IF $IDTOASC ARGUMENTS ID =-1 AND CONTEXT =0 THEN THE SERVICE SCANS THE ENTIRE RIGHTS DATABASE

2. SCANS IN ALPHABETICAL ORDER BY ID NAME

3. PROCESS UNTIL SS$_NOSUCHID STATUS IS RETURNED

4. ALSO CONVERTS ID TO ASCII FORMAT

5. PRINTS ID NAME, ID VALUE, AND ATTRIBUTES

6. CAN SCAN RIGHTS DATABASE WITH SIMILAR ARGUMENT VALUES TO $FIND_HOLD AND $FIND_HELD

EXAMPLE 7-2: USING $MOD_HOLDER

```
        Program MOD_HOLDER
*
* Modify the RESOURCE attribute of all the holders of identifiers
* to reflect the current attribute setting of the identifiers
*
        external SS$_NOSUCHID
        parameter KGB$M_RESOURCE = 1
        integer SYS$IDTOASC, SYS$FIND_HELD, SYS$MOD_HOLDER

*
* Store information about the holder here
*
        integer HOLDER(2)/2*0/
        equivalence (HOLDER(1), HOLDER_ID)
        integer HOLDER_NAME(2)/31, 0/
        integer HOLDER_ID, HOLDER_CTX/0/
        character*31 HOLDER_STRING

*
* Store attributes here
*
        integer OLD_ATTR, NEW_ATTR, ID_ATTR, CONTEXT

*
* Store information about the identifier here
*
        integer IDENTIFIER, ID_NAME(2)/31, 0/
        character*31 ID_STRING

        integer STATUS

*
* Initialize the descriptors
*
        HOLDER_NAME(2) = %loc(HOLDER_STRING)
        ID_NAME(2) = %loc(ID_STRING)

*
* Scan through all the identifiers
*
        do while
     +  (SYS$IDTOASC(%val(-1),, HOLDER_NAME, HOLDER_ID,, HOLDER_CTX)
     +  .ne. %loc(SS$_NOSUCHID))
```

```
*
* Test all the identifiers held by this identifier (our HOLDER)
*
        if (HOLDER_ID .le. 0) go to 2

        CONTEXT = 0

        do while
     +  (SYS$FIND_HELD(HOLDER, IDENTIFIER, OLD_ATTR, CONTEXT)
     +  .ne. %loc(SS$_NOSUCHID))

*
* Get name and attributes of held identifier
*
        STATUS = SYS$IDTOASC(%val(IDENTIFIER),, ID_NAME,, ID_ATTR,)

*
* Modify the holder record to reflect the state of the identifier
*
        if ((ID_ATTR .and. KGB$M_RESOURCE) .ne. 0) then
           STATUS = SYS$MOD_HOLDER
     +     (%val(IDENTIFIER), HOLDER, %val(KGB$M_RESOURCE),)

           NEW_ATTR = OLD_ATTR .or. KGB$M_RESOURCE

        else
           STATUS = SYS$MOD_HOLDER
     +     (%val(IDENTIFIER), HOLDER,, %val(KGB$M_RESOURCE))

           NEW_ATTR = OLD_ATTR .and. (.not. KGB$M_RESOURCE)
        end if

*
* Were we successful ?
*
        if (.not. STATUS) then
           NEW_ATTR = OLD_ATTR
           call LIB$SIGNAL(%val(STATUS))
        end if

*
* Report it all
*
        print 1, HOLDER_STRING, ID_STRING,
     +  OLD_ATTR, ID_ATTR, NEW_ATTR
1       format(1X, 'Holder: ', A31, ' Id: ', A31,
     +  ' Old: ', Z8, ' Id: ', Z8, ' New: ', Z8)

        end do
2       continue

        end do
        end
```

Terminal Session :

```
$ RUN MOD_HOLDER

Holder: BIRDS  Id: SONG_BIRDS      Old: 1   Id: 1   New: 1
Holder: FINCH  Id: SONG_BIRDS      Old: 0   Id: 1   New: 1
Holder: FINCH  Id: SEED_FORMULAS   Old: 1   Id: 1   New: 1
Holder: ROBIN  Id: SEED_FORMULAS   Old: 0   Id: 1   New: 1
Holder: ROBIN  Id: PAYROLL_PROGS   Old: 0   Id: 0   New: 0
Holder: SCOTT  Id: SONG_BIRDS      Old: 1   Id: 1   New: 1
Holder: SCOTT  Id: SEED_FORMULAS   Old: 1   Id: 1   New: 1
```

NOTES ON EXAMPLE 7-2

1. SCANS ALL IDENTIFIERS USING $IDTOASC

2. FINDS ALL IDENTIFIERS HELD BY HOLDER USING $FIND_HELD

3. PASSES ID AND HOLDER ARGUMENT TO $MOD_HOLDER

4. USES $MOD_HOLDER AND THE SET_ATTRIB PARAMETER TO ADD THE RESOURCE ATTRIBUTE TO HOLDER RECORD IF ITS ON IDENTIFER RECORD

5. SPECIFIED AS KGB$MRESOURCE

6. WILL CREATE A NEW HOLDER RECORD IN THE RIGHTS DATABASE

$CHKPRO SYSTEM SERVICE

The $CHKPRO system service invokes the access protection check that is used by the system. It is called by all parts of VMS making access protection checks. Layered products, utilities and application programs may call it as well.

The service does not grant or deny access, rather it performs the protection check on behalf of a layered product, application program, or other similar subsystem which in turn must specifically grant or deny access.

- INVOKES THE SYSTEM'S ACCESS PROTECTION CHECK
- ALLOWS LAYERED PRODUCTS AND OTHER SUBSYSTEMS TO BUILD PROTECTED LOCAL STRUCTURES WHOSE PROTECTION IS CONSISTENT WITH THE PROTECTION FACILITIES OF VMS
- ALLOWS A PRIVILEGED SUBSYSTEM (USER) TO PERFORM PROTECTION CHECKS ON BEHALF OF SOME REQUESTER
- FORMAT : SYS$CHKPRO ITMLST
- THE ARGUMENTS SPECIFY :
  - THE PROTECTION OF THE OBJECT BEING ACCESSED
  - THE RIGHTS AND PRIVILEGES OF THE ACCESSOR
  - THE TYPE OF ACCESS DESIRED

- PASS THE INPUT AND OUTPUT INFORMATION TO WITH AN ITEM LIST
- COMPARES THE ITEM LIST OF THE RIGHTS AND PRIVILEGES OF THE ACCESSOR TO A LIST OF THE PROTECTION ATTRIBUTES OF THE OBJECT
- RETURNS SS$_NORMAL IF THE ACCESSOR CAN ACCESS THE OBJECT
- RETURNS SS$_NOPRIV IF THE ACCESSOR CANNOT ACCESS THE OBJECT
- THE CALLER MUST GRANT OR DENY ACCESS BASED ON THE OUTPUT
- RETURNS THE RIGHTS OR PRIVILEGES THAT ALLOWED ACCESS TO THE OBJECT
- RETURNS THE NAMES OF SECURITY ALARMS RAISED BY THE ACCESS ATTEMPT
- ALARMS MAY BE REQUESTED WHETHER THE PROTECTION CHECK SUCCEEDS OR FAILS
- ITMLST ARGUMENT IS THE ADDRESS OF AN ITEM LIST OF DESCRIPTORS USED TO SPECIFY THE PROTECTION ATTRIBUTES OF THE OBJECT AND THE RIGHTS AND PRIVILEGES OF THE ACCESSOR
- FOUR ELEMENTS FOR EACH ITEM LIST CODE
- END THE LIST WITH CHP$_END (ZERO)
- ITEM CODES DEFINED IN THE SYSTEM MACRO LIBRARY ($CHPDEF)

TABLE 7-3: $CHKPRO INPUT ITEMS = ACCESSOR'S RIGHTS AND PRIVILEGES

ITEM IDENTIFIER	DESCRIPTION
CHP$_ACCESS	BIT MASK REPRESENTING THE TYPE OF ACCESS DESIRED. ($ARMDEF)
CHP$_ACMODE	ACCESSOR'S PROCESSOR ACCESS MODE.
CHP$_ADDRIGHTS	ADDITIONAL RIGHTS LIST SEGMENT TO BE APPENDED TO EXISTING RIGHTS LIST.
CHP$_FLAGS	ACCESSOR'S ACCESS TO THE OBJECT.
CHP$V_READ	ACCESSOR IS MAKING A READ ACCESS
CHP$V_WRITE	ACCESSOR IS MAKING A WRITE ACCESS
CHP$V_USEREADALL	ACCESSOR IS ELIGIBLE FOR READALL PRIVILEGE
CHP$_PRIV	ACCESSOR'S PRIVILEGE MASK.
CHP$_RIGHTS	ACCESSOR'S RIGHTS LIST

TABLE 7-4: $CHKPRO INPUT ITEMS = OBJECT'S PROTECTION ATTRIBUTES

ITEM IDENTIFIER	DESCRIPTION
CHP$_ACL	OBJECT'S ACCESS CONTROL LIST
CHP$_MODE	OBJECT'S OWNER ACCESS MODE
CHP$_MODES	OBJECT'S ACCESS MODE PROTECTION
CHP$_OWNER	OBJECT'S OWNER UIC
CHP$_PROT	OBJECT'S "SOGW" PROTECTION MASK

TABLE 7-5: $CHKPRO OUTPUT ITEMS = INFORMATION RETURNED

ITEM IDENTIFIER	DESCRIPTION
CHP$_ALARMNAME	CHARACTER STRING CONTAINING THE ALARM RECORD
CHP$_MATCHEDACE	CONTAINS THE ACE IN OBJECT'S ACL THAT ALLOWED THE ACCESSOR TO ACCESS THE OBJECT
CHP$_PRIVUSED	MASK OF FLAGS REPRESENTING PRIVILEGES USED TO GAIN THE REQUESTED ACCESS
CHP$_SYSPRV	USED SYSPRV TO GAIN THE REQUESTED ACCESS
CHP$_GRPPRV	USED GRPPRV TO GAIN THE REQUESTED ACCESS
CHP$_BYPASS	USED BYPASS TO GAIN THE REQUESTED ACCESS
CHP$_READALL	USED READALL TO GAIN THE REQUESTED ACCESS

KGB

The Key Grants Block (KGB) is a data structure in the rights database that associates identifier codes with names, and lists the holders of all identifiers in the system.

TABLE Z-6: KEY GRANTS BLOCK DEFINITIONS

KGB$M_RESOURCE	
KGB$K_HOLD_RECORD	End of holder record
KGB$K_IDENT_RECORD	End of identifier record
KGB$K_LEVEL1	Version 1 structure level
KGB$K_MAINT_RECORD	End of maintenance record
KGB$S_KGBDEF	
KGB$L_IDENTIFIER	Binary identifier code
KGB$M_RESOURCE	Resource use allowed
KGB$S_HOLDER	Holder identifier
KGB$S_NAME	Identifier name
KGB$W_LEVEL	File structure level
KGB$S_SYS_ID	System identifier
KGB$L_NEXT_ID	Next available identifier
KGB$K_BATCH_ID	Batch ID value
KGB$K_DIALUP_ID	Dialup ID value
KGB$K_INTERACTIVE_ID	Interactive ID value
KGB$K_LOCAL_ID	Local ID value
KGB$K_NETWORK_ID	Network ID value
KGB$K_REMOTE_ID	Remote ID value

7.2 GRANTING USER PRIVILEGES

- PRIVILEGES RESTRICT THE PERFORMANCE OF CERTAIN SYSTEM ACTIVITIES TO USERS WHO HOLD THOSE PRIVILEGES

- THESE RESTRICTIONS PROTECT THE INTEGRITY OF THE OPERATING SYSTEM AND THEREFORE THE SECURITY OF THE SERVICES PROVIDED TO THE USERS OF THE SYSTEM

- GRANT PRIVILEGED BASED ON TWO FACTORS:

  1. WHETHER THE USER HAS THE SKILL AND EXPERIENCE TO USE THE PRIVILEGE WITHOUT DISRUPTING THE SYSTEM

  2. WHETHER THE USER HAS A LEGITIMATE NEED FOR THE PRIVILEGE

- UAF RECORD CONTAINS DEFAULT AND AUTHORIZED PRIVILEGE VECTORS (64-BITS)

  ```
  UAF> MODIFY FINCH/PRIV=(NETMBX,TMPMBX,GRPNAM)
  UAF> MODIFY FINCH/DEFPRIV=(NETMBX,TMPMBX)
  ```

- BECOME THE PROCESS' CURRENT AND AUTHORIZED PRIVILEGES

- PROCESS PRIVILEGES ARE STORED IN THE MULTIPLE LOCATIONS IN SEVERAL PROCESS DATA STRUCTURES

- A USER CAN MODIFY HIS/HER CURRENT PRIVILEGES

  ```
  $ SET PROCESS/PRIVILEGES=(PRIV-LIST)
  ```

- CAN ENABLE ANY PRIVILEGES WHICH ARE AUTHORIZED FOR THAT USER

- SETPRV PRIVILEGE ALLOWS A USER TO ENABLE ANY PRIVILEGE

CLASSES OF PRIVILEGES

TABLE 7-7: CATEGORIES OF PRIVILEGES

- NONE - NO PRIVILEGES
- NORMAL - MINIMUM PRIVILEGES TO USE THE SYSTEM EFFECTIVELY
- GROUP - POTENTIAL TO INTERFERE WITH MEMBERS OF THE SAME GROUP
- DEVOUR - POTENTIAL TO CONSUME NONCRITICAL SYSTEM-WIDE RESOURCES
- SYSTEM - POTENTIAL TO INTERFERE WITH NORMAL SYSTEM ACTIVITY
- FILE - POTENTIAL TO COMPROMISE FILE SECURITY
- ALL - POTENTIAL TO CONTROL THE SYSTEM

SYSPRV
CMKRNL
READALL
BYPASS

NOTE

Privileges are listed in Table 5-~~1~~ 30 (pg 5-~~54~~) of the VS Guide To VAX/VMS System Security and are detailed in Appendix A of that manual.

TABLE 7-8: ASSIGNING AND MANAGING PRIVILEGES

- PRIVILEGES SERVE AN IMPORTANT FUNCTION
- THE MISUSE OF PRIVILEGES WILL COMPROMISE SYSTEM SECURITY
- DON'T OVEREXTEND THE GRANTING OF PRIVILEGES
- ONCE GRANTED IN THE UAF THE USER HAS THEM AT EACH LOGIN
- ADVISE PRIVILEGED USERS TO TAKE AWAY PRIVILEGES THEY ARE NOT USING WITH $ SET PROCESS/PRIVILEGE=(NOPRIV-NAME)
- KNOW WHAT ALL PRIVILEGES ARE INTENDED FOR AND WHAT THEY MAY ALSO ALLOW
- KNOW YOUR USERS
- BE CONSERVATIVE IN GRANTING PRIVILEGES
- MINIMUM PRIVILEGES FOR SYSTEM USERS

Type of User	Minimum Privileges
General Users	TMPMBX,NETMBX
Operators	OPER
Group Managers	GROUP,GRPPRV
System Manager/Administrator	SYSPRV,SETPRV
Security Manager	SETPRV,SECURITY

TABLE 7-9: MANAGING PRIVILEGED ACCOUNTS

- PRIVILEGED ACCOUNTS WILL BE THE MOST LIKELY TARGET FOR BREAKINS
- ENFORCE GOOD PASSWORD MANAGEMENT ON PRIVILEGED ACCOUNTS
- AVOID OBVIOUS USERNAMES FOR PRIVILEGED ACCOUNT OPERATIONS, SECURITY, RECORDS, MANAGEMENT
- RESTRICT ACCESS TO OPERATIONAL PRIVILEGED ACCOUNTS
  - /LOCAL /NODIALUP /NOREMOTE /NONETWORK /NOBATCH
  - USE SECONDARY PASSWORDS - EACH KNOWN BY A DIFFERENT PERSON
  - SET ACCOUNT /NOACCESS WHEN NOT IN USE
  - SET UP A CAPTIVE ACCOUNT (SEE SECTION 7.3)
- GIVE PRIVILEGED USERS TWO ACCOUNTS
  - SAME NON-SYSTEM UIC AND SAME DEFAULT DIRECTORY
  - CAN ACCESS ALL FILES FROM EITHER ACCOUNT
  - DIFFERENT PASSWORDS, DIFFERENT NONOBVIOUS USERNAMES
  - ONE WITH MINIMUM PRIVILEGES TO DO REGULAR WORK
  - ONE WITH PRIVILEGES THAT IS DISTINCT AND EXCEPTIONAL
  - USE IT FOR NECESSARY PRIVILEGED WORK
  - LOG ACTIVITY ON PRIVILEGED ACCOUNT

- CHECKPRIV.COM
  - A COMMAND PROCEDURE THAT WILL LOG ACTIVITY OF PRIVILEGED ACCOUNTS
  - INSERTED INTO SYS$MANAGER:SYLOGIN.COM
  - WILL BE INVOKED EVERY TIME A USER WITH PRIVILEGE ATTEMPTS TO LOGIN TO THE SYSTEM
  - PROMPTS FOR THE REASON THAT THE PRIVILEGED USER WANTS TO LOG IN
  - SEE EXAMPLE 7-3
  - RECORDS THIS INFORMATION IN SYS$MANAGER:PRIV.LOG
  - REPORT_PRIV.COM WILL SEND A REPORT ON PRIVILEGED LOGINS TO THE SYSTEM MANAGER WEEKLY
  - A BATCH JOB THAT RESUBMITS ITSELF TO RUN EACH WEEK
  - SEE EXAMPLE 7-4
  - SEE CODE IN APPENDIX A

EXAMPLE 7-3: USER VIEW OF CHECKPRIV.COM

```
PLEASE STATE YOUR REASON FOR LOGGING INTO THIS PRIVILEGED ACCOUNT.
REASON:  TO ADD A NEW USER TO PROJECT X
THANK YOU
```

EXAMPLE 7-4: MAIL MESSAGE FROM REPORT_PRIV.COM

```
FROM:  SYSTEM                    28-JAN-1985 15:20
TO:    SYSTEM
SUBJ:  REPORT ON USAGE OF PRIVILEGED ACCOUNTS

28-JUN-1985 15:09:13.64 SARA    INTERACTIVE VTA53:
TO ADD A NEW USER TO PROJECT X
28-JUN-1985 16:18:32.24 SCOTT   INTERACTIVE VTA42:
ADD GROUP LOGICAL NAMES
```

TABLE 7-10: INSTALLING KNOWN IMAGES

- AN ALTERNATIVE TO GRANTING CERTAIN PRIVILEGES
- ALLOWS A USER TO RUN A A PARTICULAR PROGRAM THAT REQUIRES ONE OF THE MORE POWERFUL PRIVILEGES
- ALSO USED FOR USER-WRITTEN SYSTEM SERVICES
  - PRIVILEGED SHAREABLE IMAGES
  - INSTALL /PROTECT
  - SEE APPENDIX D - VAX/VMS RELEASE NOTES = VERSION 4
- DON'T NEED THE PRIVILEGE AT OTHER TIMES
- INSTALL THE IMAGE WITH PRIVILEGES
- PUT AN ACL ON THE FILE TO ALLOW EXECUTE ACCESS FOR THAT USER
- THE USERS WOULD EFFECTIVELY POSSESS THE PRIVILEGE ONLY WHEN THEY ARE ACTUALLY EXECUTING THE IMAGE
- THE USER NO LONGER HOLDS THE PRIVILEGE WHEN THE IMAGE STOPS RUNNING
- PROTECTED FROM CTRL/Y - OLD PRIVILEGES RESTORED WHEN INTERRUPTED
- BEWARE OF INSTALLING USER WRITTEN IMAGES WITH PRIVILEGES
- YOU MUST KNOW THE IMAGE

- READ THE SOURCE OF USER-WRITTEN IMAGES
- COMPILE AND LINK IT YOURSELF
- LINK WITH THE /NOTRACE AND /NODEBUG QUALIFIERS TO PREVENT ONLINE DEBUGGING OR TRACEBACK
- ALL DIGITAL SUPPLIED SYSTEM PROGRAMS ARE LINKED /NOTRACE /NODEBUG
- INSTALL CHECKS FOR /TRACE AND /DEBUG

EXAMPLE 7-5: INSTALLING KNOWN IMAGES WITH PRIVILEGES

```
$ RUN SYS$SYSTEM:INSTALL

INSTALL> SDA.EXE /PRIVILEGED=CMKRNL
INSTALL> EXIT
$ SET ACL/ACL=(ident=[*,*],ACCESS=NONE) SYS$SYSTEM:SDA.EXE   !  let last ends up 2nd in list.
$ SET ACL/ACL=(IDENTIFIER=SDAUSERS,ACCESS=EXECUTE) -
  SYS$SYSTEM:SDA.EXE

$ SET PROTECTION=(WORLD) SYS$SYSTEM:SDA.EXE

$ SET DEFAULT SYS$SYSTEM

$ RUN AUTHORIZE
UAF> SHOW/IDENTIFIER=SDAUSERS

     CHECK LIST TO CONFIRM THAT ONLY THE USERS
     YOU WANT HOLD THE IDENTIFIER SDAUSERS

     GRANT THE IDENTIFIER AS NECESSARY
```

7.3 CAPTIVE COMMAND PROCEDURES

There are many reasons why you may want to establish CAPTIVE privileged accounts. Special care must be taken in establishing these accounts. The term captive is difficult to apply accurately to command procedures. Too often these accounts give a false sense of security.

Table 7-11 lists some reasons why you may need to have a captive account.

TABLE 7-11: WHY USE CAPTIVE ACCOUNTS

- WANT SEMI-SKILLED USERS TO PERFORM ROUTINE OPERATIONS UNDER TIGHT CONTROL
- WANT CERTAIN OPERATIONS TO RUN DURING HOURS WHEN THERE IS LITTLE SUPERVISION
- HAVE SPECIAL APPLICATIONS THAT NEED LIMITED ACCESS TO THE SYSTEM
  I.E. PRINTING PAYCHECKS, REPORTING STUDENT GRADES, ORDER ENTRY
- NEED SPECIAL PURPOSE PRIVILEGED ACCOUNTS
  I.E. BACKUP ACCOUNT WITH READALL PRIVILEGE
- WANT TO LOG USE OF SPECIAL ACCOUNTS

DCL command procedures may not be very secure. There are ways out of supposed CAPTIVE command procedures. One thing to never do is use an INQUIRE statement to take in input from the user. The INQUIRE statement performs DCL symbol substitution and allows the execution of LEXICAL functions. General guidelines for writing captive command procedures are given in Table 7-12.

TABLE 7-12: GUIDELINES FOR ESTABLISHING CAPTIVE COMMAND PROCEDURES

- MAKE THE PROCEDURE PART OF SYS$SYLOGIN OR /LGICMD
- DON'T USE INQUIRE use READ/ERR instead.
- USE $ READ/ERROR=LABEL/PROMPT=PROMPT SYS$COMMAND SYMBOL
- KEEP THE MENU ITEMS TO 4 OR LESS
- DON'T EVALUATE SYMBOL ('SYMBOL')

  I.E. $ SET HOST 'NODE'
- USE F$LOCATE TO SEARCH INPUT SYMBOL FOR STRINGS WHICH WOULD INDICATE A BREAKOUT ATTEMPT ("F$","F$","@","=")
- FOR BACKUP COMMAND PROCEDURES DON'T INPUT THE NAME OF THE DEVICE - CODE IT IN
- SET THE FILE AND DIRECTORY PROTECTION TO EXECUTE ONLY
- /FLAGS=(CAPTIVE, DISCTLY, LOCKPWD, DEFCLI, DISWELCOME, DISMAIL, DISNEWMAIL) Set on privileged account depending on application. Allinl needs 2.
- MAIL CAN SPAWN OTHER PROCESSES - DISALLOW IT IF POSSIBLE (not in VMS 5.0)
- SET PRCLM QUOTA TO 0 FOR ACCOUNTS THAT USE MAIL

Allinl DCL flag
Do flag.

EXAMPLE 7-6: GUIDELINES FOR CAPTIVE COMMAND PROCEDURE

```
$ ON CONTROL_Y THEN LOGOUT
$ ON ERROR THEN LOGOUT
$ DELETE = "DELETE"                    !DELETE ALL SYMBOLS
$ DELETE/SYMBOL/GLOBAL/ALL
$ DELETE/SYMBOL/LOCAL/ALL
$!
$ READ/PROMPT="...."/ERROR=EXIT  SYS$COMMAND  ANSWER
$ !
$ ! CHECK FOR LEXICAL FUNCTIONS
$ !
$ IF F$LOCATE("F$", ANSWER) .NE. F$LENGTH(ANSWER) THEN LOGOUT
$ IF F$LOCATE("f$", ANSWER) .NE. F$LENGTH(ANSWER) THEN LOGOUT
$ !
$ ! CHECK FOR AN ATTEMPT TO EXECUTE A COMMAND PROCEDURE
$ !
$ IF F$LOCATE("@", ANSWER) .NE. F$LENGTH(ANSWER) THEN LOGOUT
$ !
$ ! CHECK FOR THE TASK OBJECT
$ !
$ IF F$LOCATE("=", ANSWER) .NE. F$LENGTH(ANSWER) THEN LOGOUT
$ !
        .
        .
        .
        .
$ EXIT:
$       LOGOUT
```

Example 7-7 shows a command procedure that uses the INQUIRE statement and a terminal session with the problems that can result.

EXAMPLE 7-7: INSECURE CAPTIVE COMMAND PROCEDURE

```
$!  INSECURE.COM
$!
$   SAVE_VERIFY = F$VERIFY(0)
$!
$ GET_PASSWORD:
$       ON WARNING THEN GOTO LOGOUT
$       INQUIRE PASS "Password"
$ IF PASS .NES. "SECRET" THEN GOTO LOGOUT
$!
$       WRITE SYS$OUTPUT "YOU GUESSED THE PASSWORD = ''PASS'"
$       IF SAVE_VERIFY THEN SET VERIFY
$       EXIT
$!
$ LOGOUT:
$       WRITE SYS$OUTPUT "LOGGING OUT . . ."
$       STOP/ID=0
```

Terminal Session :

```
$ @INSECURE
Password: "''F$VERIFY(1)'"
$ IF PASS .NES. "SECRET" THEN GOTO LOGOUT
$ LOGOUT:
$       WRITE SYS$OUTPUT "LOGGING OUT . . ."
$       STOP/ID=0
```

NOTE

The next time through the same procedure the password will be known. Example 7-8 shows a way to correct the problems that occurred in Example 7-7.

EXAMPLE 7-8: MORE SECURE CAPTIVE COMMAND PROCEDURE

```
$!   SECURER.COM
$!
$    SAVE_VERIFY = F$VERIFY(0)
$!
$ GET_PASSWORD:
$      ON WARNING THEN GOTO LOGOUT
$      READ/ERROR=LOGOUT/PROMPT="Password: " SYS$COMMAND  PASS
$ IF PASS .NES. "SECRET" THEN GOTO LOGOUT
$!
$      WRITE SYS$OUTPUT "YOU GUESSED THE PASSWORD = SECRET"
$      IF SAVE_VERIFY THEN SET VERIFY
$      EXIT
$!
$ LOGOUT:
$      WRITE SYS$OUTPUT "LOGGING OUT . . ."
$      STOP/ID=0
```

Terminal Session :

```
$ @SECURER
Password: "''F$VERIFY(1)'"
LOGGING OUT . . .
```

NOTE

In the above examples, it is assumed that Control Y is disabled before entering this procedure and that the user will execute these procedures. (i.e. /FLAGS=CAPTIVE)

TABLE 7-13: BYPASSING CAPTIVE COMMAND PROCEDURES

- USERNAME: FINCH/DISK=TT:

  THIS CAUSES LOGIN.COM TO BE PROMPTED FOR AND READ FROM THE TERMINAL

- USERNAME: FINCH/NOCOMMAND

  THIS CAUSES NO USER LEVEL LOGIN COMMAND PROCEDURE TO BE EXECUTED

- USERNAME: FINCH/CLI=MCR

  THIS CAUSES THE SEARCH FOR LOGIN.CMD

NOTE

All of these methods of bypassing the login command procedure can be prevented by establishing the account as /FLAGS=CAPTIVE in the UAF record.

GUEST AND MAIL ACCOUNTS

The practice of setting up "guest" accounts is not recommended. However, sometimes a "guest" account is needed for people to read/send mail or to read reports on a given node.

Table 7-14 lists some recommendations to follow in setting up these accounts. These recommendations can be checked by a command procedure.

TABLE 7-14: GUIDELINES FOR ESTABLISHING A GUEST ACCOUNT

- USE AN OBSCURE PASSWORD
- DON'T HAVE A GUEST/GUEST OR USER/USER ACCOUNT
- MAINTAIN A LIST OF USERS ALLOWED TO USE THE ACCOUNT
- PUT THE GUEST ACCOUNT IN IT'S OWN NONSYSTEM UIC GROUP
- MOVE THE DEFAULT LOGIN COMMAND PROCEDURE AND RELATED FILES TO A PROTECTED DIRECTORY I.E. SYS$MANAGER: (MAKES IT MORE DIFFICULT TO TAMPER WITH IT)
- FOLLOW GUIDELINES OF EXAMPLE 7-6
- UAF> MODIFY GUEST-ACCOUNT/FLAGS=(DISCTLY, DEFCLI, LOCKPWD, CAPTIVE)
- LIMIT THE NUMBER OF SUBPROCESSES THAT THE ACCOUNT CAN CREATE TO 0
  - UAF> MODIFY GUEST-ACCOUNT/PRCLM=0
  - PREVENTS SPAWN
  - PREVENTS MAIL>SPAWN

- GIVE ACCOUNT ONLY TMPMBX PRIVILEGE

NETMBX possibly MOUNT

- RESTRICT ACCESS TO PREVENT /NOBATCH
  - PREVENTS MAIL USERS FROM EXECUTING COMMAND PROCEDURES
  - MAIL> PRINT/QUEUE=SYS$BATCH

- MAIL AND PHONE ARE INSTALLED WITH TMPMBX AND NETMBX
  - MAY WANT TO REINSTALL THESE IMAGES WITHOUT NETMBX
  - WILL DENY NETWORK ACCESS TO MAIL/PHONE USERS WITHOUT NETMBX PRIVILEGE
  - USERS WITH NETMBX WILL STILL BE ABLE TO ACCESS NETWORK

```
$ RUN SYS$SYSTEM:INSTALL
INSTALL> MAIL/DELETE
INSTALL> MAIL/PRIV=(TMPMBX)/OPEN/SHARE
```

PSI mail

create different default a/c for PSI mail without NETMBX for PSI access to use.

Prevents users using PSI object from using your system to jump to other systems.

AUTOMATIC LOGIN FACILITY

The Automatic Login Facility permits you to enable a particular terminal to be tied to a specific account. When a user access the terminal he/she is automatically tied in to the account to which the terminal is associated.

TABLE 7-15: USING THE AUTOMATIC LOGIN FACILITY

- <RETURN> OR <BREAK> ENTERED AT THE TERMINAL ACTIVATES THE ALF FACILITY
- WILL NOT GIVE USERNAME: PROMPT
- MAY DISABLE OR ENABLE PASSWORDS AS WITH ANY ACCOUNT
- ALF ONLY SUPPRESSES USERNAME
- MAY ALSO WANT TO SUPPRESS LOGIN MESSAGES
- RECORDS KEPT IN SYS$SYSTEM:SYSALF.DAT
- ONE RECORD PER TERMINAL
- ASSOCIATES A TERMINAL WITH A USERNAME
- MULTIPLE TERMINALS CAN BE TIED TO THE SAME ACCOUNT
- CANNOT USE THAT TERMINAL FOR ANY OTHER ACCOUNT
- GOOD FOR UNSOPHISTICATED USERS
- OFTEN WILL LOGIN TO A CAPTIVE ACCOUNT
- MANAGE FACILITY WITH SYS$MANAGER:ALFMAINT.COM
  SEE EXAMPLE 7-9

EXAMPLE 7-9: ADDING RECORDS TO THE AUTOMATIC LOGIN FACILITY

```
$ SET DEFAULT SYS$MANAGER:

$ @ALFMAINT
Enter the name of the terminal that you would like to set
for automatic login, or a blank line or EXIT to exit.

Terminal (ddcu)? TTD6:

Enter the username you would like to automatically login on TTD6:
Enter a blank line to display and optionally delete the record for TTD6:

Username? ACCOUNTS_PAYABLE
Terminal TTD6: user ACCOUNTS_PAYABLE record added.

Enter the name of the terminal that you would like to set
for automatic login, or a blank line or EXIT to exit.

Terminal (ddcu)? TTD7:

Enter the username you would like to automatically login on TTD7:
Enter a blank line to display and optionally delete the record for TTD7:

Username? ACCOUNTS_PAYABLE
Terminal TTD7: user ACCOUNTS_PAYABLE record added.
```

EXAMPLE 7-10: REMOVING A RECORD FROM THE ALF FACILITY

```
$ @ALFMAINT
Enter the name of the terminal that you would like to set
for automatic login, or a blank line or EXIT to exit.

Terminal (ddcu)? TTD7

Enter the username you would like to automatically login on TTD7:
Enter a blank line to display and optionally delete the record for TTD7:

Username?
Terminal TTD7:, current user is ACCOUNTS_PAYABLE.
Do you want to delete this record (Y/N)? Y
Terminal TTD7: record deleted.
```

7.4 REMOVING DCL COMMANDS

DCL command tables may be modified by the COMMAND DEFINITION UTILITY. A process' copy of the DCLTABLES may be modified or a modified copy of the DCLTABLES may be specified in AUTHORIZE.

TABLE 7-16: REMOVING DCL COMMANDS

- USE SET COMMAND COMMAND OF THE COMMAND DEFINITION UTILITY
- SYS$LIBRARY:DCLTABLES.EXE IS THE MASTER COPY OF THE TABLES USED TO INTERPRET DCL COMMANDS
- BY DEFAULT THEY ARE COPIED TO THE PROCESS' P1 VIRTUAL ADDRESS SPACE AT LOGIN
- MASTER COPY, PROCESS' COPY, OR AN ALTERNATE COPY MAY BE MODIFIED
- $ SET COMMAND/DELETE=COMMAND

  REMOVES THAT COMMAND FROM THE PROCESS' COPY IN MEMORY

  PLACE THAT COMMAND IN THE USERS /LGICMD LOGIN FILE
- $ SET COMMAND /DELETE=COMMAND /TABLES=SYS$LIBRARY:DCLTABLES /OUTPUT=NEW_TABLE

  CREATES A NEW TABLE WITH THE COMMAND REMOVED
- UAF> MODIFY USERNAME/CLI_TABLES=NEW_TABLE - /FLAGS=CAPTIVE

  SPECIFIES THAT THE USER WILL AUTOMATICALLY USE THE MODIFIED COPY OF THE TABLES
- INSTALL> NEW_TABLES/OPEN/SHARE

EXAMPLE 7-11: COMMANDS TO REMOVE DCL COMMANDS

```
$ SET PROTECTION=W      SYS$LIBRARY:DCLTABLES.EXE
$ DIRECTORY/SECURITY  SYS$LIBRARY:DCLTABLES.EXE
DCLTABLES.EXE;1     [VMS,SYSTEM]     (RWED,RWED,RE,)
```

ADD THE FOLLOWING LINE TO THE USERS /LGICMD

```
    $ SET COMMAND/DELETE=MOUNT
```

EXAMPLE 7-12: REMOVING DCL COMMANDS = USERS VIEW

```
$ MOUNT
%DCL-W-IVVERB, UNRECOGNIZED COMMAND VERB
- CHECK VALIDITY AND SPELLING
 \MOUNT\

$ SET COMMAND/TABLES=SYS$LIBRARY:DCLTABLES.EXE
%CDU-F-OPENIN, ERROR OPENING SYS$COMMON:[SYSLIB]DCLTABLES.EXE;
-RMS-E-PRV, INSUFFICIENT PRIVILEGE OR FILE PROTECTION VIOLATION
```

NOTE

The default protection on SYS$LIBRARY:DCLTABLES.EXE is (S:RWED,O:RWED,G;RE,W;RE). In order to prevent a user from reestablishing the old tables, they must be denied READ access to the file. This presents no problems for INTERACTIVE, NETWORK, and BATCH logins as LOGINOUT.EXE (the program that reads the table at process creation time) will run in the context of a privileged process.

It does however prevent a user from creating a process with SPAWN or using $CREPRC system service and specifying LOGINOUT.EXE as the image to be run. For those users who need read access to DCLTABLES, an ACL can be placed on the file.

CHAPTER 8
SECURITY AUDITING

LIST OF TOPICS

- Event Detection and Logging
  - User Auditing
  - System/Security Manager Auditing
- Auditing Methods
  - DCL Commands
  - MONITOR UTILITY
  - SECURITY AUDITING
  - ACCOUNTING Utility
  - Auditing World Readable Files
- Auditing Checklist

8.1 EVENT DETECTION AND LOGGING

Auditing or monitoring of your system is an essential practice in maintaining a secure system.

VMS provides tools which help you keep extensive logging of specific system events considered significant to system security. These tools include informational DCL commands, the MONITOR utility, the ACCOUNTING utility, and SECURITY AUDITS. AUDIT ALARMS are the only automatic security auditing component offered by VMS V4. Audit alarms provide summary information of security related events on video or hard copy terminals, and to the operator's log file.

Security auditing cannot be effective without frequent analysis of the audit records.

Much can also be gained by users' and your observation. Don't overlook the more routine observations of your system because you think that the audit trail is being kept. Methods of auditing are discussed in Section 8.2.

FOR AUDITING TO BE EFFECTIVE
SOMEONE MUST READ THE AUDIT RECORDS

USER AUDITING

THE USERS ARE A NECESSARY AND USEFUL AID IN OBSERVING THE SYSTEM. THEY NEED TO BE EDUCATED AND MADE AWARE OF GOOD PRACTICES THAT WILL AID IN MAINTAINING SYSTEM SECURITY.

- EDUCATE YOUR USERS TO WATCH FOR
  - MISSING FILES
  - UNEXPLAINED NEW COMMAND PROCEDURES OR PROGRAMS
  - LAST LOGIN AND LOGIN FAILURES MESSAGES THAT DON'T MATCH WITH THE USER'S RECOLLECTION OF EVENTS
  - MISSING MAIL MESSAGES
  - SHOW USERS COMMAND INDICATING THAT THE USER IS LOGGED ON AT ANOTHER TERMINAL
  - DISCONNECTED JOB MESSAGE APPEARING FOR A PROCESS NEVER INITIATED BY THE USER
  - SHOW QUOTA COMMAND INDICATING MUCH MORE SPACE BEING USED THAN DIR/TOTAL/SIZE=ALL SHOWS
  - SUDDEN DROP IN AVAILABILITY OF RESOURCES (DISK, CPU, DIALUP LINES)
- LISTEN TO THE USERS
- FOLLOW UP VALID OBSERVATIONS
- GET AND KEEP THE USERS' COOPERATION
- KNOW YOUR USER COMMUNITY

SYSTEM/SECURITY MANAGER AUDITING

YOU ARE RESPONSIBLE FOR AUDITING YOUR SYSTEM. THERE ARE MANY THINGS THAT SHOULD BE DONE ON A REGULAR BASIS TO WATCH FOR ATTEMPTED BREACHES OF SECURITY.

- KNOW YOUR SYSTEM
- SHOW USERS INDICATES A USER WHO YOU KNOW CANNOT OR SHOULD NOT BE LOGGED IN AT THAT TIME
- THERE IS AN UNEXPLAINED CHANGE IN SYSTEM RESPONSE
- PHYSICAL SECURITY HAS BEEN COMPROMISED I.E. A FILE CABINET, DESK HAS BEEN ENTERED MEDIA IS MISSING
- CHECK SYS$SYSTEM: AND SYS$MANAGER: FOR UNEXPLAINED PROGRAMS, COMMAND PROCEDURES
- USERS YOU DID NOT AUTHORIZE ARE ON THE SYSTEM
- ACCOUNTING UNCOVERS UNUSUAL AMOUNTS OF ACTIVITY AT NORMALLY QUIET TIMES
- NETUAF.DAT, SYSUAF.DAT, RIGHTSLIST.DAT HAVE ENTRIES FOR WHICH YOU CANNOT ACCOUNT
- THE COMPANY IS HAVING PERSONNEL, MORALE, AND/OR ECONOMIC DIFFICULTIES

8.2 AUDITING METHODS

DCL COMMANDS FOR AUDITING

$ SHOW USERS

- WILL SHOW INTERACTIVE USERS
- REQUIRES NO PRIVILEGE
- DISPLAYS USERNAMES, PIDS, AND TERMINALS
- WILL SHOW REMOTE TERMINAL USERS
- ALLOWS YOU TO

  KEEP AN EYE ON WHO'S LOGGED ON

  LOOK FOR UNKNOWN USERS OR USERS LOGGED IN AT UNUSUAL TIMES OR TERMINALS

EXAMPLE 8-1: $ SHOW USERS COMMAND

```
VAX/VMS Interactive Users 25-JAN-1985 09:55:17.39
  Total number of interactive users = 7
Username    Process Name     PID       Terminal
DEMO        DEMO             21A0114E  VTA1650       TTH3:
FALCON      FALCON           21A01147  RTA1:
FINCH       _VTA1653:        21A01151  VTA1653       TTH2:
HAWK        Bird Man         21A0124F  VTA1651       TTH4:
ROBIN       ROBIN            21A0114A  VTA1647       TTB6:
SWALLOW     Cliff Hanger     21A01248  VTA1648       TTB7:
WARBLER     Lucy             21A0123D  VTA1636       TTH0:
WREN        WREN             21A014F3  VTA1660    disconnected
```

$ SHOW SYSTEM

- DISPLAYS ALL PROCESSES ON THE SYSTEM
  SEE EXAMPLE 8-2
- REQUIRES NO PRIVILEGE
- /FULL WILL ALSO DISPLAY UIC'S
- /NETWORK, /BATCH, /SUB, /DETACHED CAN SELECT PROCESS TYPES TO DISPLAY
- LOOK FOR UNUSUAL AMOUNTS OF ACTIVITY
  - LARGE AMOUNTS OF TOTAL CPU TIME
  - LARGE AMOUNTS OF TOTAL I/O
  - LARGE NUMBERS OF TOTAL PAGE FAULTS

$ SHOW PROCESS/CONTINUOUS/ID=PID

- DISPLAYS MOST INFORMATION ABOUT A SINGLE USER
- SHOWS IMAGE THEY ARE RUNNING
- SHOWS UIC, PID, WORKING SET, CPU TIME, PC
- REQUIRES WORLD PRIVILEGE TO LOOK AT ANOTHER USER

$ SHOW QUEUE/BATCH/ALL/FULL

- LOOKS AT ALL BATCH JOBS
- REQUIRES OPER PRIVILEGE OR EXECUTE ACCESS TO THE QUEUE

EXAMPLE 8-2: $ SHOW SYSTEM COMMAND

```
VAX/VMS V4.0 on node DUCK 24-JAN-1985 20:19:24.02  Uptime  2 10:11:03
  Pid     Process Name    State Pri  I/O       CPU       Page flts Ph.Mem
21C00080 NULL            COM   0      0  1 23:41:34.84       0       0
21C00081 SWAPPER         HIB  16      0  0 00:02:56.26       0       0
21C00085 ERRFMT          HIB   7   2052  0 00:00:40.28      68      89
21C00086 CACHE_SERVER    HIB  16      6  0 00:00:00.28      57      73
21C00087 CLUSTER_SERVER  HIB  10     16  0 00:00:09.46     102     225
21C00088 OPCOM           LEF   7   2455  0 00:00:55.19    3901      59
21C00089 JOB_CONTROL     HIB   9  33752  0 00:13:30.10     176     321
21C0008A CONFIGURE       HIB  13     27  0 00:00:00.53     102     119
21C00392 NETACP          HIB  10   1032  0 00:00:32.03     297     201
21C00513 EVL             HIB   6     55  0 00:00:05.64    8198      41 N
21C00594 REMACP          HIB   9     87  0 00:00:00.68      73      41
21C0060A SYSTEM          CUR   9    136  0 00:00:18.24     496     245
21C00719 BATCH_234       LEF   9    236  0 00:00:07.16     869     150 B
21C0009E DBMS_MONITOR    LEF   5    126  0 00:00:04.94     528     384
21C00330 SYMBIONT_0004   HIB   6   3598  0 00:03:15.76   39423      48
21C005B7 ROBIN           LEF   4  69544  0 00:19:46.08   31068     246
21C00744 FINCH           LEF   5    586  0 00:00:13.41     677     510
```

EXAMPLE 8-3: $ SHOW QUEUE/BATCH/ALL/FULL COMMAND

```
Batch queue SYS$BATCH started , on DUCK:: /BASE_PRIORITY=4
/JOB_LIMIT=2 /OWNER=[VMS,SYSTEM] /PROTECTION=(S:E,O:D,G:R,W:W)

 Jobname       Username     Entry  Status
 -------       --------     -----  ------
 FINDSEEDS      FALCON       698  Executing
  Submitted 28-JAN-1985 18:16 /NOLOG /NOTIFY /PRIO=4
    _DUA1:[FALCON]FINDSEEDS.COM;5 (executing)
 MIDNIGHT       SYSTEM       650  Holding until 29-JAN-1985 00:00
  Submitted 28-JAN-1985 00:55 /CPU=01:00:00 /KEEP /NOPRINT /PRIO=4
    _DUA0:[SYS0.SYSMGR]MIDNIGHT.COM;3
```

MONITOR UTILITY

$ MONITOR

- INTERACTIVE DISPLAY
- CAN RECORD INFORMATION TO PLAY BACK LATER
- SHOWS YOU WHERE TO LOOK FURTHER
- MONITOR SYSTEM GIVES OVERALL PICTURE
  SEE EXAMPLE 8-4
- MONITOR PROCESS/TOPCPU USEFUL FOR TRACKING OVERALL SYSTEM USAGE

EXAMPLE 8-4: $ MONITOR SYSTEM

```
Node: DUCK        VAX/VMS Monitor Utility      24-JAN-1985 20:12:52
Statistic: CURRENT                SYSTEM STATISTICS
                                                Process States
          + CPU Busy (55)        -+       LEF:    18      LEFO:     0
          |***************        |       HIB:    10      HIBO:     0
CPU     0+-----------------------+ 100    COM:     1      COMO:     0
          |****                   |       PFW:     0      Other:    5
          +-----------------------+       MWAIT:   0
          Cur Top: PERCH (11)                        Total: 34

          + Page Fault Rate (4) -+         + Free List Size (4843) -+
          |***                    |         |**********                | 12K
MEMORY  0+-----------------------+ 100  0+-------------------------+
          |                       |         |*********                 | 500
          +-----------------------+         + Modified List Size(185)+
          Cur Top: EVL (2)

          + Direct I/O Rate (0) -+         + Buffered I/O Rate (0) -+
          |                      |         |                        |
I/O     0+-----------------------+ 60   0+-------------------------+ 150
          |                      |         |                        |
          +----------------------+         +------------------------+
          Cur Top:  (0)                    Cur Top:  (0)
```

SECURITY AUDITING

TABLE 8-1: SECURITY AUDITING

- CAN ENABLE A VARIETY OF ALARMS
- CAN AUDIT LOGFAIL AND BREAKIN ATTEMPTS
- CAN AUDIT TYPES OF LOGINS AND LOGOUTS
- CAN AUDIT SUCCESSFUL AND FAILED ATTEMPTS AT DIFFERENT TYPES OF FILE ACCESS
- CAN AUDIT THE USE OF PRIVILEGE TO ACCESS FILES
- CAN AUDIT WHEN SECURITY ALARM ACLS ARE USED
- CAN AUDIT ACTIVITY OF AN INDIVIDUAL ACCOUNT

  UAF> MODIFY EAGLE/FLAGS=AUDIT

- ALARMS GO TO ALL SECURITY OPERATORS TERMINALS AND OPERATORS LOG FILE

  ```
  $ REPLY/ENABLE=SECURITY
  $ REPLY/LOG
  $ PRINT SYS$MANAGER:OPERATOR.LOG;-1
  $ SEARCH/OUTPUT=ALARMS.LIS/WINDOWS=(3,6) -
      SYS$MANAGER:OPERATOR.LOG;-1   "SECURITY ALARM"
  ```

- $ SHOW AUDIT SHOWS WHAT IS CURRENTLY ENABLED

- $ SET AUDIT /ALARM /DISABLE=KEYWORD OR /ENABLE=KEYWORD ENABLES OR DISABLES SECURITY AUDITING
  - INDICATES THAT ONE OR MORE EVENTS SHOULD OR SHOULD NOT RESULT IN ANY ACTION
  - /ALARM QUALIFIER MUST BE USED WITH EITHER THE /ENABLE OR /DISABLE QUALIFIER
  - CAN ENABLE/DISABLE ALL EVENTS BY SPECIFYING /ENABLE=ALL OR /DISABLE=ALL
  - CAN ENABLE/DISABLE ALARMS FOR SELECTIVE EVENTS
  - THE SYSTEM PROCESSES THE /DISABLE QUALIFIER LAST
  - KEYWORDS
    - = ACL - AN EVENT REQUESTED BY A SECURITY ALARM ACL
    - = AUTHORIZATION - MODIFICATION OF NETUAF.DAT OR OF THE SYSUAF.DAT *NETPROXY.DAT in V5.*
    - = MOUNT - THE ISSUING OF A MOUNT OR DISMOUNT REQUEST
    - = BREAKIN=(KEYWORD[,...])
    - = LOGFAILURE=(KEYWORD[,...])
    - = LOGIN=(KEYWORD[,...])
    - = LOGOUT=(KEYWORD[,...])

      =(ALL, DETACHED, DIALUP, LOCAL, NETWORK, REMOTE, SUBPROCESS)
    - = FILE_ACCESS=(KEYWORD[,...])

      =(ALL, BYPASS, GRPPRV, READALL, SYSPRV, = FAILURE, SUCCESS, = ALL, READ, WRITE, EXECUTE, DELETE, CONTROL)

EXAMPLE 8-5: $ SET AUDIT /ALARM /ENABLE = (LOGFAILURE = ALL,BREAKIN = ALL)

```
%%%%%%%%%%  OPCOM  24-FEB-1985 15:46:01.92  %%%%%%%%%%
Security alarm on CANADA / Local interactive login failure
    Time:         24-FEB-1985 15:46:01.90
    PID:          21C00549
    User Name:    SYSTEM
    Status:       %LOGIN-E-NOTVALID, user authorization failure
    Dev Name:     _VTA584:  (_TTA4:)

                  . . . .

%%%%%%%%%%  OPCOM  24-FEB-1985 15:46:07.37  %%%%%%%%%%
Security alarm on CANADA / Local interactive breakin detection
    Time:         24-FEB-1985 15:46:07.26
    PID:          21C00549
    User Name:    SYSTEM
    Password:     VAX_VMS
    Dev Name:     _VTA584:  (_TTA4:)

                  . . . .

%%%%%%%%%%  OPCOM  24-FEB-1985 23:24:42.02  %%%%%%%%%%
Security alarm on CANADA / Remote interactive breakin detection
    Time:         24-FEB-1985 23:24:41.98
    PID:          20C0052E
    User Name:    GEESE
    Password:     CANADA
    Dev Name:     _RTA1:
    Source:       1.5  LAKE::PERCH

                  . . . .

%%%%%%%%%%  OPCOM  24-FEB-1985 23:51:01.45  %%%%%%%%%%
Security alarm on CANADA / Local interactive login failure
    Time:         24-FEB-1985 23:51:01.43
    PID:          21C0048D
    User Name:    <login>
    Status:       %LOGIN-F-CMDINPUT, error reading command input
    Dev Name:     _VTA595:  (_TTA4:)
```

EXAMPLE 8-6: $ SET AUDIT /ALARM /ENABLE = (LOGIN = SUCCESS = NETWORK)

```
%%%%%%%%%%  OPCOM  24-JAN-1985 15:39:26.97  %%%%%%%%%%
Security alarm on CANADA / Network login
    Time:         24-JAN-1985 15:39:26.95
    PID:          21C005C1
    User Name:    DECNET
    Source:       1.5  LAKE::PIKE

%%%%%%%%%%  OPCOM  24-JAN-1985 15:40:12.50  %%%%%%%%%%
Security alarm on CANADA / Network login
    Time:         24-JAN-1985 15:40:12.49
    PID:          21C006C3
    User Name:    FINCH
    Source:       1.6  NEST::21C004BB
```

EXAMPLE 8-7: AUDITING SECURITY ALARM ACLS

```
$ DIRECTORY/SECURITY DATA.LIST
DATA.LIST;1          SONG_BIRDS     (RWED,RWED,RE,)

(ALARM_JOURNAL=SECURITY,ACCESS=READ+WRITE
              +EXECUTE+DELETE+SUCCESS+FAILURE)

$ SET AUDIT/ALARM/ENABLE=(ACL)

$ SHOW AUDIT
Security alarms currently enabled for:
    ACL

                . . .

%%%%%%%%%%  OPCOM  25-JAN-1985 12:54:38.90  %%%%%%%%%%
Security alarm on DUCK / File access failure
    Time:         25-JAN-1985 12:54:38.89
    PID:          21A0114A
    User Name:    FALCON
    Image:        DUA1:[SYS0.][SYSEXE]TYPE.EXE
    File:         DUA1:[ACCOUNTS.DATA]DATA.LIST;1
    Mode:         READ
```

EXAMPLE 8-8: $ SET AUDIT /ALARM /ENABLE = (FILE ACCESS = FAILURE = ALL)

```
%%%%%%%%%%  OPCOM  24-JAN-1985 17:14:54.31  %%%%%%%%%%
Security alarm on CANADA / File access failure
    Time:          24-JAN-1985 17:14:54.29
    PID:           21C00719
    User Name:     FALCON
    Image:         DUA0:[SYS0.][SYSEXE]TYPE.EXE
    File:          DUA0:[SYS0.][SYSMGR]SYSTARTUP.COM;9
    Mode:          READ


%%%%%%%%%%  OPCOM  24-JAN-1985 17:16:07.28  %%%%%%%%%%
Security alarm on CANADA / File access failure
    Time:          24-JAN-1985 17:16:07.23
    PID:           21C00719
    User Name:     FALCON
    Image:         DUA0:[SYS0.][SYSEXE]DIRECTORY.EXE
    File:          DUA0:[SYS0.][SYSEXE]ACCOUNTNG.DAT;1
    Mode:          READ


%%%%%%%%%%  OPCOM  24-JAN-1985 17:16:42.13  %%%%%%%%%%
Security alarm on CANADA / File access failure
    Time:          24-JAN-1985 17:16:42.10
    PID:           21C00719
    User Name:     FALCON
    Image:         DUA0:[SYS0.][SYSEXE]ACC.EXE;1
    File:          DUA0:[SYS0.][SYSMGR]ACCOUNTNG.DAT;3
    Mode:          READ


%%%%%%%%%%  OPCOM  24-JAN-1985 17:17:54.77  %%%%%%%%%%
Security alarm on CANADA / File access failure
    Time:          24-JAN-1985 17:17:54.75
    PID:           21C00719
    User Name:     HAWK
    Image:         DUA0:[SYS0.][SYSEXE]EDT.EXE
    File:          DUA0:[SYS0]SYSMGR.DIR;1
    Mode:          READ WRITE
```

EXAMPLE 8-9: $ SET AUDIT /ALARM /ENABLE = (FILE_ACCESS = SUCCESS = EXECUTE)

```
%%%%%%%%%%  OPCOM  24-JAN-1985 15:35:55.61  %%%%%%%%%%
Security alarm on CANADA / Successful file access
     Time:          24-JAN-1985 15:35:55.51
     PID:           21C006A4
     User Name:     FINCH
     Image:         (None)
     File:          DUA0:[SYS0.][SYSEXE]FORTRAN.EXE;1
     Mode:          READ EXECUTE
     Privs Used:    (None)

%%%%%%%%%%  OPCOM  24-JAN-1985 15:35:56.73  %%%%%%%%%%
Security alarm on CANADA / Successful file access
     Time:          24-JAN-1985 15:35:56.71
     PID:           21C006A4
     User Name:     FINCH
     Image:         DUA0:[SYS0.][SYSEXE]FORTRAN.EXE;1
     File:          DUA0:[SYS0]SYSLIB.DIR;1
     Mode:          EXECUTE
     Privs Used:    (None)

%%%%%%%%%%  OPCOM  24-JAN-1985 15:35:57.36  %%%%%%%%%%
Security alarm on CANADA / Successful file access
     Time:          24-JAN-1985 15:35:57.33
     PID:           21C006A4
     User Name:     FINCH
     Image:         DUA0:[SYS0.][][SYSEXE]FORTRAN.EXE;1
     File:          DUA1:[000000]FINCH.DIR;1
     Mode:          EXECUTE
     Privs Used:    (None)
          . . . .


%%%%%%%%%%  OPCOM  24-JAN-1985 17:42:54.36  %%%%%%%%%%
Security alarm on CANADA / Successful file access
     Time:          24-JAN-1985 17:42:54.33
     PID:           21C005B7
     User Name:     PUFFIN
     Image:         (None)
     File:          DUA1:[000000]ROBIN.DIR;1
     Mode:          EXECUTE
     Privs Used:    (None)

%%%%%%%%%%  OPCOM  24-JAN-1985 17:42:56.71  %%%%%%%%%%
Security alarm on CANADA / Successful file access
     Time:          24-JAN-1985 17:42:54.42
     PID:           21C005B7
     User Name:     PUFFIN
     Image:         (None)
     File:          DUA1:[ROBIN]TEST.EXE;1
     Mode:          READ EXECUTE
     Privs Used:    GRPPRV
```

ACCOUNTING UTILITY

The accounting log file (SYS$MANAGER:ACCOUNTNG.DAT) has records that keep track of system usage. The VAX ACCOUNTING UTILITY will process records in the file and will produce formatted reports on those records.

TABLE 8-2: USING ACCOUNTING FOR SECURITY MONITORING

- $ SET ACCOUNTING/ENABLE=(KEYWORD) - CONTROLS RECORDS WRITTEN
- /ENABLE = (LOGIN_FAILURE, BATCH, DETACHED, NETWORK, INTERACTIVE, SUBPROCESS, PRINT, PROCESS, MESSAGE)
- ALL EXCEPT IMAGE ACCOUNTING ENABLED BY DEFAULT
- MAY ENABLE SELECTIVE IMAGE ACCOUNTING INSTALL ADD CDU /ACCOUNTING
- $ SHOW ACCOUNTING - SHOWS WHAT IS ENABLED
- $ SET ACCOUNTING/NEW_FILE - OPENS A NEW FILE
- READ FILE WITH ACCOUNTING UTILITY
- $ ACCOUNTING/SINCE=YESTERDAY WILL GIVE DAILY REPORT CHECK FOR :
  - UNFAMILIAR USERNAMES
  - UNFAMILIAR PATTERNS OF USE
  - ABNORMAL USE OF RESOURCES
  - UNFAMILIAR SOURCES OF LOGINS
  - EXCESSIVE LOGIN FAILURES

EXAMPLE 8-10: $ ACCOUNTING /SINCE=YESTERDAY

DATE / TIME	TYPE	SUBTYPE	USERNAME	ID	SOURCE	STATUS
24-JAN-1985 15:09:58	PROCESS	INTERACTIVE	INVENTORY	21C006B4	VTA570	10000001
24-JAN-1985 15:27:26	LOGFAIL		SYSTEM	21C00639	DUCK	10D3803A
24-JAN-1985 15:32:39	LOGFAIL		<LOGIN>	21C0063D	VTA576	00D3803A
24-JAN-1985 15:37:39	LOGFAIL		<LOGIN>	21C0063E	VTA577	10D38064
24-JAN-1985 15:38:53	PROCESS	INTERACTIVE	SARA	21C00540	DUCK	10000000
24-JAN-1985 15:40:16	PROCESS	NETWORK	FINCH	21C006C3	DUCK	100028A4
24-JAN-1985 15:41:38	LOGFAIL		DEMO	21C00645	VTA580	10D3803A
24-JAN-1985 15:41:58	LOGFAIL		SYSTEM	21C00646	VTA581	10D3803A
24-JAN-1985 15:44:55	PROCESS	NETWORK	DECNET	21C005C1	DUCK	10000004
24-JAN-1985 15:45:46	LOGFAIL		FINCH	21C00648	VTA583	10D3803A
24-JAN-1985 15:46:08	LOGFAIL		<LOGIN>	21C00549	VTA584	10D3803A
24-JAN-1985 15:47:24	LOGFAIL		<LOGIN>	21C004CD	VTA588	10D3803A
24-JAN-1985 15:47:47	PROCESS	INTERACTIVE	FINCH	21C0054E	VTA589	1000000C
24-JAN-1985 15:48:22	LOGFAIL		FINCH	21C006CF	VTA590	10D3803A

EXAMPLE 8-11: $ ACCOUNTING /SINCE=YESTERDAY /TYPE=LOGFAIL /SUMMARY=HOUR

FROM: 24-JAN-1985 00:00 TO: 24-JAN-1985 17:58

HH	TOTAL RECORDS
09	14
10	12
11	5
12	5
13	1
15	15
17	3

EXAMPLE 8-12: $ ACCOUNTING /SINCE=15 /TYPE=LOGFAIL

DATE / TIME	TYPE	USERNAME	ID	SOURCE	STATUS
24-JAN-1985 15:27:26	LOGFAIL	SYSTEM	21C00639	LAKE	10D3803A
24-JAN-1985 15:32:39	LOGFAIL	<LOGIN>	21C0063D	VTA576	00D3803A
24-JAN-1985 15:37:39	LOGFAIL	<LOGIN>	21C0063E	VTA577	10D38064
24-JAN-1985 15:41:38	LOGFAIL	DEMO	21C00645	VTA580	10D3803A
24-JAN-1985 15:41:58	LOGFAIL	SYSTEM	21C00646	VTA581	10D3803A
24-JAN-1985 15:45:46	LOGFAIL	FINCH	21C00648	VTA583	10D3803A
24-JAN-1985 15:46:08	LOGFAIL	<LOGIN>	21C00549	VTA584	10D3803A
24-JAN-1985 15:46:25	LOGFAIL	<LOGIN>	21C0054A	VTA585	10D3803A
24-JAN-1985 15:46:49	LOGFAIL	<LOGIN>	21C005CB	VTA586	10D3803A
24-JAN-1985 15:47:10	LOGFAIL	<LOGIN>	21C006CC	VTA587	10D3803A
24-JAN-1985 15:47:24	LOGFAIL	<LOGIN>	21C004CD	VTA588	10D3803A
24-JAN-1985 15:48:22	LOGFAIL	FINCH	21C006CF	VTA590	10D3803A
24-JAN-1985 15:50:03	LOGFAIL	FINCH	21C00684	LAKE	10D3803A
24-JAN-1985 15:50:35	LOGFAIL	<LOGIN>	21C0050B	VTA593	00D3803A
24-JAN-1985 15:51:01	LOGFAIL	<LOGIN>	21C0048D	VTA595	10D38064

EXAMPLE 8-13: $ ACCOUNTING /SINCE=15 /TYPE=LOGFAIL /FULL

LOGIN FAILURE

```
Username:           SYSTEM          UIC:            [VMS,SYSTEM]
Account:            <LOGIN>         Finish time:    24-JAN-1985 15:41:58.28
Process ID:         21C00646        Start time:     24-JAN-1985 15:41:43.71
Owner ID:                           Elapsed time:              0 00:00:14.57
Terminal name:      VTA581:         Processor time:            0 00:00:01.95
Remote node addr:                   Priority:                  4
Remote node name:                   Privilege <31-00>: FFFFFFFF
Remote ID:                          Privilege <63-32>: FFFFFFFF
Queue entry:                        Final status code: 10D3803A
Queue name:
Job name:
Final status text: %LOGIN-E-NOTVALID, user authorization failure

Page faults:                  165   Direct IO:                  21
Page fault reads:               2   Buffered IO:                41
Peak working set:             188   Volumes mounted:            0
Peak page file:               411   Images executed:            1
```

AUDITING WORLD READABLE FILES

To prevent leakage of proprietary information, it is desirable to have a weekly check on files that are world readable on your system.

- A PRIME CANDIDATE FOR USER PROBING
- MAY PROVIDE A WAY INTO YOUR SYSTEM i.e. list of users, accounts, addresses
- A FILE MIGHT BECOME WORLD READABLE BECAUSE
  - A USER ACCIDENTALLY SET THE DEFAULT PROTECTION TO W:R
  - A USER, FOR COMMUNICATION PURPOSE, SET A FILE'S PROTECTION TO W:R
  - AN APPLICATION PROGRAM CHANGED THE PROTECTION CODE FOR A FILE DURING ITS FILE PROCESSING
  - THE PROTECTION CODE WAS NOT RESET TO W:NOACCESS
- CAN MONITOR WORLD READABLE FILE WITH REPORT_WORLD_V4.COM
  - PRODUCES A LIST OF ALL WORLD READABLE FILES ON A DEVICE
  - ALSO CHECKS FOR WRITEABLE FILES IN SYSTEM DIRECTORIES
  - REQUIRES CMKRNL OR SETPRV PRIVILEGE
  - PARAMETERS: P1 = Device to check
  - MAY BE SUBMITTED AS A BATCH JOB
  - PRODUCES FILE WORLDnn.LIS AND SENDS MAIL MESSAGE TO SYSTEM
  - SEE APPENDIX A

EXAMPLE 8-14: SAMPLE MAIL MESSAGE FROM REPORTWORLDV4.COM

```
From:   SYSTEM                  28-MAR-1985 15:06
To:     SYSTEM
Subj:   List of World Readable Files on GOOSE$DJA0

GOOSE$DJA0:[CLARK.BAS]KITS.COM;2
GOOSE$DJA0:[LEWIS]MY_MAPS.DATA;6
        . . .
```

8.3 AUDITING CHECKLIST

- KNOW YOUR SYSTEM
- OBSERVE THE ROUTINE OPERATIONS
- LOOK AT THE ACCOUNTING FILE ON A REGULAR BASIS
- USE DCL COMMANDS TO OBSERVE YOUR SYSTEM
- LOOK FOR UNEXPLAINABLE DIFFERENCES FROM NORMAL
- KNOW YOUR USERS / EDUCATE YOUR USERS
- KEEP AUDITS RECORDS ON ITEMS YOU WILL LOOK AT
- LOOK AT THE AUDIT RECORDS YOU HAVE KEPT
- DON'T AUDIT EVERYTHING
- AUTOMATE WHAT YOU CAN
- SELECTIVELY EXAMINE REPORTS IN MORE DETAIL WHEN YOU SUSPECT A PROBLEM
- KEEP OLD ACCOUNTS AROUND AND SET THEM /NOACCESS
- KEEP DEMO, USER, GUEST ACCOUNTS SET /NOACCESS

CHAPTER 9
DATA ENCRYPTION FACILITIES

LIST OF TOPICS

- VAX-11 Data Encryption Services
- Key Creation, Deletion, and Management
  - Encryption Algorithms
- Data Encryption Program Interface
- DCL File Encryption Commands
- Data Encryption in BACKUP

9.1 VAX-11 DATA ENCRYPTION SERVICES

VAX-11 Data Encryption Services provide a software implementation of data encryption, for use from programs, with BACKUP, or for files through a DCL command interface.

Presently the only algorithm implemented in the encryption services is the National Bureau Of Standards Data Encryption Standard. This is a publicly known and proven encryption algorithm based on the use of a key. The Data Encryption Services provide base level encryption services without the addition of any additional hardware. They also allow for future expansion to more algorithms and the use of hardware encryption devices.

NOTE

By Department of Commerce ruling, the export of any implementation of the NBS Data Encryption Standard must be by permit only. It is an explicit condition of the license from Digital Equipment Corporation for this software that the licensee agrees not to export nor cause to be exported any portion of this software or related materials thereof from the United States without obtaining the requisite export permit from the Department of Commerce.

9.2 KEY CREATION, DELETION AND MANAGEMENT

The management of the security of the key(s) used is essential, since the algorithm is public. The keys must be kept private and passed by a means separate from the file, from the encryptor to the decryptor. Only knowledge of the key will allow an encrypted file to be decrypted.

Keys may be created once during a session and used after that. They may also be named.

Table 9-1 discusses key creation, deletion, and management.

TABLE 9-1: KEY CREATION, DELETION, AND MANAGEMENT

- DEFINE KEY WITH DCL OR LIBRARY ROUTINE
- MUST KEEP KEY SECRET
- PASSED FROM SOURCE TO DESTINATION APART FROM FILE
- KEYS NAMES CAN BE 1-64 CHARACTERS
- KEY VALUES CAN BE 8 TO 255 ALPHANUMERIC CHARACTERS
- MAY BE ASCII CHARACTERS (DEFAULT) OR BINARY VALUES
- DES ALGORITHM ALWAYS USES 8 BYTE KEY
- KEYS KEPT IN KEY STORAGE TABLES IN MEMORY (PROCESS, GROUP, OR SYSTEM)
- $ ENCRYPT/CREATE_KEY/QUALIFIERS KEY-NAME - KEY-VALUE-STRING

  - /PROCESS, /GROUP, /SYSTEM SPECIFIES KEY STORAGE TABLE

```
$ ENCRYPT/CREATE_KEY
_Key Name: MY_KEY_0
_Key Value: "THIS IS MY KEY001234567889!"

$ ENCRYPT/CREATE_KEY/GROUP  OUR_KEY_1  KEY234_FOR_GROUP890
```

- $ ENCRYPT/REMOVE_KEY/QUALIFIERS KEY-NAME

9.3 PROGRAM INTERFACE TO VAX-11 DATA ENCRYPTION SERVICES FACILITY

- IMPLEMENTED IN 3 PARTS AS A SET OF INTERRELATED PROCEDURES TO PERFORM VARIOUS STEPS OF THE ENCRYPTION PROCESS

- HIGH-LEVEL LANGUAGE INTERFACE
  - FOLLOWS VAX-11 PROCEDURE CALLING STANDARD
  - DESIGNED FOR FORTRAN, PASCAL, PLI AND OTHER VAX LANGUAGES
  - ENCRYPT$INIT, ENCRYPT$ENCRYPT, ENCRYPT$DECRYPT, ENCRYPT$FINI

- INTERMEDIATE INTERFACE LEVEL
  - MODELED AFTER SERVICES OF THE VMS EXECUTIVE
  - PROVIDES INITIALIZE, ENCRYPT, DECRYPT, AND FINALIZE ENTRY POINTS
  - HAS PROVISION FOR ASYNCHRONOUS ENCRYPT OPERATIONS
  - ENCRYPT$INIT_ASYN, ENCRYPT$ENCRYPT_ASYN, ENCRYPT$DECRYPT_ASYN, ENCRYPT$FINI_ASYN

NOTE

In the version 1 release of VMS Encryption services, there is no supplied algorithm which operates asynchronously.

- PRIMITIVE LEVEL ROUTINES
  - ACTUAL ENCRYPTION ALGORITHMS
  - SOFTWARE IMPLEMENTATION OF THE NATIONAL BUREAU OF STANDARDS DATA ENCRYPTION STANDARD
  - PRIMITIVE ALGORITHM INTERFACE

Entry Point	Description	Environment
ENCRYPT$$DES	DES Encryption Algorithm	Synchronous
ENCRYPT$$NUL	Null Algorithm for test	not available
ENCRYPT$$RSA	Rivest/Shamir/Adleman PKS	not available
ENCRYPT$$VMP	VMS password one-way encryption	not available

- MISCELLANEOUS SUPPORT ROUTINES

Entry Point	Description	Environment
ENCRYPT$DEFINE_KEY	Define key	Synchronous
ENCRYPT$DELETE_KEY	Remove key definition	Synchronous
ENCRYPT$ENCRYPT_FILE	Perform file encryption	Synchronous
ENCRYPT$STATISTICS	Return stream statistics	Synchronous
ENCRYPT$ENCRYPT_ONE_RECORD	Use temporary stream	Synchronous
ENCRYPT$DECRYPT_ONE_RECORD	Use temporary stream	Synchronous

ENCRYPT$ENCRYPT_FILE

(INPUT-FILE-DESC.WT.DX, OUTPUT-FILE-DESC.WT.DX,
KEY-NAME-DESC.WT.DX, ALGORITHM-DESC.WT.DX,
FILE-FLAGS.RL.R)

FILE-FLAGS

ENCRYPT$K_FILE_ENCRYPT	SET TO CAUSE FILE ENCRYPTION, CLEAR FOR FILE DECRYPT OPERATIONS
ENCRYPT$K_FILE_DELETE	SET TO CAUSE DELETION OF THE INPUT FILE AFTER THE OPERATION IS COMPLETE.
ENCRYPT$K_FILE_ERASE	SET TO CAUSE THE FILE TO BE ERASED TO THE SECURITY DATA PATTERN BEFORE DELETION.

ENCRYPT$DEFINE_KEY (KEY-NAME.RT.DX, KEY-VALUE.RT.DX,
KEY-FLAGS.RL.R)

ENCRYPT$DELETE_KEY (KEY-NAME.RT.DX, KEY-FLAGS.RL.R)

KEY-FLAGS

ENCRYPT$KEY_PROCESS	PLACE DEFINITION IN PROCESS TABLE
ENCRYPT$KEY_GROUP	PLACE DEFINITION IN GROUP TABLE
ENCRYPT$KEY_SYSTEM	PLACE DEFINITION IN SYSTEM TABLE
ENCRYPT$KEY_LITERAL	DON'T COMPRESS KEY BEFORE STORING
ENCRYPT$KEY_PARITY	FORCE ODD PARITY ON EACH BYTE OF KEY

ENCRYPTION ALGORITHMS AVAILABLE

DEFINED ALGORITHM CODES

ENCRYPT$K_ALG_DES	NBS DATA ENCRYPTION STANDARD
ENCRYPT$K_ALG_NUL	NULL ALGORITHM (NOT IMPLEMENTED)

SUB-MODE CODES DES ALGORITHM

ENCRYPTK_DESECB	ELECTRONIC CODE BOOK
ENCRYPTK_DESCBC	CIPHER BLOCK CHAINING
ENCRYPTK_DESCFB	CIPHER FEEDBACK

DEFINED GLOBAL VALUES FOR A MODE AND SUBMODE

ENCRYPT$K_ALGORITHM_DESECB	NBS DES IN CODE BOOK MODE
ENCRYPT$K_ALGORITHM_DESCFB	NBS DES IN CIPHER FEEDBACK MODE
ENCRYPT$K_ALGORITHM_DESCBC	NBS DES IN CIPHER BLOCK CHAIN MODE
ENCRYPT$K_ALGORITHM_DESDAC	NBS DES IN AUTHENTICATION MODE

ALGORITHM CODES FOR DCL AND BACKUP /ALGORITHM=(CODE)

DES	SAME AS DESCBC
DESCBC	ENCRYPTS IN CIPHER BLOCK CHAINING MODE
DESCFB	ENCRYPTS IN 8 BIT CIPHER FEEDBACK MODE
DESECB	ENCRYPTS IN ELECTRONIC CODEBOOK MODE
DESDAC	COMPUTES A MESSAGE AUTHENTICATION CODE

9.4 DCL FILE ENCRYPTION COMMANDS

The ENCRYPT and DECRYPT commands permit the encryption and decryption of single files.

EXAMPLE 9-1: FORMAT OF ENCRYPT/DECRYPT DCL COMMANDS

```
ENCRYPT[/QUALIFIERS] [INPUT-FILE] [KEY-NAME]

DECRYPT[/QUALIFIERS] [INPUT-FILE] [KEY-NAME]
```

COMMAND QUALIFIERS	DEFAULTS
/ALGORITHM	/ALGORITHM=DESCBC
/BEFORE	/BEFORE=TODAY
/BYOWNER	/NOBYOWNER
/CONFIRM	/NOCONFIRM
/DELETE	/NODELETE
/ERASE	/ERASE
/EXCLUDE	/NOEXCLUDE
/LOG	/NOLOG
/OUTPUT	/OUTPUT
/SINCE	/SINCE=TODAY
/STATISTICS	/NOSTATISTICS

PROMPTS

```
_INPUT FILE(S):
_KEY NAME:
```

EXAMPLE 9-2: USING ENCRYPT/DECRYPT DCL COMMAND

```
$ DIRECTORY/SIZE FILE1.TXT

FILE1.TXT;1                10

$ TYPE FILE1.TXT


    Don't force it - use a hammer

    . . .


$ ENCRYPT/STATISTICS/LOG   -
_Input File(s): FILE1.TXT
_Key Name: MY_KEY_0

%ENCRYP-S-ENCRYPTED, WORK1:[SONG_BIRDS]FILE1.TXT;1
encrypted to WORK1:[SONG_BIRDS]FILE1.TXT; (10 blocks)
%ENCRYP-S-STATISTICS, Encryption Stream Statistics:
Total Records: 3
Total Bytes: 5336
Total Time: 00:00:01.66


$ TYPE FILE1.TXT

YK<o'><0'><u"><XB4>1wQ<U^><12><+-><CSI><LF><a"><2^>^WI<OE>
<DCS><MW>6J<e"><>>>J<SSA><XAE><o^><C/><SPA>f<XDE><c,><STS>
:<RI>-<X80><P!><X98>^Bv<NEL><0'><3^>t^N<SS2><I'><EPA><U'>
<A^><0^>X_^Z<a'>$I^T<VTS><e^>Z<A'><!!>x#<SS@><RI><o'><e"><X0>


$ DECRYPT   FILE1.TXT  MY_KEY_0


$ DIRECTORY/SIZE FILE1.TXT

FILE1.TXT;3                10
FILE1.TXT;2                11
FILE1.TXT;1                10
```

9.5 DATA ENCRYPTION IN BACKUP

Data encryption is available with BACKUP at any part of the command line. The /ENCRYPT qualifier must be put on the backup command. A key value may be entered and verified or a named key may be used

Using /ENCRYPT with BACKUP will either indicate that the created save set is to be encrypted or the input save set is to be decrypted.

The algorithm used with BACKUP uses a randomizes the key based on the time of day, so that two save sets created with the same information at different times will not necessarily be the same in their encrypted form. They can, however be decrypted with the use of the same key.

The following examples demonstrate the use of the data encryption services with BACKUP.

EXAMPLE 9-3: SAVING AND RESTORING WITH THE SAME KEY

```
$ ENCRYPT/CREATE MYKEY_28JULY "I like sugar cookies"

$ BACKUP/ENCRYPT=(name=mykey_28july) * TAPE:28JULSAVE/SAVE


$ ENCRYPT/CREATE mykey "i like sugar cookies"

$ BACKUP/ENCRYPT=(name=MYKEY)
$_From: TAPE:28JULSAVE.BCK/SELECT=SALARY.DAT
$_To: SALARY28J.DAT
```

EXAMPLE 9-4: USING ENCRYPTION WITH BACKUP

```
$ BACKUP/ENCRYPT  FILE*.TXT TEXT_FILES.BCK/SAVE
Enter key value:
Verification:


$ BACKUP/LIST  TEXT_FILES.BCK/SAVE
Listing of save set

%BACKUP-F-ENCSAVSET, save set is encrypted,
/ENCRYPT must be specified



$ BACKUP/ENCRYPT/LIST  TEXT_FILES.BCK/SAVE
Enter key value:
Verification:
Listing of save set

Save set:            TEXT_FILES.BCK
Written by:          THRUSH
UIC:                 [100,025]
Date:                28-JAN-1985 13:20:46.55
Command:             BACKUP/ENCRYPT FILE*.TXT TEXT_FILES.BCK/SAVE
Operating system:    VAX/VMS version V4.0
BACKUP version:      V4.0
CPU ID register:     013680BD
Node name:           _DUCK::
Written on:          _GOOSE$DUA1:
Block size:          32256
Group size:          10
Buffer count:        3
Saveset Encrypted

[SONG_BIRDS]NEWFILE.TXT;4          2  28-JAN-1985 13:07
[SONG_BIRDS]FILE1.TXT;3            1  28-JAN-1985 13:05
[SONG_BIRDS]FILE1.TXT;2            2  28-JAN-1985 13:19
[SONG_BIRDS]FILE1.TXT;1            1  28-JAN-1985 13:05

Total of 4 files, 6 blocks
End of save set
```

CHAPTER 10
NETWORK SECURITY

LIST OF TOPICS

- DECnet Overview
- DECnet TASKS
- DECnet and the Security Reference Monitor
- Authentication and Access Control across DECnet
  - Making a Logical Link Connection
  - Specifying Access Control
- PROXY Logins
  - Overview of PROXY
  - Using PROXY
  - Setting Up PROXY Access
- DECnet Management Security Considerations
  - Other Nodes on the Network
  - Controlling Default Accounts
  - Protection of the DECnet VAX Database
  - DECnet Security and Passwords
  - FAL and NML Security Concerns
  - Poor Man's Routing
- Review of Communication Security

10.1 DECNET OVERVIEW

DECnet is the collective name for the software and hardware products that are a means for various DIGITAL operating systems to participate in a network. DECnet-VAX is the implementation of DECnet that causes a VAX/VMS operating system to function as a network node. As the VAX/VMS network interface, DECnet-VAX supports both the protocols necessary for communicating over the network and the functions necessary for configuring, controlling, and monitoring the network.

Phase IV DECnet is the current version of DECnet, and it supports the configuration of very large, as well as small, networks.

Table 10-1 lists some characteristics of DECnet-VAX. Table 10-2 lists the DECnet physical links. Figure 10-1 shows a sample DECnet configuration and

Table 10-3 lists and defines basic terms used with DECnet.

TABLE 10-1: OVERVIEW OF DECNET

- SUPPORTS COMMUNICATION WITH VMS, OTHER DIGITAL OPERATING SYSTEMS, AND NONDIGITAL OPERATION SYSTEMS IN THE NETWORK

- IS DECENTRALIZED
  .I.E. MANY NODES CAN COMMUNICATE WITH EACH OTHER WITHOUT HAVING TO GO THROUGH A CENTRAL NODE

- THERE NEED BE NO HIERARCHY
  .I.E. ONE NODE NEED NOT BE DEPENDENT ON THE EXISTENCE OF ANOTHER

- DIFFERENT APPLICATIONS RUNNING ON SEPARATE NODES CAN SHARE THE FACILITIES OF ANY OTHER NODE

- OPTIONALLY, CAN BE DIVIDED INTO MULTIPLE AREAS, FOR THE PURPOSE OF HIERARCHICAL (AREA) ROUTING

- EACH NODE IN A MULTIPLE-AREA NETWORK CAN STILL COMMUNICATE WITH ALL OTHER NODES IN THE NETWORK

- WITHOUT MULTIPLE AREAS, SUPPORTS UP TO 1023 NODES

  OPTIMUM NUMBER APPROXIMATELY 200 TO 300 NODES

- AREA ROUTING TECHNIQUES PERMIT CONFIGURATION OF VERY LARGE NETWORKS CONSISTING OF UP TO 63 AREAS, EACH CONTAINING A UP TO 1023 NODES

DECNET CONFIGURATIONS

TABLE 10-2: DECNET PHYSICAL LINKS

- ETHERNET CIRCUIT IN A LAN CONFIGURATION
- SYNCHRONOUS AND ASYNCHRONOUS CONNECTIONS USING DIGITAL DATA COMMUNICATIONS MESSAGE PROTOCOL (DDCMP)
  - DMR-11, DMC-11, DMP-11, DMV-11
  - DZ-11, DMF-32, DMZ-11
- CONNECTION OVER A COMPUTER INTERCONNECT (CI)
  - STAR COUPLER
  - CI-780, CI-750
- X.25 PACKET SWITCHING DATA NETWORK (PSDN)

FIGURE 10-1: SAMPLE DECNET NETWORK

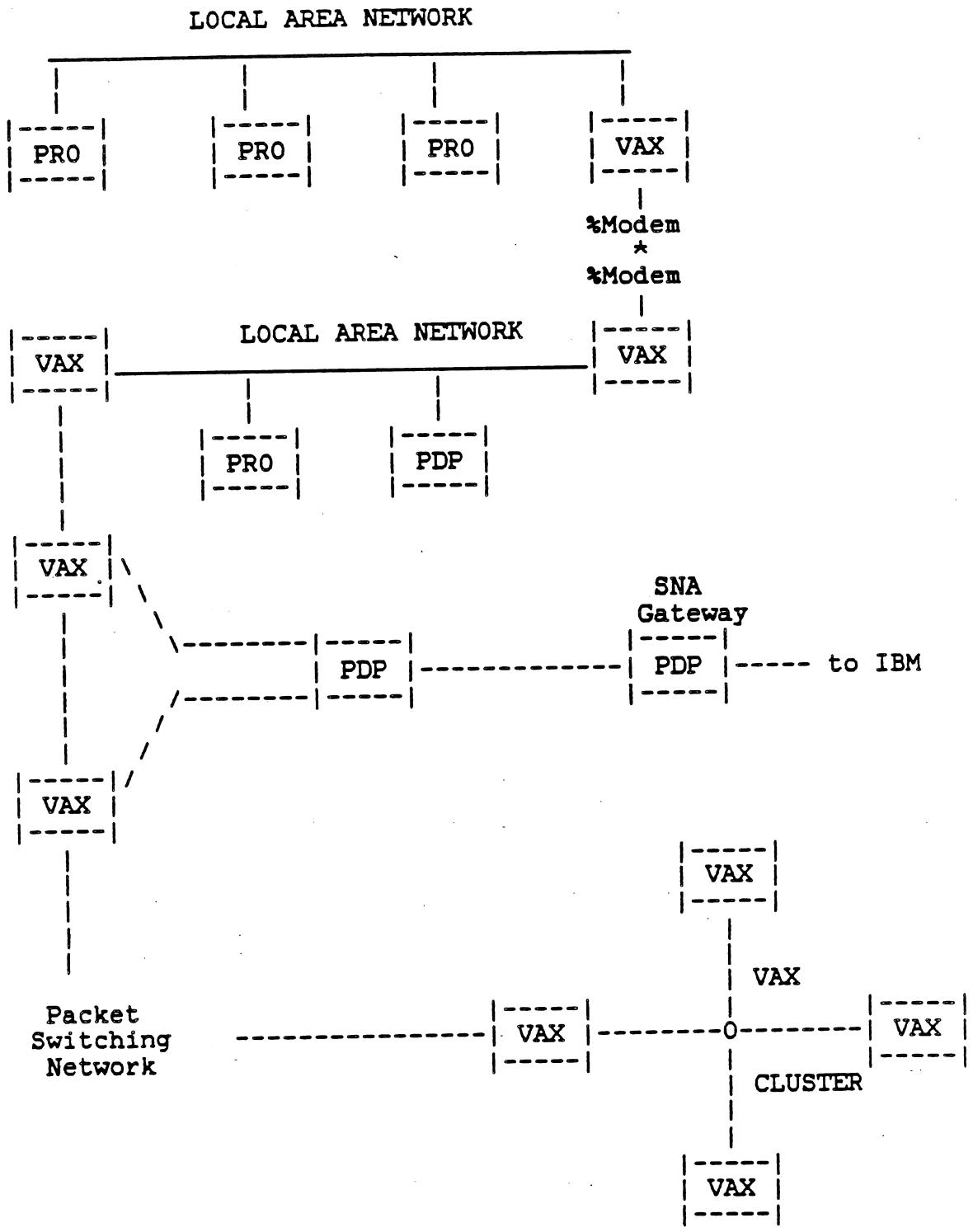
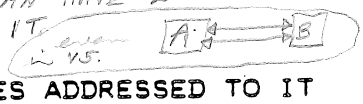


TABLE 10-3: DECNET TERMS

- DATA NETWORK : SYSTEM VIA WHICH COMPUTER PROCESSES COMMUNICATE WITH ONE ANOTHER
- NODE : COMPUTER SYSTEMS ON THE NETWORK
- LOGICAL LINK : A CONNECTION, AT THE USER LEVEL, BETWEEN TWO PROCESSES
- ADJACENT NODES : NODES CONNECTED BY A SINGLE PHYSICAL LINE
- CIRCUIT : COMMUNICATIONS DATA PATH OVER WHICH ALL INPUT AND OUTPUT (I/O) BETWEEN NODES TAKES PLACE
  MAY SUPPORT MANY CONCURRENT LOGICAL LINKS
- PACKET SWITCHING DATA NETWORK (PSDN) : SET OF EQUIPMENT AND INTERCONNECTING LINKS THAT PROVIDES A PACKET-SWITCHING COMMUNICATIONS SERVICE
- VIRTUAL CIRCUITS : LOGICAL ASSOCIATIONS BETWEEN NODES FOR THE EXCHANGE OF DATA
- ROUTING : THE PROCESS OF DIRECTING A DATA MESSAGE FROM A SOURCE TO A DESTINATION NODE THROUGH AN INTERMEDIATE NODE
- ADAPTIVE ROUTING : ROUTING PROCEDURE WHICH PERMITS MESSAGES TO BE ROUTED THROUGH THE NETWORK OVER THE MOST COST EFFECTIVE PATH
- ROUTING NODE : CAN SEND AND RECEIVE MESSAGES AND CAN FORWARD OR ROUTE MESSAGES FROM ONE NODE TO ANOTHER – CAN HAVE 2 OR MORE LINKS TO IT

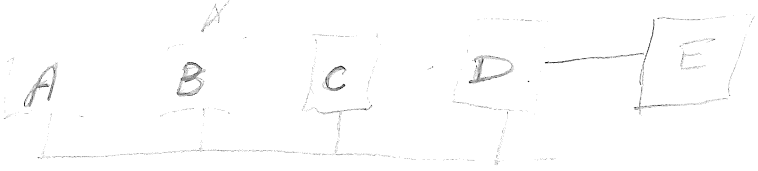


- NONROUTING (END) NODE : CAN ONLY SEND MESSAGES TO OTHER NODES AND RECEIVE MESSAGES ADDRESSED TO IT FROM OTHER NODES, BUT CANNOT ROUTE MESSAGES THROUGH TO OTHER NODES – HAS ONLY ONE LINK.
- AREAS : GROUPS OF NODES THAT CAN OPERATE INDEPENDENTLY AS A SUBNETWORK

V5.

PATH SPLIT Equal cost links. uses both.

Designated Router - has Node Data Base.



V5 has Reverse Path Caching.

10.2 DECNET TASKS

Most tasks on DECnet VAX are command procedures that will be executed in the context of the remote process. These tasks may be executed as objects or not. The main difference is how DECnet finds the command procedure to execute. The command procedure will typically request that a particular program be executed to perform the network function.

10.2.1 DECNET OBJECTS

Objects provide known general-purpose network services. These objects may be used by any user software that requests them. An object is identified by an object type number. DIGITAL supplies several network objects by default:

File Access Listener FAL, Network Management Listener (NML), Event Logger (EVL), MAIL, and PHONE.

There also is a default object TASK with a 0 object type. This is discussed in section 10.2.2.

All objects must be defined in the configuration database. The command procedures for these objects typically reside in SYS$SYSTEM: although they can be anywhere that is specified in the database.

EXAMPLE 10-1: SETTING UP A USER-DEFINED OBJECT

```
NCP> DEFINE OBJECT XYZ NUMBER 200 FILE  SYS$SYSTEM:XYZ

     SYS$SYSTEM:XYZ.COM RUNS TARGET IMAGE
```

EXAMPLE 10-2: USING A USER-DEFINED OBJECT

```
OPEN (....,NAME='VAX750::"XYZ="'....)
                'VAX750::"200="'
                'VAX750"NET PWD"::"XYZ="'
                'VAX750"NET PWD"::"200="'
```

TABLE 10-4: REMOTE FILE ACCESS COMMANDS

- MOST FILE-RELATED DCL COMMANDS WILL TAKE A REMOTE FILE SPECIFICATION AS A PARAMETER
  - $ COPY
  - $ DELETE
  - $ PURGE
  - $ DIRECTORY
  - $ TYPE
  - $ SUBMIT/REMOTE
  - $ PRINT/REMOTE

- WILL INITIATE A REMOTE FILE ACCESS
- USES THE FILE ACCESS LISTENER (FAL)
- DCL INTERFACE FOR REMOTE FILE ACCESS
- FAL WILL USUALLY BE A KNOWN NETWORK OBJECT
- WILL RUN FAL.COM WHICH RUNS FAL.EXE

10.2.2 DEFAULT TASK OBJECT

There can also be tasks executed via DECnet that are not explicitly defined in the configuration database. These execute under the default TASK OBJECT. The default task object has a 0 object type and is used to execute command procedures on behalf of the user.

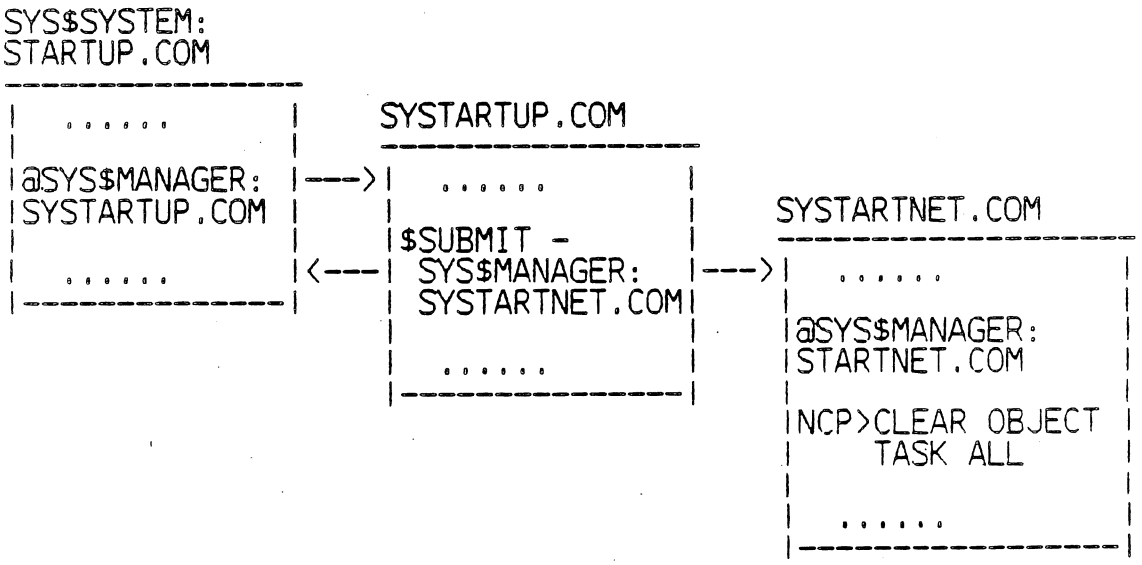
- "TASK" OBJECT IS CREATED DURING DECNET STARTUP
- MAY USE IT WITH OR WITHOUT EXPLICIT ACCESS CONTROL INFORMATION
- TASK= AND 0= ARE EQUIVALENT

```
$ TYPE CANADA"GRAY_GOOSE XYZ"::TASK=FINDSEED
$ TYPE CANADA::"0=FINDSEED"
```

- REMOTE NODE CHECKS ACCESS CONTROL INFORMATION
- REMOTE COMMAND PROCEDURE MUST EXIST IN SYS$LOGIN:
- ESTABLISHES LOGICAL LINK IF ACCESS IF PERMITTED
- SECURITY CONCERN WHEN USED WITH DEFAULT DECNET ACCOUNT
- ALLOWS USERS TO PROBE YOUR SYSTEM VIA DEFAULT ACCOUNT
- MANY BREAKINS UNDER VMS V3.x ARE RELATED TO THE TASK OBJECT
- STRONGLY RECOMMENDED TO ELIMINATE THE DECNET "TASK" OBJECT
- IF A USER HAS THE NEED TO RUN A PROGRAM ON YOUR SYSTEM FROM A REMOTE NODE YOU CAN DEFINE A DECNET OBJECT FOR THIS PURPOSE INSTEAD OF USING THE TASK OBJECT (SEE EXAMPLE 10-1)

- OTHER WAYS OF DEALING WITH THE TASK OBJECT MAY NOT BE AS EFFECTIVE
- NEED TO PUT THE COMMANDS TO ELIMINATE THE TASK OBJECT IN TO YOUR SITE-SPECIFIC NETWORK STARTUP COMMAND PROCEDURE
- CALL THE SITE-SPECIFIC DECnet STARTUP PROCEDURE FROM SYS$MANAGER:SYSTARTUP.COM
- INCLUDE THE FOLLOWING COMMANDS :

```
$ @SYS$MANAGER:STARTNET.COM
$ RUN SYS$SYSTEM:NCP
CLEAR OBJECT TASK ALL
EXIT
```



ELIMINATING THE TASK OBJECT WILL GO A LONG WAY TOWARDS SECURING YOUR NETWORK

10.3 DECNET AND THE SECURITY REFERENCE MONITOR

The security reference monitor discussed in Chapter 2 may be extended to network security. The subjects are on one computer and the objects on another. There is a network reference monitor that controls access of the subjects to the objects. The authorization database consists of the UAF files, the rights databases, ACL's, and the file protection scheme, all on each node.

Because these entities exist on each node, there is a new level of complexity brought in when security is discussed on a network. Figure 10-2 pictures the simple case of a network reference monitor, and Figure 10-3 expands this to a more realistic picture.

FIGURE 10-2: SIMPLE VIEW OF REFERENCE MONITOR IN A NETWORK

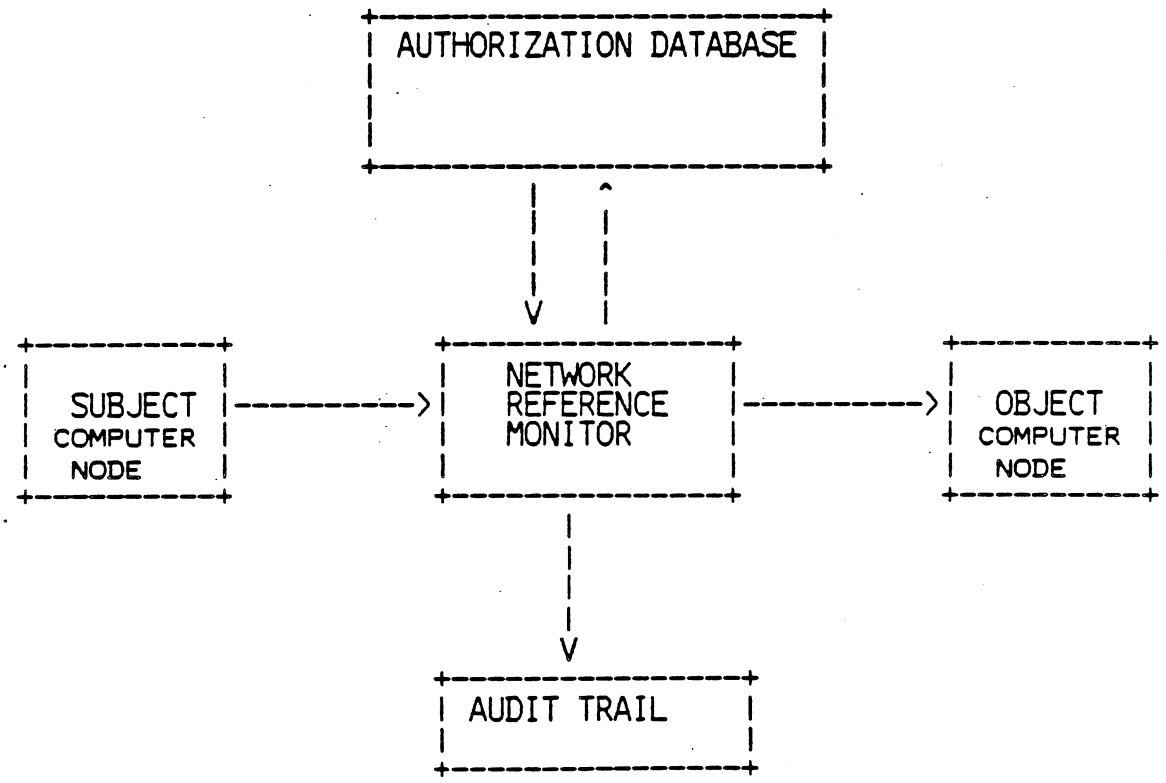


FIGURE 10-3: ADVANCED VIEW OF REFERENCE MONITOR IN A NETWORK

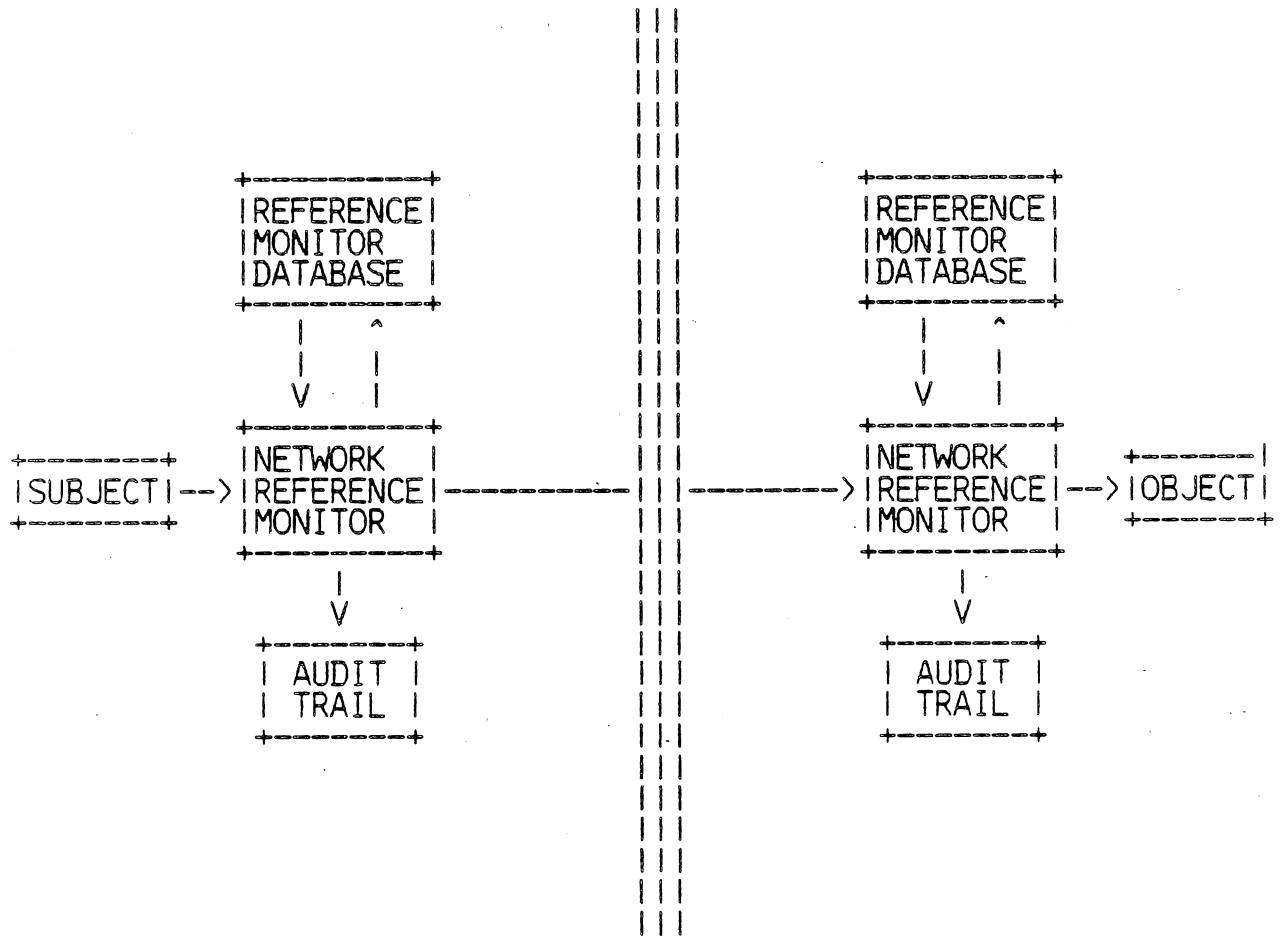


TABLE 10-5: NOTES ON REFERENCE MONITOR IN A NETWORK

- MANY PIECES OF SOFTWARE CONFORM TO FIGURE 10-2
  - SUBJECT IS ON ONE COMPUTER IN NETWORK
  - OBJECT IS ON ANOTHER
  - REFERENCE MONITOR FOR NETWORK CONTROLS ACCESS

- REALITY IS MORE LIKE FIGURE 10-3
  - TWO COMPUTER SYSTEMS
  - ONE HOLDS THE SUBJECT THAT NEEDS TO GAIN ACCESS
  - THE OTHER HOLDS OBJECT THAT IS THE TARGET OF ACCESS ATTEMPT
  - EACH SYSTEM HAS ITS OWN IMPLEMENTATION OF THE REFERENCE MONITOR
  - THEY MUST BE COORDINATED IN ORDER TO OBTAIN SECURITY IN THE NETWORK
  - THERE WILL BE A PHANTOM OBJECT ON THE SYSTEM WITH REAL SUBJECTS (THE SOURCE MACHINE)
  - THERE WILL BE A PHANTOM SUBJECT ON THE SYSTEM WITH REAL OBJECTS (THE TARGET MACHINE)

TABLE 10-6: 3 CRITICAL REQUIREMENTS FOR ACHIEVING SECURITY IN A NETWORK ENVIRONMENT

1. THERE MUST BE A CORRESPONDENCE BETWEEN THE REAL SUBJECT ON THE SOURCE AND THE PHANTOM SUBJECT ON THE TARGET
   - DONE BY AUTHENTICATION AND ACCESS CONTROL

2. THE AUTHORIZATION DATABASE ON THE TARGET MUST EXPRESS AN ACCESS AUTHORIZATION FOR A PHANTOM SUBJECT THAT CORRESPONDS TO THE CORRECT REAL SUBJECT
   - THE IMPLEMENTATION DEPENDS ON THE ACCESS CONTROL METHOD USED

3. THERE MUST BE A PROTECTED MEANS OF COMMUNICATION BETWEEN THE TWO REFERENCE MONITORS (NODES)
   - OUTSIDE CURRENT REALM OF VAX/VMS
   - INVOLVES ENCRYPTION AT LINK LEVEL OR IN HIGHER LEVEL OF NETWORK SOFTWARE
   - OTHER PHYSICAL PROTECTION OF LINES

10.4 AUTHENTICATION AND ACCESS CONTROL ACROSS DECNET

The method of making a correspondence between the real subject on the source and the phantom subject on the target is the authentication of the user. This association will be determined by the access control that a node uses.

Access control is the control that a node exercises over inbound logical link connections. DECnet VAX provides several methods of doing this. The method chosen will help to ensure that an access authorization for a phantom subject corresponds to the correct real subject.

MAKING THE LOGICAL LINK CONNECTION

Table 10-7 overviews how a logical link is made.

Table 10-8 lists and shows how access control is specified.

Figure 10-4 shows the how the outbound connection request on the sources is processed.

Figure 10-5 shows how the inbound connection request is handled on the target.

TABLE 10-7: MAKING THE LOGICAL LINK CONNECTION

- RMS OR APPLICATION PROGRAM FROM LOCAL NODE SENDS REQUEST TO DECNET SOFTWARE
- DECNET SOFTWARE ON LOCAL NODE SENDS REQUEST TO DECNET SOFTWARE ON REMOTE NODE
- ALL NETWORK TASKS RUN WITHIN THE CONTEXT OF A PROCESS
- NETWORK PROCESS MUST (EXIST OR) BE CREATED ON THE REMOTE NODE
- REMOTE NODE NEEDS USERNAME/PASSWORD OR PROXY ACCOUNT TO CREATE NETWORK PROCESS
- ACCESS CONTROL INFORMATION IS USED IN CREATING REMOTE PROCESS TO MAKE ASSOCIATION BETWEEN THE PHANTOM AND REAL SUBJECT
- REMOTE PROCESS RUNS LOGINOUT
  - SYS$OUTPUT = SYS$LOGIN:NETSERVER.LOG
  - LOG FILE IS NOT PRINTED NOR DELETED
  - EXECUTES SYSTEM AND PROCESS LOGIN COMMAND PROCEDURE(S)
  - F$MODE() = "NETWORK"
- REMOTE PROCESS EXECUTES SYS$SYSTEM:NETSERVER.COM
  - RUNS NETSERVER.EXE
  - CHOOSES COMMAND PROCEDURE TO RUN FOR INCOMING REQUEST
  - COMMAND FILE STARTS IMAGE TO IMPLEMENT DECNET WORK
- WHEN COMMAND PROCEDURE FINISHES, PROCESS WILL STAY AROUND FOR NETSERVER$TIMEOUT TIME

TABLE 10-8: ACCESS CONTROL IN REMOTE NODE FILE SPECIFICATION

- USER CAN SPECIFY EXPLICIT ACCESS CONTROL INFORMATION IN REMOTE NODE SPECIFICATION

  ```
  $ COPY  CANADA"GRAY_GOOSE XYZ"::SEEDS.DATA  *.*

  $ COPY  CANADA"GRAY_GOOSE XYZ"::WORK2:[GEESE]SEEDS.DATA  *.*
  ```

- DIFFICULT TO AUDIT - WHO KNOWS USERNAMES/PASSWORDS
- DOES NOT SUPPORT SECONDARY PASSWORDS
- USER CAN USE DEFAULT ACCESS CONTROL INFORMATION OR PROXY ACCESS
- WILL USE PROXY IF IT'S ENABLED ON BOTH NODES AND USER HAS AN ENTRY IN SYSUAF.DAT

  ```
  $ COPY  CANADA::NEW_SEEDS.DATA  *.*

  $ COPY  CANADA::WORK2:[GEESE]NEW_SEEDS.DATA  *.*
  ```

- CAN FORCE USE OF DEFAULT DECNET ACCOUNT

  ```
  $ COPY  CANADA"":WORK2:[GEESE]NEW_SEEDS.DATA  *.*
  ```

- ACCESS DECISION MADE AT REMOTE NODE BASED ON CHARACTERISTICS OF REMOTE PROCESS

FIGURE 10-4: OUTBOUND CONNECTIONS ACCESS CONTROL (FROM LOCAL NODE)

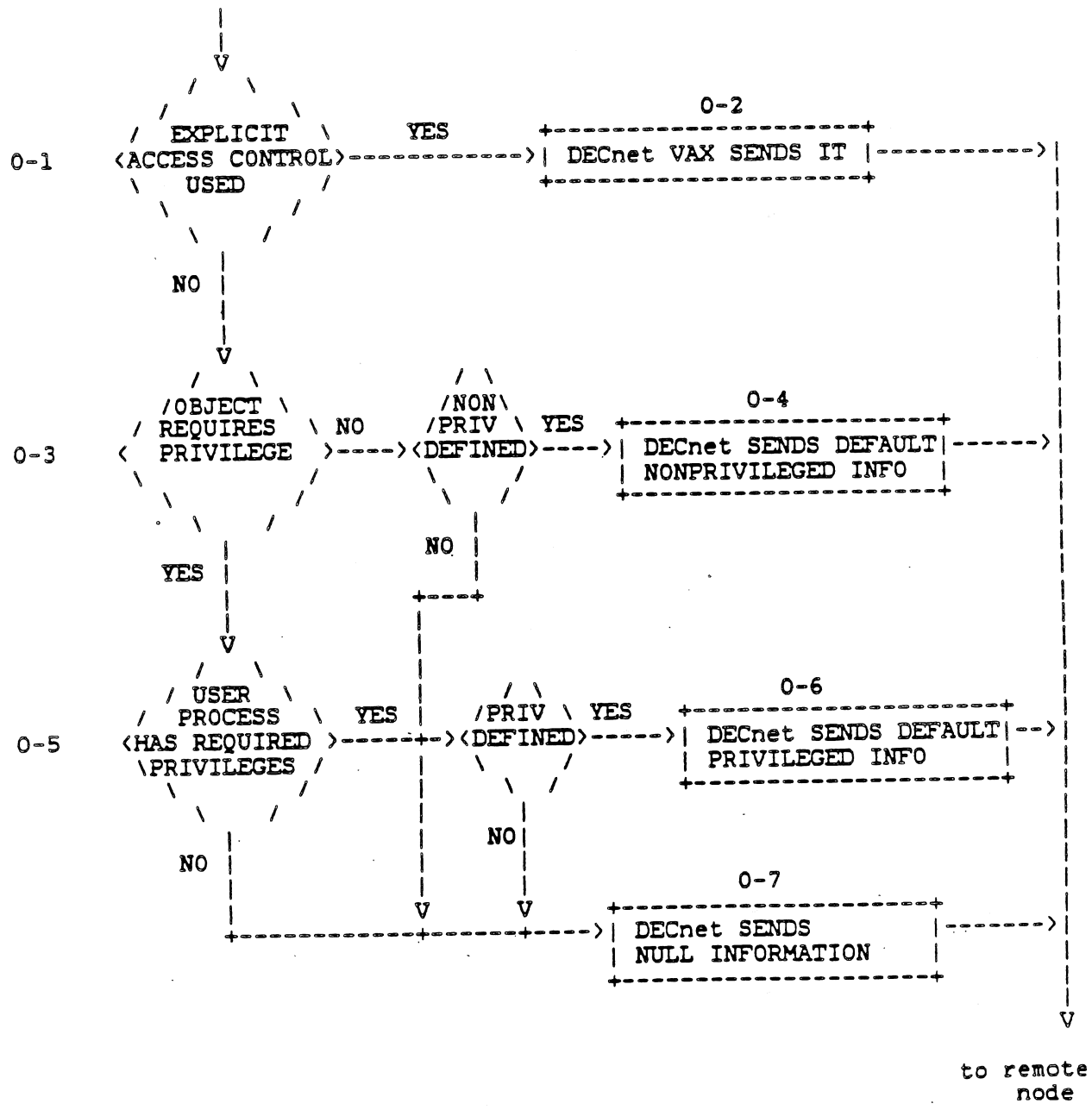
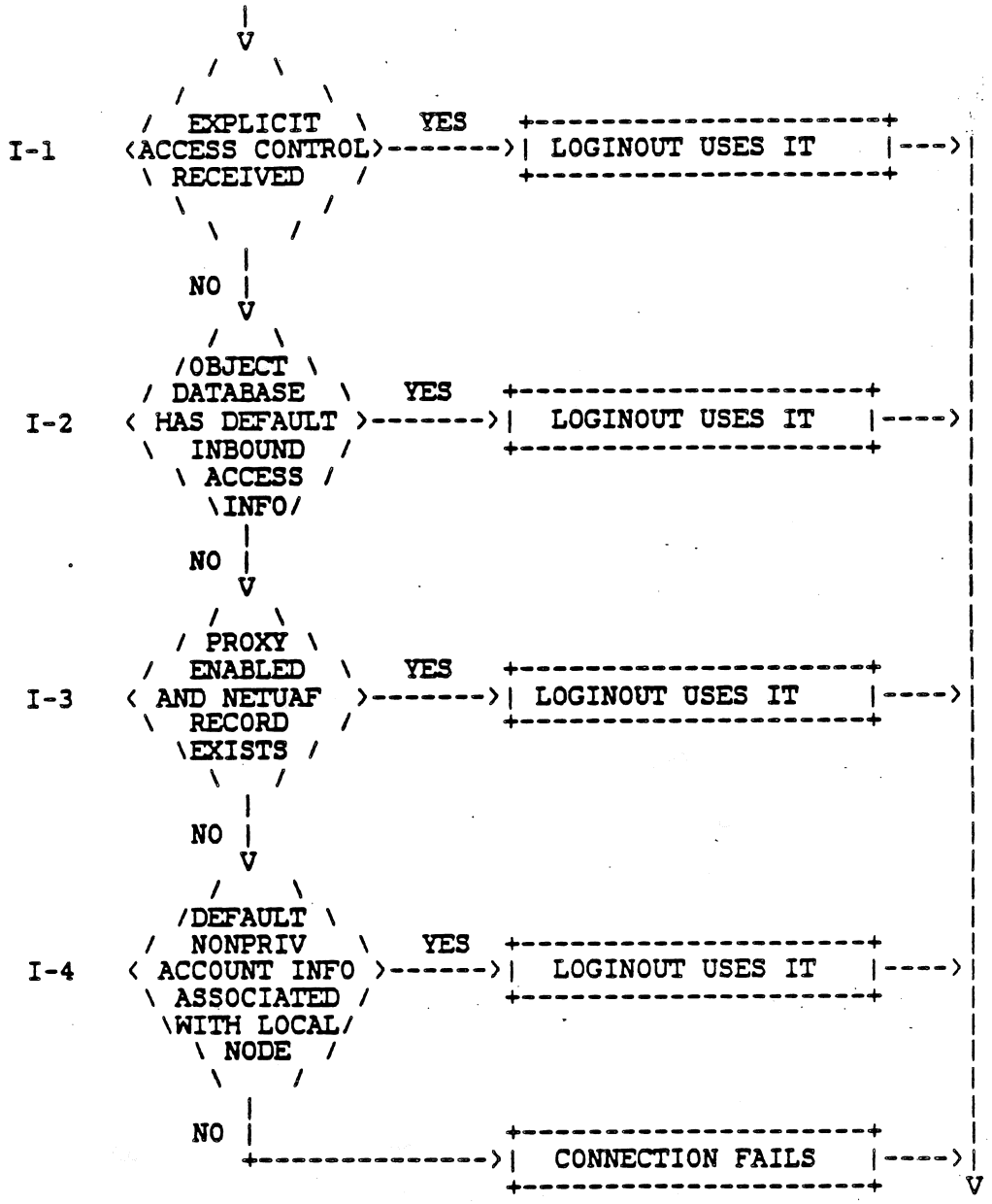


FIGURE 10-5: INCOMING CONNECTIONS ACCESS CONTROL PROCESSING (FROM LOCAL NODE)



EXAMPLE 10-3: ACCESS REQUEST EXAMPLES

local node = (BOSTON)
remote node = (BANGOR)

1. Explicit access control

   O1 ==> I1

2. No explicit access control

   Connection NOT to a network object

   Non-privileged account defined for BANGOR in the BOSTON configuration data base

   NO PROXY enabled

   O1 ==> O3 ==> O4 ==> I1

3. No explicit access control

   Connection NOT to a network object

   OUTGOING PROXY enabled for BOSTON

   INCOMING PROXY enabled for BANGOR

   O1 ==> O3 ==> O5 ==> O7 ==> I1 ==> I2 ==> I3

4. No explicit access control

   Connection to a network object NOT defined with privilege

   Non-privileged account NOT defined for BANGOR in the BOSTON configuration data base

   Network object defined in BANGOR configuration data base with user name and password

   O1 ==> O3 ==> O5 ==> O7 ==> I1 ==> I2

5. No explicit access control

   Connection to a network object NOT defined with privilege

   Network object defined in BANGOR configuration data base WITHOUT user name and password

   Non-privileged account NOT defined for BANGOR in the BOSTON configuration data base

   Non-privileged account defined for EXECUTOR in BANGOR configuration data base

   No proxy record matching BOSTON::USERNAME

   01 ==> 03 ==> 07 ==> I1 ==> I2 ==> I3 ==> I4

6. No explicit access control

   Connection to a network object defined with OPER privilege

   Privileged account defined for BANGOR in BOSTON configuration data base

   Source process has the OPER privilege

   01 ==> 03 ==> 05 ==> 06 ==> I1

7. No explicit access control

   Connection to a network object defined with OPER privilege

   Privileged account NOT defined for BANGOR in BOSTON configuration data base
   OR
   Source process does NOT have the OPER privilege

   Network object NOT defined in BANGOR configuration data base

   NO PROXY record BOSTON::username

   Nonprivileged account NOT defined for BANGOR in BOSTON configuration data base

   01 ==> 03 ==> 05 ==> 07 ==> I1 ==> I2 ==> I3 ==> I4

10.5 PROXY LOGINS

OVERVIEW OF PROXY

When users on different nodes or in different groups want to share files across DECnet you are somewhat limited in how to establish those files. You can give world access to the file and let users come in through the default DECnet account. Or you can give out the username and password of an account that has access to the file. The user can explicitly supply the access control information in the remote file specification. Each of these methods has its advantages and disadvantages. Neither is very satisfactory from a security standpoint. Proxy logins are a form of DECnet access control intended to solve the problems with the above approaches.

Proxy accounts permit one or more user on a remote node to obtain access privileges on a node without necessarily having a private account on that node. Proxy accounts also allow transmission of data without putting explicit access control and therefore passwords in the request. The remote user can access data to which his/her proxy account has access.

One or more remote users can be mapped to a single local proxy account. Or each user can have his/her own proxy account on the remote node. The creation of this mapping is controlled by the system manager on the local node through the use of AUTHORIZE.

Proxy logins are intended to provide all of the advantages of both explicit and default access control while avoiding their drawbacks. Table 10-9 lists some features and advantages of PROXY access.

TABLE 10-9: ADVANTAGES AND FEATURES OF PROXY ACCOUNTS

- PROVIDE MORE SECURE NETWORK ACCESS
  - PASSWORDS ARE NEVER ECHOED AT THE TERMINAL DEVICE WHERE THE REQUEST ORIGINATES
  - PASSWORDS ARE NOT PASSED BETWEEN SYSTEMS WHERE THEY MIGHT BE INTERCEPTED IN UNENCRYPTED FORM
  - REMOVES THE TEMPTATION TO STORE PASSWORDS IN COMMAND FILES THAT WOULD DO THE REMOTE ACCESS
- INVOLVE MUCH SIMPLER PROCEDURES FOR THE USERS
  - REMOTE USERS NEED NOT KNOW THE PASSWORD OF THE LOCAL ACCOUNT
  - CAN STILL USE EXPLICIT ACCESS CONTROL
  - WILL USE DEFAULT DECNET ACCOUNT IF THE USER HAS NO REMOTE PROXY RECORD
- ACCESS IS CONTROLLED BY SYSTEM MANAGER ON LOCAL NODE
  - SYSTEM MANAGER CAN ALLOW OR DENY ACCESS TO SPECIFIC REMOTE USERS
  - REQUIRES MORE SETUP EFFORT ON THE PART OF THE MANAGERS
  - CAN MAP SPECIFIC REMOTE USERS TO SPECIFIC LOCAL ACCOUNT
  - CAN MAP MULTIPLE REMOTE USERS TO ONE LOCAL ACCOUNT
- ALL PROXY LOGINS ARE NETWORK LOGINS AND ARE NONINTERACTIVE

TABLE 10-10: SECURITY CONSIDERATIONS WITH PROXY ACCESS

- A PERSONAL COMPUTER COULD FAKE PROXY LOGIN MECHANISM BY PRETENDING TO BE ONE OF THE AUTHORIZED NODES
- USE INCOMING PROXY SPARINGLY ON SENSITIVE NODES
- DON'T GIVE PROXY ACCOUNTS PRIVILEGES THAT COULD HARM YOUR SYSTEM

EXAMPLE 10-4: COPYING WITH PROXY

```
$ COPY  CANADA::SEED.DATA  NEW_SEED.DATA
```

- USER MUST HAVE LOCAL ACCOUNT AND REMOTE PROXY ACCOUNTS
- PASSWORD IS NOT DISPLAYED
- NETWORK DOES THE INTERNAL VERIFICATION THAT THE INCOMING USER IS NEST::GRAY_GOOSE
- A NETWORK LOGIN IS PERFORMED ON THE NODE CANADA FOR MATCHING PROXY ACCOUNT
- FILE IS ACCESSED IN DEFAULT DIRECTORY OF PROXY ACCOUNT ON NODE CANADA

TABLE 10-11: HOW PROXY FUNCTIONS ON THE RECEIVING NODE

- IF NO ACCESS CONTROL STRING IS SUPPLIED
  (A NULL STRING WAS SENT)
- AND PROXY ACCESS IS ENABLED AT THE PARTICIPATING NODES
- LOGINOUT CHECKS NETUAF.DAT TO SEE IF SENDING USER HAS A PROXY ACCOUNT
- IF THEY DO - A PROXY LOGIN IS PERFORMED
- IF NOT - THE DEFAULT DECNET ACCOUNT IS USED

TABLE 10-12: CONTROLLING PROXY LOGINS WITH NCP

- PROXY ACCESS IS CONTROLLED WITH NCP
- EXECUTOR DATABASE HAS A DEFAULT PROXY PARAMETER
  (USED TO SUPPLY VALUE TO LOCAL NODES AND OTHER NODES NOT EXPLICITLY SET UP IN CONFIGURATION DATABASE)
- OBJECT DATABASE HAS AN OBJECT PARAMETER TO CONTROL PROXY ACCESS TO SPECIFIC OBJECTS
- 4 POSSIBLE VALUES FOR PROXY PARAMETERS
  - INCOMING - ALLOW PROXY LOGINS/ACCESS FROM REMOTE NODES
  - OUTGOING - ALLOW PROXY LOGINS/ACCESS REQUESTS TO BE INITIATED FROM THE LOCAL NODE
  - BOTH - ALLOW BOTH INCOMING AND OUTGOING PROXY REQUESTS
    (THE DEFAULT SETTING)
  - NONE - ALLOW NEITHER INCOMING OR OUTGOING PROXY LOGIN ACCESS CONNECTIONS
- CANNOT SET EXECUTOR DATABASE PROXY PARAMETER TO OUTGOING OR BOTH IF THERE ALSO IS A DEFAULT OUTGOING ACCOUNT

  (CONFLICTS ON RECEIVING NODE - CAN ONLY GET NULL STRING OR ACCESS CONTROL INFO)
- PROXY IS IGNORED BY NON-VMS NODES

NOTE

There are advantages to disallowing incoming proxy access to an object that does not need it (i.e. MAIL). Logical link connections are more likely to reoccur to the default account. This will somewhat reduce network overhead.

Outgoing proxy access should be enabled, as it send the username and node name with outbound request instead of the node and PID. This will be displayed with the NCP SHOW KNOWN LINKS command.

EXAMPLE 10-5: SHOW KNOWN LINKS COMMAND

```
NCP> SHOW KNOWN LINKS CHAR

Known Link Volatile Characteristics as of 23-JAN-1985 20:56:54

Link = 40

State                   = run
PID                     = 21A00C3A
Remote node             =  1.4 (LAKES)
Delay time              = 3
Remote link             = 45
Remote user             = PERCH
Username                = FISH
Process name            = FAL_40


NCP> SHOW KNOWN LINKS SUMMARY

Known Link Volatile Summary as of 23-JAN-1985 20:56:55

Link    Node          PID       Process   Remote link   Remote user

40   1.4 (LAKE)   21A00C3A  FAL_40            45    PERCH
```

EXAMPLE 10-6: SETTING DEFAULT PROXY IN THE VOLATILE DATABASE

```
NCP>CLEAR EXECUTOR PRIVILEGED USER PASSWORD ACCOUNT

NCP>SET EXECUTOR DEFAULT PROXY BOTH
                                  INCOMING PROXY ENABLE   V5
                                  OUTGOING

NCP>SHOW EXECUTOR CHAR
Node Volatile Characteristics as of 22-JAN-1985 20:13:27

Executor node = 1.1 (CANADA)

Identification              = DECnet-VAX V4.0,  VMS V4.0
Management version          = V4.0.0
Nonprivileged user id       = DECNET
Default access              = incoming and outgoing
Default proxy access        = incoming and outgoing
```

EXAMPLE 10-7: DISALLOWING PROXY ACCESS ON AN OBJECT

```
NCP>SET OBJECT MAIL PROXY NONE

NCP>SHOW OBJECT MAIL CHARACTERISTICS
Object Volatile Characteristics as of 22-JAN-1985 20:14:11

Object = MAIL

Number                      = 27
File id                     = MAIL.EXE
Proxy access                = none
```

TABLE 10-13: ENABLING PROXY WITH AUTHORIZE

- NETWORK AUTHORIZATION FILE (NETUAF.DAT) MUST BE CREATED
- EACH RECORD MAPS A REMOTE USER TO A LOCAL PROXY ACCOUNT

  REMOTE_NODE::REMOTE_USERNAME LOCAL_USERNAME
- CAN HAVE MULTIPLE REMOTE USERS MAP TO ONE PROXY ACCOUNT
  - CAN ESTABLISH GENERAL ACCESS ACCOUNT THAT PERMITS ONLY NETWORK LOGINS
  - NONE OF THE USERS NEEDS TO KNOW THE PASSWORD
- EACH REMOTE USER CAN MAP TO AT MOST ONE PROXY ACCOUNT

V4
V5 ... restriction

- UAF PROXY COMMANDS
  - UAF>CREATE/PROXY
  - UAF>ADD/PROXY NODE::REMOTEUSER LOCALUSER
  - UAF>ADD/PROXY NODE::* LOCALUSER
  - UAF>ADD/PROXY NODE::* *
  - UAF>LIST/PROXY
  - UAF>SHOW/PROXY NODE::REMOTEUSER
  - UAFSHOW/PROXY *
  - UAF>SHOW/PROXY NODE::*
  - UAF>SHOW/PROXY *::REMOTEUSER
  - UAF>REMOVE/PROXY NODE::REMOTEUSER
- IF LOCAL USERS ARE CHANGED (I.E. RENAMED, REMOVED) CORRESPONDING PROXY RECORDS ARE UPDATED

EXAMPLE 10-8: USING AUTHORIZE TO ESTABLISH A NETUAF.DAT FILE

```
UAF> SHOW /PROXY *
%UAF-W-NAFDNE, NETUAF.DAT DOES NOT EXIST

UAF> CREATE/PROXY

UAF> ADD/PROXY CANADA::WREN  WREN
%UAF-I-NAFADDMSG, RECORD SUCCESSFULLY ADDED TO NETUAF.DAT

UAF> ADD/PROXY CANADA::ROBIN ROBIN

UAF> ADD/PROXY LAKES::ROBIN ROBIN

UAF> ADD/PROXY LAKES::FINCH FINCH

UAF> ADD/PROXY LAKES::FINCH PERCH
%UAF-W-BADUSR, USERNAME DOES NOT EXIST \PERCH\

UAF> ADD/PROXY LAKES::FINCH  WARBLER
%UAF-W-NAFUAEERR, ENTRY ALREADY EXISTS IN NETUAF.DAT

UAF> SHOW/PROXY  CANADA::ROBIN
 NODE    REMOTE USER       LOCAL USER
CANADA::ROBIN              ROBIN

UAF> SHOW/PROXY  LAKES::*
 NODE    REMOTE USER       LOCAL USER
LAKES ::FINCH              FINCH
LAKES ::ROBIN              ROBIN

UAF> SHOW/PROXY *
 NODE    REMOTE USER       LOCAL USER
CANADA::ROBIN              ROBIN
CANADA::WREN               WREN
LAKES ::FINCH              FINCH
LAKES ::ROBIN              ROBIN

UAF> REMOVE /PROXY LAKES::FINCH
%UAF-I-PREMMSG, RECORD REMOVED FROM NETUAF.DAT
```

EXAMPLE 10-9: CREATING A GENERAL ACCESS PROXY ACCOUNT

```
$ DIRECTORY/SECURITY   WORK2:[SEED_DATA]NEW_SEEDS
Directory WORK2:[SEED_DATA]

NEW_SEEDS.FORMULA;1
          [GRAINS,WRITE_DATA]          (RWED,RWED,R,)
    (IDENTIFIER=ENGINEERING,ACCESS=WRITE)
    (IDENTIFIER=SONG_BIRDS,ACCESS=READ)

UAF> ADD READ_DATA /DIRECTORY=[SEED_DATA] /DEVICE=WORK2:     -
     /NOACCESS /PASSWORD=XYZ123 /ACCOUNT=GRAINS/UIC=[GRAINS,2]

UAF> MODIFY READ_DATA/NETWORK/FLAGS=(CAPTIVE,DISMAIL,DISCTLY)


UAF> ADD/PROXY  CANADA::SPARROW READ_DATA

UAF> ADD/PROXY  CANADA::ORIOLE  READ_DATA


UAF> ADD/PROXY  LAKES::PIKE  READ_DATA

UAF> ADD/PROXY  LAKES::PERCH READ_DATA

UAF> ADD/PROXY  LAKES::TROUT READ_DATA


UAF> SHOW/PROXY  *::READ_DATA
 Node    Remote User      Local User

CANADA::ORIOLE            READ_DATA
CANADA::SPARROW           READ_DATA
LAKES ::PERCH             READ_DATA
LAKES ::PIKE              READ_DATA
LAKES ::TROUT             READ_DATA
```

NOTES ON EXAMPLE 10-9

1. HAVE A FILE THAT NEEDS MULTIPLE READERS FROM DIVERSE NODES
2. DIRECTORY AND FILE ARE OWNED BY [GRAINS,WRITE_DATA]
3. PROTECTED TO ALLOW GROUP READ BUT ONLY SYSTEM OR OWNER CONTROL
4. ACL ALLOWS SPECIFIC READ AND WRITE
5. ESTABLISH GENERAL ACCOUNT READ_DATA
   (IN THE SAME GROUP BUT NOT THE OWNER I.E. CAN READ)
6. ADD PROXY RECORDS
7. CAN MODIFY ACL FOR MORE LOCAL READERS/WRITERS
8. CAN MODIFY PROXY RECORDS FOR CHANGES IN REMOTE ACCESS
9. CAN PROPAGATE ACL TO ALL FILES IN DIRECTORY

EXAMPLE 10-10: ESTABLISHING SINGLE ACCOUNT ACCESS FOR ALL USERS FROM A NODE

```
UAF>ADD/PROXY  CARIBU::*     READ_DATA

UAF>ADD/PROXY  CARIBU::DEMO  DECNET

UAF>ADD/PROXY  CARIBU::DECNET  DECNET
```

NOTES ON EXAMPLE 10-10

1. CAN ALLOW ALL USERS FROM A NODE TO USE ONE PROXY ACCOUNT
2. KEEPS ONE PROXY RECORD
3. CAN OVERRIDE IT WITH ALTERNATE ENTRIES FOR SPECIFIC REMOTE USERS

10.6 DECNET MANAGEMENT SECURITY CONSIDERATIONS

OTHER NODES

Generally, the network is no more secure than its least secure node. Other nodes present a special security consideration. NON-VMS nodes present a special concern, as they do not have the same security feature that are available in VMS. Other VMS nodes may not be as well managed and may be a concern as well.

TABLE 10-14: OTHER DECNET NODES

- NON-VMS NODES MAY NOT HAVE ENCRYPTED PASSWORDS
- OTHER VMS NODES MAY NOT HAVE GOOD PASSWORD MANAGEMENT
- DISCOURAGE THE USE OF THE SAME PASSWORD ACROSS SYSTEMS (PARTICULARLY TO NON-VMS SYSTEMS)
- DISCOURAGE USE OF PASSWORDS IN COMMAND PROCEDURES

```
$ SET HOST CANADA
GRAY_GOOSE
XYZ

$ ASSIGN "CANADA""GRAY_GOOSE XYZ"":: "    CDA
```

CONTROLLING DEFAULT ACCOUNTS

A DECnet default account is needed to receive mail from other nodes, to have your node function as a pass-through router, and to distribute software or other files to the network from your node. However, a DECnet default account makes your system wide open to the outside world. Some guidelines to follow in managing your DECnet default account are summarized in Table 10-15.

TABLE 10-15: GUIDELINES FOR MANAGING DEFAULT ACCOUNTS

- IT IS DIFFICULT TO TRACK USAGE OF DEFAULT ACCOUNT
- DON'T SET UP DEFAULT PRIVILEGED ACCOUNT
- DON'T GIVE THE DEFAULT NONPRIVILEGED ACCOUNT PRIVILEGES BEYOND TMPMBX AND NETMBX
- DEFAULT ACCOUNTS SHOULD BE IN THEIR OWN NON-SYSTEM UIC GROUP
- DON'T USE OBVIOUS PASSWORDS - KEEP THEM SECRET
- SENSITIVE FILES AND DIRECTORIES SHOULD BE PROTECTED AGAINST WORLD ACCESS
- KEEP DISK QUOTA OF DEFAULT ACCOUNT SMALL
- DON'T PUT LOGIN COMMAND FILE IN THE DEFAULT DIRECTORY
- PERMIT ONLY NETWORK ACCESS IN UAF RECORD

```
UAF>MODIFY DECNET/NETWORK/NOBATCH/NOLOCAL/NOREMOTE/NODIALUP
UAF>MODIFY DECNET/FLAGS=(DISCTLY,CAPTIVE,LOCKPWD,DISMAIL,DEFCLI)
```

PROTECTION OF DECNET VAX DATABASE

DECnet VAX maintains two databases. A volatile database maintained in memory and accessed via NCP (NETWORK CONTROL PROGRAM), and a permanent database maintained in several files in SYS$SYSTEM:.

The files to protect are :

NETCIRC.DAT, NETCONF.DAT, NETLINE.DAT, NETLOGING.DAT, NETNODE.DAT, NETOBJECT.DAT, and NETUAF.DAT.

Once these files are protected to make changes to the DECnet database will require OPER privilege and WRITE access to database files. TO START THE NETWORK YOU MUST HAVE ACNT, CMKRNL, SYSNAM, and DETACH PRIVILEGE.

TABLE 10-16: ROUTING PASSWORDS

- DEFINE TRANSMIT AND RECEIVE PASSWORDS FOR ALL NODES IN THE NETWORK
  - USED FOR ROUTING INITIALIZATION
  - YOU DEFINE THEM FOR OTHER NODES
  - TRANSMIT - ONE YOU SEND TO OTHER NODE
  - RECEIVE - ONE YOU EXPECT TO RECEIVE FROM OTHER NODE
  - IF THEY MATCH CONNECTION IS MADE
  - THEIR USE IS OPTIONAL - BUT A GOOD SECURITY MEASURE
  - ESPECIALLY USEFUL FOR UNPROTECTED LINES
  - PASSWORDS SHOULD BE DIFFERENT, NONOBVIOUS, AND PRIVATE
  - ONLY MANAGER OF ADJACENT NODES NEED TO KNOW YOUR RECEIVE PASSWORD
  - UP TO 8 CHARACTERS
  - ESTABLISH PASSWORDS WITH NCP
- USE ROUTING PASSWORDS AND OTHER VERIFICATION METHODS ON ALL CIRCUITS THAT GO OUTSIDE THE COMPUTER ROOM

EXAMPLE 10-11: ESTABLISHING TRANSMIT AND RECEIVE PASSWORDS FOR THE LOCAL NODE

```
NCP>SET NODE CANADA TRANSMIT PASSWORD "VAX_BIRDS" -
    RECEIVE PASSWORD "WHITE_SNOW"
```

EXAMPLE 10-12: REMOVING ROUTING PASSWORDS FROM THE VOLATILE DATABASE

```
NCP> CLEAR NODE CANADA RECEIVE PASSWORD TRANSMIT PASSWORD
```

NOTE

Use DEFINE and PURGE to make the above changes in the permanent database.

EXAMPLE 10-13: ESTABLISHING A DEFAULT ACCESS RIGHT FOR NODE CANADA

```
NCP> SET NODE CANADA -
     NONPRIVILEGED USER DECNET PASSWORD SCRAMBLEDXYZ
```

FAL AND NML SECURITY CONCERNS

- DON'T SET UP PRIVILEGES FOR NML FROM A DEFAULT ACCOUNT
- SET IT UP AS AN OBJECT OR USE SET HOST

  NCP> SET OBJECT NML PRIVILEGES OPER

- ACCOUNT FOR THE FAL OBJECT SHOULD BE IN ITS OWN GROUP
- MEMBER NUMBER OF FAL UIC SHOULD BE DIFFERENT FROM MEMBER NUMBER OF FAL DIRECTORY
- ENABLE FAL LOGGING BY PLACING IN DECNET LOGIN COMMAND PROCEDURE : $ DEFINE FAL$LOG "01/DISABLE=8"
- ALLOWS THE SYSTEM MANAGER TO SEE WHAT FILES ARE BEING ACCESSED VIA DECNET
- CAUSES EACH NETSERVER.LOG FILE TO INCLUDE A LIST OF THE FILES BEING ACCESSED
- /DISABLE=8 HELPS DISCOVER INTRUDER USING A NODE IN THE POOR MAN'S ROUTING SCHEME
- CAN KEEP CERTAIN ACCOUNTS FROM DOING ANY REMOTE FILE ACCESS

  1. GUARANTEE THAT ALL REMOTE FILE ACCESS REQUESTS WILL RUN FAL.COM

     NCP> DEFINE OBJECT FAL FILE FAL.COM

  2. HAVE RESTRICTED USERS LOGGED OFF WHEN THEY ATTEMPT A REMOTE FILE ACCESS BY PLACING THE FOLLOWING LINE IN THEIR LOGIN FILE

     $ FAL$COMMAND :== LOGOUT

EXAMPLE 10-14: SAMPLE DECNET LOGIN COMMAND FILE

```
$
$ set noon
$ set prot=(o,g,w) *.*;
$ set prot=(o,g,w) *.*;-1
$ define fal$log "01/disable=8"
$ if f$mode() .eqs. "NETWORK" then exit 1
$
$ IF "''NETSERVER$COMMAND'" .NES. "" THEN NETSERVER$COMMAND
$ IF "''NETSERVER$VERIFY'"  .EQS. "" THEN NETSERVER$VERIFY = 0
$ V = F$VERIFY(NETSERVER$VERIFY)
```

EXAMPLE 10-15: OUTPUT FROM NETSERVER.LOG

```
------------------------------------------------------------

    Connect request received at 31-DEC-1985 07:42:27.87
    from remote process RDGMRC::"0=DECNETXFR    "
    for object "SYS$SYSROOT:[SYSEXE]FAL.COM"

------------------------------------------------------------

============================================================
FAL V04-000 started execution on 31-DEC-1985 07:42:28.26
with SYS$NET = RDGMRC::"0=DECNETXFR    " and
with FAL$LOG = 01/disable=8
Requested file access operation: Directory List
Specified file: [SECURITY]*.*;*
DAP status code of 4055 generated

FAL terminated execution on      31-DEC-1985 07:42:34.35
============================================================

DECNET       job terminated at 31-DEC-1985 07:47:35.65
Accounting information:
Buffered I/O count:       127     Peak working set size:    253
Direct I/O count:         103     Peak page file size:      675
Page faults:             1031     Mounted volumes:            0
Charged CPU time: 0 00:00:05.26   Elapsed time:  0 00:05:14.06
```

NOTE

The DAP status code above simply means that the file access failed. If the file access succeeds NETSERVER.LOG will give a list of file names that were accessed.

POOR MAN'S ROUTING (PMR)

- ALLOWS A USER TO SPECIFY A PATH OF NODES TO A TARGET SYSTEM ON THE NETWORK
- WAS NECESSARY IN DECNET PHASE III (MAXIMUM NUMBER OF DIRECTLY ACCESSIBLE NODES WAS 255)

EXAMPLE 10-16: USING PMR'S

```
$ SET HOST A::B::C
$ DIRECTORY A::B::C::SYS$SYSDEVICE:[DECNET]
$ MAIL FILENAME A::B::C::SMITH
```

- THE SYSTEM AUDIT MECHANISM WILL ONLY RECORD THE LAST NODE NAME USED AS A PASS THROUGH NODE
- OFTEN IS DECNET (NAME OF THE DECNET DEFAULT ACCOUNT)
- INTRUDER CAN HIDE HIS/HER REAL IDENTITY
  `$ COPY YODA::OMEGA::VORTEX::SYS$SYSTEM:SYSUAF.DAT *`
- ACCOUNTING LOG WILL RECORD ON VORTEX A NETWORK EVENT

  FROM REMOTE NODE : OMEGA

  WITH A REMOTE ID : DECNET

TABLE 10-17: STOPPING THE USE OF THE PMR SCHEME IN FAL

- CHECK IF THE PMR SCHEME IS BEING USED TO LOGIN
  - CHECK FOR THE REMOTE USERNAME = DECNET
  - IF IT IS NOT DECNET, THEN EXIT.
  - IF IT IS DECNET THEN LOGOUT

EXAMPLE 10-17: STOPING THE USE OF PMR IN FAL

```
$ ! FAL_PMR.COM          By Henry Teng   Jan 7, 1985
$ ! This command procedure is to detect the use of
$ ! poor man's routing in FAL. The procedure is called by
$ ! SYS$SYSTEM:FAL.COM when there is a FAL request.
$ ! The default login command procedure for DECNET
$ ! must have the following command:
$ !          $ FAL$COMMAND :== @SYS$MANAGER:FAL_PMR.COM
$
$ Set noon
$ Net_user = f$logical("SYS$NET")
$ If f$locate("DECNET", net_user).ne.f$length(net_user) then logout
$ Exit
```

10.7 REVIEW OF COMMUNICATION SECURITY

AREAS OF CONCERN

The goals of a network security policy should be to permit control over network specific resources, and to support mechanisms that enforce policy in other components of the system. This must be done in a manner that is consistent with the desired access control policies.

DECnet provides some methods of helping to achieve these goals, but there still are areas where the reference monitor can be bypassed or subverted. For example, passwords cannot be considered a secure form of authentication in an architecture where all messages may be routed in plain text through potentially insecure lines.

TABLE 10-18: SUMMARY OF SECURE NETWORK MANAGEMENT PRACTICES

- DON'T HAVE A DEFAULT PRIVILEGED ACCOUNT
- DON'T SET UP DEFAULT OUTBOUND ACCESS CONTROL INFORMATION
- KEEP UICS OF DEFAULT ACCOUNTS UNIQUE
- ENCOURAGE THE USE OF PROXY OVER EXPLICIT ACCESS CONTROL
- BEWARE : PROXY ACCESS IS ONLY AS SECURE AS THE SECURITY OF THE LEAST SECURE NODE
- DON'T GIVE PRIVILEGES TO PROXY ACCOUNTS
- ELIMINATE THE DEFAULT TASK OBJECT

CHAPTER 11
VAXCLUSTER SECURITY

LIST OF TOPICS

- Introduction To VAXclusters
  - VAXcluster Overview
  - VAXcluster Hardware
  - VAXcluster Software
- Security Reference Monitor on a VAXcluster
- VAXcluster Operating Environments
  - Homogeneous VAXcluster Security Considerations
  - Heterogeneous VAXcluster Security Considerations

ch. 10 - 217

11.1 INTRODUCTION TO VAXCLUSTERS

The VAXcluster is a highly integrated organization of VAX/VMS systems that communicate over a high-speed communications path. Like a single-node VAX/VMS system, the VAXcluster organization provides a single security and management domain.

Depending on how the cluster is configured, there may be some special security concerns on a cluster.

Table 11-1 overviews some basic cluster concepts. Table 11-2 lists and describes the hardware components of a VAXcluster and Table 11-3 describes the software components.

TABLE 11-1: VAXCLUSTER OVERVIEW

- USE AN EXTENDED VERSION OF THE VMS OPERATING SYSTEM
- HAVE ALL THE FUNCTIONS OF SINGLE NODE VAX/VMS SYSTEMS
- ALSO HAVE THE ABILITY TO SHARE CPU RESOURCES, QUEUES, AND DISK STORAGE
- FALL SOMEWHERE BETWEEN TIGHTLY COUPLED MULTIPROCESSOR SYSTEMS (11/782's) AND LOOSELY COUPLED NETWORKS (DECNET)
  - BOOT/FAIL SEPARATELY
  - SINGLE SECURITY/MANAGEMENT DOMAIN
  - INTEGRATED FILE SYSTEM
  - SINGLE COMPUTER ROOM (1)
  - GREAT GROWTH POTENTIAL

TABLE 11-2: VAXCLUSTER HARDWARE COMPONENTS

- VAX PROCESSOR
  - 11/780, 11/782, 11/785, 11/750 , 8600 RUNNING THE VAX/VMS OPERATING SYSTEM
  - CLUSTER SUPPORTS UP TO 16 ACTIVE NODES (CPU'S) in CI cluster.
- COMPUTER INTERCONNECT (CI)
  - HIGH-SPEED, DUAL-PATH, BUS THAT CONNECTS VAX'S AND HSC50'S IN A COMPUTER ROOM
  - CAN BE ONLY ONE CI ON EACH VAXCLUSTER
  - CONNECTED TO PROCESSOR WITH MICROCODED, INTELLIGENT CONTROLLER (CI780, CI750)
- STAR COUPLER
  - COMMON CONNECTION POINT FOR ALL NODES CONNECTED TO THE CI
  - CREATES A RADIAL OR "STAR" WITH A MAXIMUM RADIUS OF 45 METERS
- HIERARCHICAL STORAGE CONTROLLER (HSC50) (HSC70)
  - A SELF-CONTAINED, INTELLIGENT, MASS STORAGE SUBSYSTEM
  - SUPPORTS DIGITAL STANDARD ARCHITECTURE (DSA) DISKS (RA series)
  - CLUSTER SUPPORTS UP TO 15 PASSIVE NODES (HSC50s) (70s)
- LOCAL AREA CLUSTER

  Ether Net

TABLE 11-3: VAXCLUSTER SOFTWARE COMPONENTS

- SYSTEM COMMUNICATION SERVICES (SCS)
  - SOFTWARE THAT IMPLEMENTS INTERNODE COMMUNICATION
  - FOLLOWS DIGITAL'S SYSTEM COMMUNICATION ARCHITECTURE
- CONNECTION MANAGER *CNXMAN process*
  - SOFTWARE THAT DYNAMICALLY DEFINES AND COORDINATES THE CLUSTER
  - USES THE SYSTEM COMMUNICATION SERVICES
- DISTRIBUTED FILE SYSTEM
  - ALLOWS ALL VAX PROCESSORS TO SHARE ANY DISK
  - ALL CLUSTER-AVAILABLE DISKS APPEAR AS IF THEY ARE LOCAL TO EVERY VAX PROCESSOR
- MASS STORAGE CONTROL PROTOCOL (MSCP) SERVER
  - IMPLEMENTS THE MSCP PROTOCOL (PROTOCOL USED BY HSC DISKS) ON LOCAL DISKS
  - MAKES LOCALLY CONNECTED DISKS AVAILABLE TO ALL NODES IN THE CLUSTER
- DISTRIBUTED LOCK MANAGER
  - IMPLEMENTS THE $ENQ AND $DEQ SERVICES TO PROVIDE CLUSTER-WIDE SYNCHRONIZATION OF ACCESS TO RESOURCES
  - USED BY THE FILE SYSTEM, RMS, JOB CONTROLLER, OTHER CLUSTER FACILITIES AND BY USER APPLICATIONS
- DISTRIBUTED JOB CONTROLLER
  - PERMITS A CLUSTER-WIDE SET OF QUEUES

11.2 SECURITY REFERENCE MONITOR ON A VAXCLUSTER

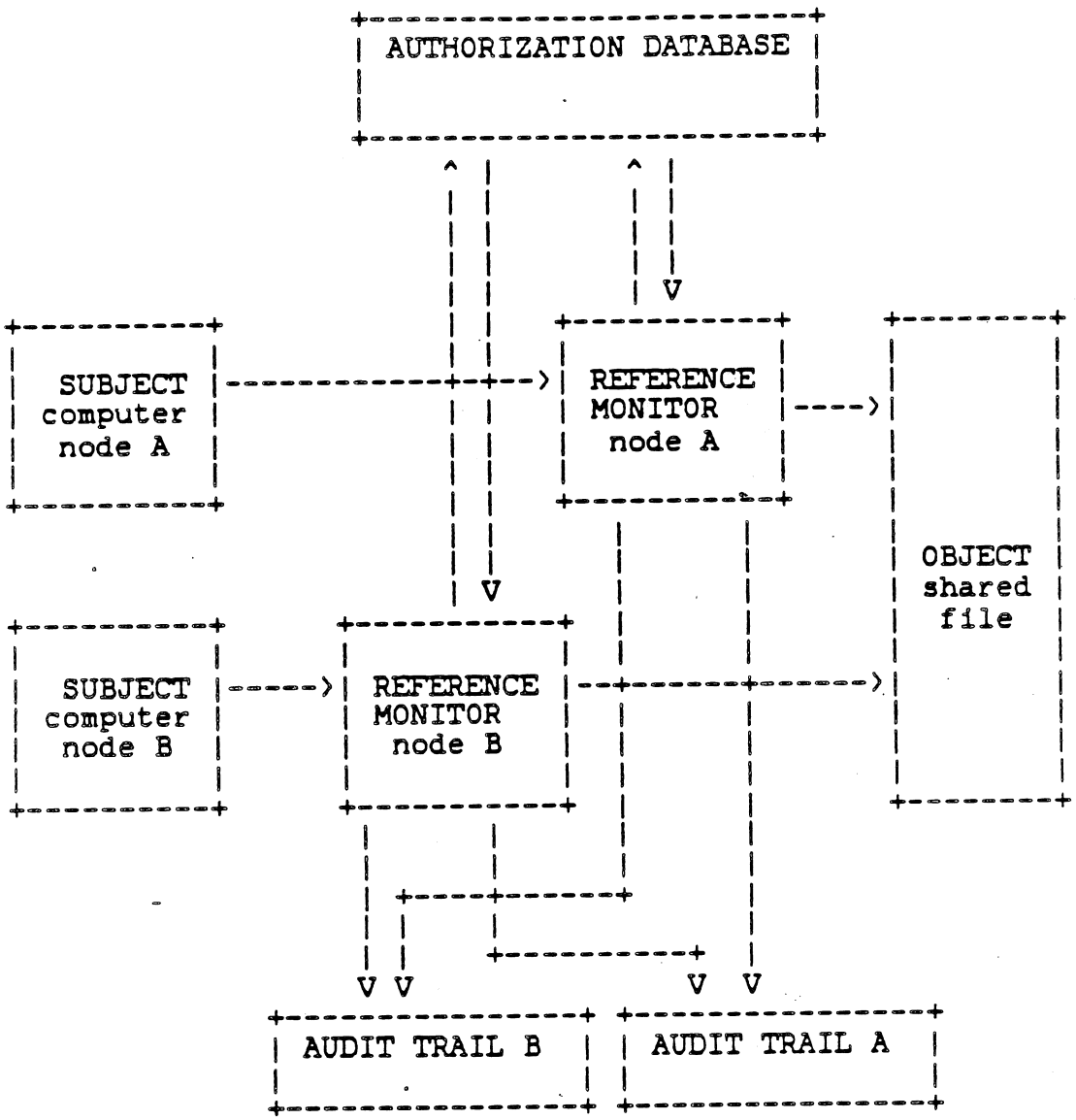
The security reference monitor discussed in chapter 2 may be applied to VAXcluster security. The subjects are on one or more computer and the objects are often shared. There is a reference monitor on each node that controls access of the subjects to the objects. The authorization database files may be shared or separate files. On a cluster, the interesting aspect of file protection is for the shared files. If the files are on a non-served, local disk then the reference monitor is a single system model.

For shared files the concern comes in when there may be more than one authorization database (multiple UAF, NETUAF, and RIGHTSLIST) files.

Figure 11-1 pictures the case of a reference monitor on a VAXcluster with one authorization database. You need to make sure that the reference monitor functions as it is pictured below. The authorization database must be coordinated, whether the files reside in shared or separate files.

Note that audit trails are both shared and separate. Security alarms are sent cluster-wide, accounting information, and DCL commands are node specific.

FIGURE 11-1: VIEW OF REFERENCE MONITOR ON A HOMOGENEOUS VAXCLUSTER



11.3 VAXCLUSTER OPERATING ENVIRONMENTS

The operating environment you may prepare for your cluster ranges from homogeneous to heterogeneous. The type you choose depends mainly on the processing needs of your site.

From a security standpoint, a cluster is a single management/security environment. A homogeneous cluster takes no special care once it is established.

A heterogeneous cluster takes more care in establishing and managing the management/security environment.

Table 11-4 describes security considerations for a homogeneous cluster environment, and Table 11-7 discusses security in a heterogeneous cluster environment.

TABLE 11-4: SECURITY CONSIDERATIONS FOR A HOMOGENEOUS CLUSTER

- THE USER OPERATING ENVIRONMENT SHOULD BE IDENTICAL ON EACH MEMBER NODE
- THE FILE SYSTEM WORKS LOCALLY ON EACH NODE TO PERFORM THE FILE PROTECTION CHECKING
- A USER ONLY LOGS INTO A SINGLE NODE AT A TIME
- MUST COORDINATE UICS AND SYSTEM FILES
- MAY BE RUN FROM THE SAME OR SEPARATE SYSTEM FILES
- ON A COMMON SYSTEM DISK, MOST OPERATING SYSTEM AND LAYERED PRODUCT FILES ARE STORED IN A COMMON ROOT DIRECTORY
- SYS$SYSROOT IS DEFINED AS A SEARCH LIST THAT POINTS TO A LOCAL ROOT FIRST (SYS$SPECIFIC), AND THEN TO THE COMMON ROOT (SYS$COMMON)
- SYS$MANAGER:, SYS$SYSTEM:, ETC. ALSO BECOME SEARCH LISTS (SYS$SPECIFIC:[SYSEXE] >>> SYS$COMMON:[SYSEXE])
- NEED TO BE AWARE OF WHERE FILES ARE CREATED (USE SYS$SPECIFIC:[SYSMGR] OR SYS$COMMON:[SYSMGR] INSTEAD OF SYS$MANAGER: FOR FILE CREATION)
- MAP USERS TO THEIR OWN ACCOUNT FOR PROXY ACCESS ON CLUSTER

  UAF> ADD/PROXY NODE::* *

Same:

UAF
RIGHTSLIST
NETUAF
SYSTEM DISK

TABLE 11-5: COORDINATING SYSTEM FILES ON A HOMOGENEOUS CLUSTER

- MUST COORDINATE THE USER ACCOUNTS FROM EACH NODE AND BUILD COMMON VERSIONS OF SYSUAF.DAT, NETUAF.DAT AND RIGHTSLIST.DAT
- VERY LITTLE COORDINATION IS NECESSARY FOR NEW SYSTEMS
- IF SYSUAF.DAT, NETUAF.DAT AND RIGHTSLIST.DAT CONTAIN THE RECORDS OF ESTABLISHED ACCOUNTS, YOU MUST COORDINATE AND COMBINE THE FILES (SEE TABLE 11-6)
- LESS COORDINATION IS NEEDED WHEN MERGING THE INDIVIDUAL NETUAF.DAT FILES
- UICS ARE NOT USED IN THE NETUAF RECORDS
- DECIDE WHICH EXISTING PROXY LOGIN RECORDS YOU WANT TO KEEP AND INCLUDE THESE RECORDS IN THE COMMON NETUAF.DAT FILE
- CAN SET UP THE COMMON FILES AS EITHER SHARED FILES IN A COMMON DIRECTORY OR SEPARATE DUPLICATE FILES
- IF YOU HAVE A COMMON SYSTEM DISK, PLACE FILES IN SYS$COMMON:[SYSEXE]

V4 NETUAF ⇒ NETPROXY in V5

- IF YOU DON'T HAVE A SHARED SYSTEM DISK, YOU MUST DECIDE WHERE TO LOCATE SECURITY DATABASE FILES

- COPY SYSUAF.DAT, NETUAF.DAT, RIGHTSLIST.DAT TO THE SHARED DIRECTORY

- DEFINE CLUSTER-WIDE LOGICAL NAMES TO POINT TO THEM

```
$ DEFINE/SYSTEM/EXEC   SYSUAF      WORK5:[SYSEXE]SYSUAF
$ DEFINE/SYSTEM/EXEC   NETUAF      WORK5:[SYSEXE]NETUAF
$ DEFINE/SYSTEM/EXEC   RIGHTSLIST  WORK5:[SYSEXE]RIGHTSLIST
                       VMSMAIL
```

- ADD THE DEFINE COMMANDS TO THE COMMON SYSTARTUP COMMAND FILE **(MUST BE DEFINED ON EACH NODE)**

- DELETE THESE FILES FROM THE SYSTEM DISK IN ORDER TO AVOID POSSIBLE CONFUSION

- MAY ALSO SHARE MAIL DATABASE (ASSIGN LOGICAL NAME VMSMAIL TO POINT TO VMSMAIL.DAT IN SHARED DIRECTORY)

TABLE 11-6: BUILDING A COMMON SYSUAF.DAT FILE ON UPGRADED SYSTEMS

1. BOOT EACH CLUSTER NODE IN THE SINGLE-SYSTEM ENVIRONMENT AND PRINT A LISTING OF SYSUAF.DAT

   ```
   $ SET DEF SYS$SYSTEM
   $ RUN AUTHORIZE
   UAF> LIST/FULL [*,*]
   ```

2. COMPARE THE ACCOUNTS FROM EACH NODE

   - DELETE ANY ACCOUNTS THAT YOU NO LONGER NEED
   - MAKE SURE THAT EACH USER ACCOUNT IN THE CLUSTER HAS A UNIQUE UIC
   - MAKE SURE THAT ACCOUNTS THAT PERFORM THE SAME TYPE OF WORK, HAVE THE SAME GROUP UIC
   - UICS MUST BE COORDINATED FOR ALL NODES ON THE CLUSTER
   - IF YOU CHANGE THE UIC FOR A PARTICULAR USER, YOU MUST ALSO CHANGE THE OWNER UICS FOR THAT USER'S EXISTING FILES AND DIRECTORIES
   - USE SET FILE/OWNER=UIC AND SET DIRECTORY/OWNER=UIC

3. CHOOSE THE SYSUAF.DAT FROM ONE OF THE NODES TO BE A MASTER SYSUAF.DAT

4. MERGE THE SYSUAF.DAT FILES FROM THE OTHER NODES TO THE MASTER SYSUAF.DAT

```
$ SET DEFAULT SYS$SYSTEM

$ CONVERT/EXCEPTION=EXCEPT.DAT [SYS1.SYSEXE]SYSUAF.DAT,-
  [SYS2.SYSEXE]SYSUAF.DAT SYSUAF.DAT


$_ASSIGN/USER  SYS$SYSTEM:EXCEPT.DAT  SYSUAF

$ RUN AUTHORIZE
UAF> LIST /FULL [*,*]
UAF> EXIT

$ PRINT SYSUAF.LIS
```

- CONVERT COMMAND ADDS THE RECORDS FROM [SYS1.SYSEXE]SYSUAF.DAT AND [SYS2.SYSEXE]SYSUAF.DAT TO THE FILE SYSUAF.DAT
- SYSUAF.DAT IS A MASTER COPY THAT CONTAINS RECORDS FROM THE OTHER SYSUAF.DAT FILES
- EXCEPTION RECORDS (DUPLICATE RECORDS NOT ADDED TO THE OUTPUT FILE) ARE WRITTEN TO EXCEPT.DAT
- READ THE EXCEPTIONS FILE WITH AUTHORIZE
- MAY CONTAIN CERTAIN UAF FIELD VALUES THAT ARE MORE APPROPRIATE THAN THOSE IN THE CORRESPONDING RECORDS THAT WERE ADDED TO THE FILE

5. MODIFY THE ACCOUNTS IN THE MASTER SYSUAF.DAT TO FIT YOUR NEW ENVIRONMENT

TABLE 11-7: SECURITY CONSIDERATIONS FOR A HETEROGENEOUS CLUSTER

- THE ENVIRONMENT ON EACH NODE IS SOMEWHAT UNIQUE
- USERS WORK IN AN OPERATING ENVIRONMENT THAT IS SPECIFIC TO THE NODE THEY ARE LOGGED IN TO
- CAN SET UP A CLUSTER THAT IS SOMEWHERE BETWEEN HOMOGENEOUS AND HETEROGENEOUS
- THE MORE HETEROGENEOUS THE CLUSTER, THE MORE CARE THAT HAS TO BE TAKEN IN THE SYSTEM/SECURITY MANAGEMENT DOMAIN
- MUST COORDINATE UAF, NETUAF, AND RIGHTSLIST FILES
- UICS AND IDENTIFIERS MUST BE UNIQUE CLUSTER-WIDE
- STILL HAVE ONE SECURITY DOMAIN
- MUST LOOK AT ACCESS FROM EACH NODE INDIVIDUALLY
- SHARED DISKS WILL BE ACCESSIBLE CLUSTER-WIDE
- MAY SET UP LOCAL NON-SHARED DISKS
  - DON'T NEED TO LOAD MSCP SERVER ON SYSTEMS WHERE NO LOCAL DISKS WILL BE SHARED
  - CAN STILL ACCESS THEM REMOTELY VIA DECNET

CHAPTER 12
CONCLUSION

LIST OF TOPICS

- Selling Security
- Other Considerations
- VMS Security Futures
  - NON-DISCRETIONARY Controls
    - › Lattice Model - Military Access Classes
    - › Preventing the 'Trojan Horse'
  - VMS V4 DoD Security Evaluation
- Security Kernels
  - › Security Kernel Principles
  - › Design Methodology
  - › Operating Systems Implementations
- OTHER ENHANCEMENTS AND TRENDS

12.1 SELLING SECURITY

We have discussed many feature that VMS V4 provides for enhancing system security. Keep in mind that not all features are for all sites.

Increasing the security of a site will cost something. The costs of all protective measures must be weighed against the alternative of having the information lost or destroyed.

Once there is an established need for security, the idea must be sold to management and to the users.

TABLE 12-1: NEEDS OF SECURITY

- WHAT IS YOUR PRESENT SECURITY PROGRAM?
- DO YOU HAVE PROPRIETARY INFORMATION?
- IS IT BEING THREATENED?
- WHAT ARE THE POSSIBLE SOURCES OF THREAT?
- WHAT CAN I DO TO PROTECT MY SITE AGAINST THESE THREATS?
- HOW CAN I ENLIST THE SUPPORT OF MANAGEMENT IN IMPLEMENTING THESE MEASURES?
- HOW CAN I ENLIST THE SUPPORT OF THE USERS IN IMPLEMENTING THESE MEASURES?
- WHAT EFFECT ON USERS WILL THE SECURITY MEASURES HAVE?
- HOW WILL I EDUCATE THE USERS ABOUT THIS PROGRAM?
- WHAT ARE THE COSTS?
  - DOLLARS
  - DECREASE IN MACHINE PERFORMANCE AND COMPANY PRODUCTIVITY
  - USER TIME AND INCONVENIENCE
  - ADMINISTRATIVE TIME

TABLE 12-2: SELLING SECURITY TO MANAGEMENT

- SECURITY IS A BUSINESS PROBLEM
- ESTABLISH THE NEED FOR SECURITY
- SECURITY IS A STATE OF MIND
  IT MUST BE IMPLEMENTED AT ALL LEVELS OF THE COMPANY
- DO THE BENEFITS OUTWEIGH THE COSTS
- MANAGEMENT DOES NOT NEED TO KNOW THE TECHNICAL DETAILS OF THE PROGRAM
- START OUT WITH A SMALL PROPOSAL
- DATA IS AN INTANGIBLE OBJECT
- SECURITY PROTECTS A SERVICE
- SECURITY HELPS YOU SLEEP BETTER

TABLE 12-3: EDUCATING THE USER

- TELL USERS YOUR POLICIES AND PROCEDURES
- PROTECT THEIR FILES AND PASSWORDS
- OBSERVE THE SYSTEM
- FOR NEW USERS LET THEM KNOW
  - LOCATION OF THE ACCOUNT AND TERMINALS TO BE USED
  - TYPES OF RESTRICTIONS ON THE ACCOUNT
  - THE PHONE NUMBER AND PROCEDURE FOR DIALING IN
  - ACCOUNT/PASSWORD DURATION AND MAINTENANCE
  - BACKUP PROCEDURES
  - USERNAME/IDENTIFIERS/PRIVILEGES/QUOTAS
  - DEFAULT DEVICE/DIRECTORY
  - DEFAULT FILE PROTECTION
  - DISK QUOTA
  - PROXY/NETWORK ACCESS

12.2 OTHER CONSIDERATIONS

Protection of proprietary data continues to be of major concern in industry and government. VAX/VMS provides many tools to ensure the security of data from a system management level. Through effective use of access control, UIC and ACL based protection, security auditing, disk maintenance, and system monitoring, data can be protected from unauthorized abuse.

In addition to the technical safeguarding of proprietary information, the system/security manager must monitor the behavior of data processing personnel in regards to the use of that information.

Ethical use of sensitive data is of critical importance in guarding the integrity of your company. Infiltrators of security breaches and violations of company policy must be discovered and disciplined in order to successfully maintain system security.

To prevent the unethical intruder from gaining access to data, you must be technically competent, follow ethical professional standards, and take breaches of those standards and company policy seriously.

12.3 VMS SECURITY FUTURES

VMS V4 has implemented many advanced security features. It is by no means a completely secure operating system. The network is not a completely secure network. Digital is continuing to work on enhancing VMS security.

These enhancements include work on nondiscretionary controls (see section 12.3.1, a VMS security kernel, (see section 12.3.3, extensions of ACL's to other objects and additional DCL commands to aid in implementing security features (see Section 12.3.4.)

NOTE

Discussion of features desired in future versions of VMS in no way makes a commitment from Digital regarding the availability of any of these features.

NON-DISCRETIONARY CONTROLS

Non-discretionary controls are such that the owner of a file may not arbitrarily grant access to anyone he/she wishes. The system is a fixed authorization system based on some predetermined policy such as the Department of Defense controls for national security information. These type of controls also have many commercial applications such as for use in large time-sharing systems.

Lattice Model - Military Access Classes -

The most widely know non-discretionary security policy is the Department of Defense controls for national security information. A user must have a clearance in order to use a piece of information. Each piece of information has a classification. The owner of a file does not have the ability to change his/her clearance or to change the classification of a piece of information.

The DoD classification system consists of four security levels : UNCLASSIFIED, CONFIDENTIAL, SECRET and TOP SECRET. There are also a large number of categories such as NATO, NUCLEAR, FOREIGN, etc. These categories represent a 'need to know' classification.

For a user to gain access to a document, he/she must not only be cleared for the security level of the document, but also must be authorized for all of the compartments of that document. e.g. If a CLASSIFIED document has both NUCLEAR and FOREIGN data, for someone to read that document, he/she must be classified at SECRET or TOP SECRET and must all be authorized for both NUCLEAR and FOREIGN data.

Lattice controls also are useful in describing non-military applications. It would allow for two corporations to buy time on the same computer, and have complete access to their own information, but have no access to the others data.

Preventing The 'Trojan Horse' -

Non-discretionary security systems are a method of preventing 'Trojan horse' attacks. Bell and LaPadula from the Mitre Corporation developed a mathematical model in 1976 that effectively protects against 'Trojan horses'.

TABLE 12-4: ACCESS RULES FOR BELL AND LAPADULA MODEL

1. FOR READ PERMISSION, THE ACCESS CLASS OF THE OBJECT (THE DATA) MUST BE LESS THAN OR EQUAL TO THE ACCESS CLASS OF THE SUBJECT (THE READER).

   THIS IS THE 'SIMPLE SECURITY PROPERTY'.

2. FOR WRITE PERMISSION, THE ACCESS CLASS OF THE SUBJECT (THE WRITER) MUST BE LESS THAN OR EQUAL TO THE ACCESS CLASS OF THE OBJECT (THE DESTINATION).

   THIS PREVENTS DOWNGRADING INFORMATION AND WILL PREVENT A 'TROJAN HORSE' FROM LEAKING INFORMATION.

NOTE

VMS V4 DOES NOT have a non-discretionary security system in place.

VMS V4 DOD SECURITY EVALUATION

The DoD Computer Security Evaluation Center (CSEC) classifies operating systems and programming languages based on their suitability for processing classified information. These classifications define four hierarchical divisions, A, B, C, and D (A being the most trusted). Within each division except D there are numbers classes. The higher the class number the greater the trust that can be placed in the system.

Table 12-5 lists the evaluation of some operating systems.

The current DoD CSEC evaluation of VMS V4 is C2.

TABLE 12-5: DOD CSEC EVALUATIONS

LEVEL	DEFINITION	EXAMPLE
D	COMMON PRACTICE	OS/360
C1	DISCRETIONARY SECURITY	VMS V3.0
C2	CONTROLLED ACCESS	MVS W/ACF2 VMS V4.0
B1	LABELED SECURITY PROTECTION	
B2	STRUCTURED PROTECTION	MULTICS
B3	SECURITY DOMAINS	
A1	VERIFIED DESIGN	HONEYWELL SCOMP, KSOS-11
A2	VERIFIED IMPLEMENTATION	???

SECURITY KERNELS

A TRULY SECURE OPERATING SYSTEM NOT ONLY HAS THE CORRECT SET OF FEATURES BUT ALSO MUST HAVE THOSE FEATURE ENFORCED CORRECTLY. A SECURITY KERNEL, A SET OF SECURITY SOFTWARE THAT WOULD EFFECTIVELY ENFORCE COMPUTER SECURITY IS A METHOD OF DEVELOPING AND IMPLEMENTING A VERIFIABLE SECURE OPERATING SYSTEM.

TABLE 12-6 LISTS THE THREE BASIC SECURITY KERNEL PRINCIPLES, WHILE TABLE 12-7 LISTS THE GUIDELINES THAT THE DEVELOPMENT OF A SECURITY KERNEL MUST FOLLOW.

SECURITY KERNEL PRINCIPLES -

TABLE 12-6: SECURITY KERNEL PRINCIPLES

1. COMPLETE MEDIATION

   THE KERNEL MUST CHECK EVERY REFERENCE TO DATA FOR SECURITY MEDIATION.

2. ISOLATION

   THE KERNEL SOFTWARE MUST BE PROTECTED FROM TAMPERING BY NON-SECURITY SOFTWARE.

3. SIMPLICITY

   THE KERNEL MUST BE SMALL AND SIMPLE ENOUGH TO BE VERIFIED CORRECT.

DESIGN METHODOLOGY FOR A SECURITY KERNEL -

TABLE 12-7: DESIGN METHODOLOGY FOR A SECURITY KERNEL

1. DEVELOP A MATHEMATICAL MODEL OF THE SECURITY POLICY THAT WILL BE ENFORCED BY THE KERNEL. (I.E. LIKE THE LATTICE MODEL)

2. DEVELOP A FORMAL SPECIFICATION OF THE SECURITY KERNEL.

3. CODE THE KERNEL IN A HIGH-LEVEL LANGUAGE FROM THE FORMAL SPECIFICATION.

4. PROVE THE HIGH-LEVEL LANGUAGE IMPLEMENTATION CONSISTENT WITH THE FORMAL SPECIFICATION.

OPERATING SYSTEMS IMPLEMENTATIONS -

UNTIL ALL FACETS FROM THE MODELS TO THE CHIPS CAN BE VERIFIED, A COMPLETELY SECURE OPERATING SYSTEM KERNEL CANNOT BE GUARANTEED. USING THE DEVELOPMENT METHODS HOWEVER WILL HELP TO PROVIDE A MORE SECURE OPERATING SYSTEM.

THERE IS CURRENTLY DEVELOPMENT WORK BEING DONE ON OPERATING SYSTEMS USING THESE METHODS. VMS V4 DOES NOT HAVE A SECURITY KERNEL, BUT THE ARCHITECTURE (A RING-BASED SYSTEM) DOES NOT PRECLUDE ITS DEVELOPMENT FOR THE VAX. WORK IS CURRENTLY BEING DONE ON DEVELOPING A SECURITY KERNEL FOR THE VAX/VMS ARCHITECTURE.

OTHER ENHANCEMENTS AND TRENDS

- AUDITING OF TERMINAL I/O
  - $ SET TERMINAL/AUDIT/ENABLE=(ITEM) DEVICE-NAME:
  - $ SET TERMINAL/AUDIT/DISABLE=(ITEM) DEVICE-NAME:
  - $ SET TERMINAL/PERMANENT/AUDIT/ENABLE=(READ,WRITE) TTC3:

- AUDITING OF MAILBOX WRITES
  - $ SET DEVICE/AUDIT/ENABLE=(WRITE) DEVICE-NAME:
  - $ SET DEVICE/AUDIT/DISABLE=(WRITE) DEVICE-NAME:
  - $ SET DEVICE/AUDIT/ENABLE=(WRITE) MBA135:

- AUDIT JOURNALING
  - $ SET AUDIT /JOURNAL /ENABLE=(TYPE)
  - $ SET AUDIT /JOURNAL /DISABLE=(TYPE)
  - $ JOURNAL FILE WILL HAVE DETAILED INFORMATION ON EVENTS
  - $ CAN ANALYZE WITH ANALYZE/AUDIT

- ACL'S ON MORE SYSTEM OBJECTS
  - LOGICAL NAME TABLES, QUEUES, MAILBOXES

- PROTECTED SUBSYSTEMS
  - MORE SECURE THAN CAPTIVE COMMAND PROCEDURES

- NON-DISCRETIONARY CONTROLS
  - FLOW OF INFORMATION OUTSIDE OF USERS' CONTROL
  - 0-255 SECURITY LEVELS
  - 64 SECURITY CATEGORIES

- MORE SELECTIVE PROXY ACCESS
  - PROXY ACCESS TO FILES
  - MAP ONE REMOTE USER TO MORE THAN ONE PROXY ACCOUNT - DEPENDING ON REQUEST
  - USER CONTROL OF ESTABLISHING PROXY

APPENDIX A
SAMPLE SECURITY COMMAND PROCEDURES

EXAMPLE A-1: CHECKPRIV.COM

```
$ ! CHECKPRIV.COM
$
$ ! This command procedure is to be inserted in SYS$MANAGER:SYLOGIN.COM
$ ! and is to prompt a user for reason to log in a privileged account.
$
$ ! Modified Oct 30, 1984 by Henry Teng
$ !    Change the scheme to append priv.tmp to priv.log
$ !     so that priv.log has world no access.
$
$ ! Modified Feb 20, 1985 by Scott Stuart
$ !    Change the method of checking for privileges to include
$ !    authorized and default privileges
$
$ On control_y then logout
$ Set noon
$ !    i. get both the authorized and default privileges
$ !   ii. try to subtract any desired privilege
$ !  iii. compare the old and new string
$ !       if they are different
$ !         then the process has at least one of the privileges
$ !
$ all_privs= f$getjpi("","AUTHPRIV") + f$getjpi("",DEFPRIV")
$ new_priv = all_priv - "OPER" - "WORLD" - "DIAGNOSE" - "SYSGBL" -
             - "VOLPRO" - "BYPASS" - "CMEXEC" - "CMKRNL" - "DETACH" -
             - "LOG_IO" - "PFNMAP" - "PHY_IO" - "SETPRV" - "SYSNAM" -
             - "SYSPRV" - "READALL"
$ !
$ if all_privs .eqs. new_priv then exit    !nonprivileged user
$ !
$ Priv_confirmed:
$
$ Open/error=blowup/write priv$log priv.tmp
$ X:= 'f$getjpi("","LOGINTIM")'" "'f$getjpi("", "USERNAME")'" "-
  'f$getjpi("", "MODE")'" "'f$getjpi("", "TERMINAL")'
$ Write/error=blowup priv$log x
$
$ If f$mode() .NES. "INTERACTIVE" then goto c6
$
$ ! The following assignment trims off trailing spaces.
$
$ Username := 'f$getjpi("", "USERNAME")'
$ If username .eqs. "BACKUP" then goto c6
$ Type sys$input
  Please state your reason for logging into this privileged account.
$
```

```
$
$ C4:
$
$ Read/end_of_file=c4/error=c4/prompt="Reason:  " sys$command reason
$ If reason .eqs. "" then goto c4
$ Write/error=blowup priv$log reason
$ Write sys$output "Thank you"
$
$ C6:
$
$ Close/error=blowup priv$log
$ Append priv.tmp sys$manager:priv.sav
$ Delete priv.tmp;
$
$ If .not.f$privilege("SETPRV") then goto no_setprv
$
$ Old_priv = f$setprv("SYSPRV")
$
$ No_setprv:
$
$ If .not.f$privilege("SYSPRV") then goto blowup
$ ! Time to update priv.log from priv.sav
$ Append sys$manager:priv.sav sys$manager:priv.log
$ Delete sys$manager:priv.sav;
$
$ ! Create another priv.sav
$
$ Open/write priv sys$manager:priv.sav
$ Close priv
$ Set prot=(w:rw) sys$manager:priv.sav
$ If f$privilege("SETPRV") then old_priv = f$setprv(old_priv)
$
$ Blowup:
$
$ Exit
```

EXAMPLE A-2: REPORTPRIV.COM

```
$! REPORTPIRV.COM by Henry Teng
$!
$! This command file is to send a report on activities of privileged
$! accounts to the system manager every week and also cleans up
$! priv.old. To activate REPORTPRIV.COM, type the following command:
$! $ submit sys$manager:reportprv.com
$! To have reportpriv.com working properly, sys$manager:checkpriv.com
$! has to be inserted in sylogin.com.
$
$ Set noon
$ Old_uic = f$user()
$ Old_priv = f$setprv("all")
$ Set uic [1,4]
$
$! Resubmit this command file for a week from now.
$
$ Submit sys$manager:reportprv.com/after="03:00+7-"/keep/noprinter
$
$! Deal with log of privileged logins.
$! Update priv.log from priv.sav
$
$ Append sys$manager:priv.sav sys$manager:priv.log
$ Delete sys$manager:priv.sav;*
$ Open/write privsav sys$manager:priv.sav
$ Close privsav
$ Set prot=(w:rw) sys$manager:priv.sav
$
$ Open/write privlog sys$manager:priv.new
$ Close privlog
$ Set prot:(s:rwe,o:rwed,g:rw,w) sys$manager:priv.new
$ Rename/log/new_version sys$manager:priv.new *.log;
$ Rename/log/new_version sys$manager:priv.log;-1 *.old;
$ Set prot:w sys$manager:priv.old;*
$
$! Clean up files older than 10 weeks.
$
$ Delete/log/modified/before="-70-" sys$manager:priv.old;*
$ If f$search("sys$manager:priv.old").nes."" then goto mail_report
$ Open/write priv_old sys$manager:priv.old
$ Write priv_old "No privileged logins recorded yet."
$ Close priv_old
$
$ Mail_report:
$
$ Mail/subject="Report on Usage of Privileged Accounts"  -
                sys$manager:priv.old system
$ Purge/keep=3 reportprv.log
$ Set uic 'old_uic'
$ Old_priv = f$setprv("''old_priv'")
$ Exit
```

EXAMPLE A-3: REPORT_WORLD_V4.COM

```
$ !     REPORT_WORLD_V4.COM by Henry Teng
$ !                    See FIND
$ !     This command procedure produces a list of all world readable
$ !     files on a device.  It does so using DIRECTORY and SEARCH.
$ !     It requires CMKRNL privilege or SETPRV, since it sets
$ !     the UIC of the process to world.  It clears the privs SYSPRV
$ !     and BYPASS so the the directory checking will work properly.
$ !
$ ! Parameters:
$ !
$ !     P1 = Device to check
$ !
$ !
$ ! Do parameter defaulting/checking.  The variable DEVNAM is
$ ! set to the device for which the check should be done.
$ !
$     verisav = f$verify (0)
$     olduic = f$user()
$ Old_priv = f$setprv("all")
$ Set uic [1,4]
$ Set verify
$     Submit/parameter='p1' sys$manager:report_world_v4.com -
        /after="04:00+7-"/keep/noprinter
$
$ ! Create version number via time stamp to handle two disks or more
$
$      Time_stamp = f$time()
$     Dot = f$locate(".", time_stamp)
$     Version_length = f$length(time_stamp) - dot - 1
$     Version = f$extract(dot+1, version_length, time_stamp)
$     DEVNAM = P1 - ":"
$ !
$     LISTNAME = "WORLD''version'.LIS"
$ !
$ !
$ ! Create a file which contains the top level directories which need
$ ! to be checked.
$ !
$     set proc/priv=(nobypass,nosysprv,cmkrnl)
$     Set on
$     on control_y then goto cleanup
$     on ERROR then goto MFDCLS
$     open/write world_file 'listname'
$
$     directory/notrail/prot/owner/out=0MFD'VERSION'.TMP -
 /exclude=(000000.dir,001001.dir,001002.dir,001006.dir,010040.dir, -
 netlib.dir,syserr.dir,sysexe.dir,syshlp.dir,syslib.dir,sysmaint.dir, -
 sysmsg.dir,system.dir,systest.dir,sysupd.dir,syslost.dir) -
    'DEVNAM':[0,0]*.DIR;
$
$ !
$ ! Open the directory listing file for input
$ !
```

```
$     open/read/error=MFDOPN 0MFD 0MFD'VERSION'.TMP
$ !
$ ! Main program loop.   Read directory entries from the master file
$ ! until an end-of-file is reached.
$ !
$ Set noverify
$MFDNXT:
$ Set on
$ On error then goto mfdnxt
$     read/error=MFDERR/end_of_file=MFDCLS 0MFD MFDREC
$
$ if f$locate("Directory", MFDREC).ne.f$length(MFDREC) then goto mfdnxt
$ !
$ ! Extract the directory name from the master file entry
$ !
$     RECLEN = f$length (MFDREC)
$     CB = f$locate ("]", MFDREC) + 1
$     if CB .gt. RECLEN then goto MFDNXT
$     FILEN = RECLEN - CB
$     FILREC = f$extract (0, FILEN, MFDREC)
$     EXT = f$locate (".", FILREC)
$     if EXT .eq. RECLEN then goto MFDNXT
$     UFDNAM == f$extract (0, EXT, FILREC)
$     User = ufdnam
$ !
$ ! Next extract the owner of the directory and skip those that
$ ! are owned by [001,004].
$ !
$     OB = f$locate ("[", MFDREC)
$     CB = f$locate ("]", MFDREC)
$     LEN = CB - OB + 1
$       if LEN .eq. 1 then goto MFDNXT
$     OWNER = f$extract (OB, LEN, mfdrec)
$ if OWNER.eqs."[001,004]".OR. OWNER.EQS."[SYSTEM]" then goto MFDNXT
$ !
$ ! Check the directory for world read access
$ !
$ !      ob = f$locate("(", mfdrec)
$ !      cb = f$locate(")", mfdrec)
$ !     filen = cb - ob + 1
$ !     FILREC = f$extract (ob, FILEN, mfdREC)'
$ !     COMMA = 'f$locate (",", FILREC)' + 1
$ !     if 'COMMA' .gt. 'FILEN' then goto MFDNXT
$ !     TMP = FILEN - COMMA
$ !     PROT := 'f$extract (COMMA, TMP, FILREC)'
$ !     COMMA = 'f$locate (",", PROT)' + 1
$ !     if 'COMMA' .gt. 'TMP' then goto MFDNXT
$ !     TMP = TMP - COMMA
$ !     TMPSTR := 'f$extract (COMMA, TMP, PROT)'
$ !     COMMA = 'f$locate (",", TMPSTR)' + 1
$ !     if 'COMMA' .gt. 'TMP' then goto MFDNXT
$ !     PROT := 'f$extract (COMMA, 1, TMPSTR)'
$ !     if "''PROT'" .nes. "R" then goto MFDNXT
$ !     write world_file ""
$ !     Write world_file "''devnam':[0,0]''user'.dir;1"
```

```
$ !    Write world_file ""
$ !
$ ! The directory has world read access.  Get a directory listing of
$ ! files and subdirectories which are world readable.
$ !
$
$ File_count = 0
$
$ Search_file:
$
$    File_checked = f$search ("''devnam':[''user'...]*.*;*", 1)
$    If file_checked.eqs."" then goto mfdnxt
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=R", protection).ne.f$length(protection) -
    Then goto world_found
$    If f$locate("W:R", protection).eq.f$length(protection) -
    Then goto search_file
$ World_found:
$
$ ! found file world readable
$    if file_count.eq.0 then write world_file "<FF>"
$    file_count = file_count + 1
$    write world_file file_checked
$    goto search_file
$
$MFDCLS:
$
$ Search_sysuaf:
$ ! Check protection for sysuaf.dat
$
$    File_checked = f$search ("sys$system:sysuaf.*;*", 1)
$    If file_checked.eqs."" then goto search_systarnet
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=R", protection).ne.f$length(protection) -
    Then goto sysuaf_world_found
$    If f$locate("W:R", protection).eq.f$length(protection) -
    Then goto search_sysuaf
$ Sysuaf_world_found:
$
$ ! found file world readable
$    write world_file file_checked
$    goto search_sysuaf
$
$ Search_systarnet:
$ ! Check protection for systarnet.*
$
$    File_checked = f$search ("sys$manager:systarnet.*;*", 1)
$    If file_checked.eqs."" then goto search_netuaf
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=R", protection).ne.f$length(protection) -
    Then goto sysuaf_world_found
$    If f$locate("W:R", protection).eq.f$length(protection) -
    Then goto search_systarnet
$ Systarnet_world_found:
$
```

```
$ ! found file world readable
$    write world_file file_checked
$    goto search_systarnet
$
$ Search_netuaf:
$ ! Check protection for netuaf.*
$
$    File_checked = f$search ("sys$system:netuaf.*;*", 1)
$    If file_checked.eqs."" then goto search_systartup
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=R", protection).ne.f$length(protection) -
    Then goto netuaf_world_found
$    If f$locate("W:R", protection).eq.f$length(protection) -
    Then goto search_netuaf
$ Netuaf_world_found:
$
$ ! found file world readable
$    write world_file file_checked
$    goto search_netuaf
$
$ Search_systartup:
$ ! Check protection for systartup.*
$
$    File_checked = f$search ("sys$manager:systartup.com;*", 1)
$    If file_checked.eqs."" then goto search_sys$manager
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=R", protection).ne.f$length(protection) -
    Then goto systartup_world_found
$    If f$locate("W:R", protection).eq.f$length(protection) -
    Then goto search_systartup
$ Systartup_world_found:
$
$ ! found file world readable
$    write world_file file_checked
$    goto search_systartup
$
$!Search for world writable files in sys$manager,sys$system,sys$update
$
$ Search_sys$manager:
$ ! Check protection for sys$manager:*.com;
$
$    File_checked = f$search ("sys$manager:*.com;", 1)
$    If file_checked.eqs."" then goto search_sys$system
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=RW", protection).ne.f$length(protection) -
    Then goto sys$manager_world_found
$    If f$locate("W:RW", protection).eq.f$length(protection) -
    Then goto search_sys$manager
$ Sys$manager_world_found:
$
$ ! found file world writable
$    write world_file "The following file is WORLD WRITABLE.",-
                      " Your system may need a security checkup."
$    write world_file file_checked
$    goto search_sys$manager
```

```
$
$ Search_sys$system:
$ ! Check protection for sys$system:*.com;
$
$    File_checked = f$search ("sys$system:*.com;", 1)
$    If file_checked.eqs."" then goto search_sys$update
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=RW", protection).ne.f$length(protection) -
     Then goto sys$system_world_found
$    If f$locate("W:RW", protection).eq.f$length(protection) -
     Then goto search_sys$system
$ sys$system_world_found:
$
$ ! found file world writable
$    write world_file "The following file is WORLD WRITABLE.", -
                      " Your system may need a security checkup."
$    write world_file file_checked
$    goto search_sys$system
$
$ Search_sys$update:
$ ! Check protection for sys$update:*.com;
$
$    File_checked = f$search ("sys$update:*.com;", 1)
$    If file_checked.eqs."" then goto cleanup
$    Protection = f$file_attribute(file_checked, "pro")
$    If f$locate("WORLD=RW", protection).ne.f$length(protection) -
     Then goto sys$update_world_found
$    If f$locate("W:RW", protection).eq.f$length(protection) -
     Then goto search_sys$update
$ sys$update_world_found:
$
$ ! found file world writable
$    write world_file "The following file is WORLD WRITABLE.",-
                      " Your system may need a security checkup."
$    write world_file file_checked
$    goto search_sys$update
$
$cleanup:
$
$ Set noon
$ close 0MFD
$ close world_file
$
$ If f$file_attri("''listname'", "eof").ne.0 then goto mail_list
$ Open/write world_file 'listname'
$ Write world_file "No files were found world readable."
$ Close world_file
$
$ Mail_list:
$
$ Mail/subject="List of World Readable Files on ''devnam'" 'listname' system
$ delete 0MFD'VERSION'.TMP;*, 'listname';*
$ Purge/keep=6 reportwrd.log
$ verisav = f$verify(verisav)
$ Set uic 'olduic'
```

```
$ Old_priv = f$setprv("''old_priv'")
$ set on
$ exit
$
$MFDOPN:
$ write sys$output "Error opening master file - program aborted"
$ goto error_exit
$MFDERR:
$ write sys$output "Error reading master file - program aborted
$
$ Error_exit:
$
$ Set noon
$ close 0MFD
$ close world_file
$ set uic 'olduic'
$ Old_priv = f$setprv("''old_priv'")
$ Purge/keep=6 reportwrd.log
$ Set on
$ Exit
```

EXAMPLE A-4: SET UP FOR GUEST ACCOUNT - GUEST.COM

```
$ ! HOST.COM  By Henry Teng  Jan 17, 1985
$
$ ! This command procedure with the following account setup
$ ! should be secure enough to prevent an intruder from
$ ! breaking out of the captive command procedure.
$ ! Make sure that CONTROL_Y is disabled
$ ! Make sure that HOST.COM has the protection code of W:E.
$ ! Make sure that the group UIC is unique.
$ !
$ ! Account setup for HOST:
$ ! Username: HOST                                Owner:
$ ! Account:                                      UIC:     [444,1] ([HOST])
$ ! CLI:      DCL                                 Tables: DCLTABLES
$ ! Default:  SYS$SYSDEVICE:[HOST]
$ ! LGICMD:   SYS$MANAGER:HOST
$ ! Login Flags:Disctly Defcli Lockpwd Captive Diswelcome Disnewmail
$ !                Dismail Genpwd
$!Primary days:   Mon Tue Wed Thu Fri
$!Secondary days:                    Sat Sun
$!Primary  000000000011111111112222 Secondary000000000011111111112222
$!Day Hours012345678901234567890123 Day Hours012345678901234567890123
$!Network: -----  No access  ------           -----  No access  ------
$!Batch:   -----  No access  ------           -----  No access  ------
$!Local:   ##### Full access ######           ##### Full access ######
$!Dialup:  ##### Full access ######           ##### Full access ######
$!Remote:  ##### Full access ######           ##### Full access ######
$! Expiration:            (none)   Pwdminimum:  6   Login Fails:     0
$ ! Pwdlifetime:           (none)  Pwdchange:  17-JAN-1985 14:21
$ ! Maxjobs:          0  Fillm:        20  Bytlm:        20000
$ ! Maxacctjobs:      0  Shrfillm:      0  Pbytlm:           0
$ ! Maxdetach:        0  BIOlm:        18  JTquota:       1024
$ ! Prclm:            2  DIOlm:        18  WSdef:          150
$ ! Prio:             4  ASTlm:        24  WSquo:          200
$ ! Queprio:          0  TQElm:        10  WSextent:       500
$ ! CPU:         (none)  Enqlm:        30  Pgflquo:      10000
$ ! Authorized Privileges:
$ !   TMPMBX NETMBX
$ ! Default Privileges:
$ !   TMPMBX NETMBX
$
$ On control_y then logout
$ On error then logout
$ Delete/sym/global/all
$ Delete/sym/local/all
$ Read/prompt="Please enter node name: " sys$command node_name
$ If f$locate("F$", node_name).ne.f$length(node_name) then logout
$ If f$locate("f$", node_name).ne.f$length(node_name) then logout
$ If f$locate("@", node_name).ne.f$length(node_name) then logout
$ If f$locate("=", node_name).ne.f$length(node_name) then logout
$ Set host 'node_name
$ Logout
```

APPENDIX B
SECURITY EXERCISES

1. One of your programmers has returned from DECUS with a new language sensitive editor. He gives you the tape and asks you to install the program. What do you do?

2. What DCL command will allow you to look for 'PASSWORD GRABBER' programs on the disk WORK3:?

3. Supply the AUTHORIZE commands to add a user SARA with the following restrictions :
   a. works 9-6, Wednesday through Sunday
   b. is off Monday and Tuesday.
   c. she needs local, dialup and batch access on the days that she is working
   d. she will need to run batch jobs every evening from 6-12
   e. she should be permitted no other access
   f. her password should be at least 8 characters and generated
   g. make the initial password 'USER'
   h. make the password expired
   i. set up a password lifetime of 3 months
   j. set up the account to become inactive on July 4, 1986

4. Work through the access request examples on pages 4-43 and 4-44 of the Guide To VAX/VMS System Security.

5. Set up the file WORK2:[TEXTFILES.HIDDEN]JUNK.TXT to be a READ ONLY FILE (no copying of the file) to any holder of the identifer READ_ONLY. Use a command procedure WORK2:[COM_FILE]VIEW.COM.

6. The type of access protection needed determines the type of ACE used in a given situation. What are the three types of ACEs?
   a.
   b.
   c.

7. System-defined identifiers are automatically created by the system when the system manager creates a rights database. List the six system-defined identifiers.

a.
b.
c.
d.
e.
f.

8. The ACCESS field of an identifier specifies what type of access is allowed. Match the following access actions to the type of access they allow. Each access action may be used once, more than once, or not at all.

Type of Access Allowed	Access Action
_____ Accessor can read a file, read from a disk, or allocate a device	a. Execute
_____ Accessor can read or write å file	b. None
_____ Accessor can execute an image file, and look up entries in a directory without using wildcard characters	c. Control
_____ Accessor has all privileges of the objects actual owner	d. Write
_____ Accessor has no access to the object	e. Delete
	f. Read

9. The OPTIONS field of an identifier controls whether an ACE is propagated, can be displayed, or even deleted. Match the following options to the system action(s) they indicate. Each option may be used once, more than once, or not at all.

System Action	Option
_____When copying an ACL from one version of a file to another, the ACE is not duplicated	a. Default b. Protected c. Nopropagate d. None
_____An ACE is to be included in the ACL of any files created within a directory	
_____An ACE will be preserved even when an attempt is made to delete the entire ACL	

10. The AUTHORIZE utility is used to create new identifiers and allow users to become holders of them. Supply the AUTHORIZE command that will perform the following:
    a. Add a general identifier called LUCKY
    b. Add a UIC identifier for group [300,*] called PROJECT_300
    c. Grant identifier LUCKY to user DANTON
    d. Show all holders of identifier LUCKY
    e. Show identifiers held by DANTON

11. What steps should you take if a user tells you that she cannot explain why the last login message indicates that her last interactive login was at 2:00 pm on Sunday, and that she was out of town over the weekend.

12. Work through the example of setting up protected file sharing on DECnet on pages 7-23 and 7-24 of the Guide To VAX/VMS System Security.

13. What DCL command will allow you to look for passwords embedded in command procedures on the disk WORK3:?

SOLUTIONS TO EXERCISES

1. DON'T PUT IT ON

   a. INSIST ON GETTING THE SOURCE PROGRAM

   b. READ THE SOURCES

   c. HAVE SOMEONE ELSE READ THE SOURCES

   d. IF SATISFIED, COMPILE, LINK AND INSTALL THAT VERSION

2. From a BATCH job or as a SPAWNED/NOWAIT subprocess execute the following command:

```
$ SEARCH/OUTPUT=PWD.LIS  WORK3:[*...]*.*;*  "PASSWORD","USERNAME"
```

3. EXECUTE THE FOLLOWING AUTHORIZE COMMANDS

```
UAF> ADD SARA                                                    -
     /PRIMEDAYS=(NOMON,NOTUE,WED,THU,FRI,SAT,SUN)                -
     /LOCAL=(PRIMARY,9-17) /DIALUP=(PRIMARY,9-17)                -
     /BATCH=(PRIMARY,6-23,SECONDARY,17-23)                       -
     /FLAGS=(GENPWD) /PWDMINIMUM=8 /PWDEXPIRED                   -
     /PWDLIFETIME="90-" /EXPIRATION=4-JUL-1986
```

4. SEE PAGE 4-57 **Guide To VAX/VMS System Security**

5. CREATE THE FILE VIEW.COM AND ADD THE FOLLOWING LINE :

```
$ TYPE WORK2:[TEXTFILES.HIDDEN]JUNK.TXT
```

NEXT, SET UP THE FOLLOWING ACLS EITHER WITH THE DCL COMMANDS SHOWN BELOW OR BY USING THE ACL EDITOR.

```
$ SET DIRECTORY/ACL=(ID=READ_ONLY,ACCESS=NONE)                   -
        WORK2:[TEXTFILES]

$ SET DIRECTORY/ACL=(ID=READ_ONLY,ACCESS=EXECUTE)                -
        WORK2:[TEXTFILES.HIDDEN]

$ SET FILE/ACL=(ID=READ_ONLY,ACCESS=READ)                        -
        WORK2:[TEXTFILES.HIDDEN]JUNK.TXT

$ SET DIRECTORY/ACL=(ID=READ_ONLY,ACCESS=EXECUTE)                -
        WORK2:[COM_FILES]

$ SET FILE/ACL=(ID=READ_ONLY,ACCESS=EXECUTE)                     -
        WORK2:[COM_FILES]VIEW.COM
```

6. What are the three types of ACEs?
   a. Identifier
   b. Default Protection
   c. Security Alarm

7. List the six system-defined identifiers.
   a. Interactive
   b. Batch
   c. Network
   d. Local
   e. Dialup
   f. Remote

8. Match the following access actions to the type of access they allow.

Type of Access Allowed	Access Action
__f__ Accessor can read a file, read from a disk, or allocate a device	a. Execute
__d__ Accessor can read or write a file	b. None
__a__ Accessor can execute an image file, and look up entries in a directory without using wildcard characters	c. Control
__c__ Accessor has all privileges of the objects actual owner	d. Write
__b__ Accessor has no access to the object	e. Delete
	f. Read

9. Match the following options to the system action(s) they indicate.

System Action	Option
__c__ When copying an ACL from one version of a file to another, the ACE is not duplicated	a. Default
__a__ An ACE is to be included in the ACL of any files created within a directory	b. Protected
__b__ An ACE will be preserved even when an attempt is made to delete the entire ACL	c. Nopropagate
	d. None

10. Supply the AUTHORIZE command that will perform the following:
    a. ADD/IDENTIFIER LUCKY
    b. ADD/IDENTIFIER/VALUE=UIC:[300,*] PROJECT_300
    c. GRANT/IDENTIFIER LUCKY DANTON
    d. SHOW/IDENTIFIER/FULL LUCKY
    e. SHOW/RIGHTS DANTON

11.

    a. CHANGE HER PASSWORD

    b. IF NOT ON TURN ON AUDITING FOR LOGFAILURES AND BREAKINS

    c. IF ON LOOK AT THE OPERATORS LOG FILE FOR LOGIN FAILURES AROUND THAT TIME

    d. LOOK AT FULL ACCOUNTING RECORDS OVER THAT PERIOD AND CHECK THE SOURCE OF LOGINS

    e. BEWARE

12. SEE PAGES 7-24 THROUGH 7-27 Guide To VAX/VMS System Security

13. From a BATCH job or as a SPAWNED/NOWAIT subprocess execute the following command:

```
$ SEARCH/OUTPUT=PWD2.LIS  WORK3:[*...]*.*;*   "SET HOST","""::"
```

APPENDIX C

NEW SECURITY FEATURES FOR VMS VERSION 5.0

o Access Control Lists on Queues

- Batch and Device (Printer, Server, Terminal) Queues
- Saved in SYS$SYSTEM:JBCSYSQUE.DAT
- Do Not Need to be Reestablished on Boot
- To Set up or Modify ACLs on Queues:
  - Use ACL Editor, or
  - SET ACL/OBJECT_TYPE=QUEUE/ACL=(ace_list) queue_name
- Generic Queue ACLs Should Match Associated Execution Queue ACLs
- Queue Protection Meanings
  - READ Access - Applies to Jobs Within the Queue - Allows User to See the Attributes of a Job
  - DELETE Access - Applies to Jobs Within the Queue - Allows User to Delete a Job
  - WRITE Access - Applies to Operations on the Queue - Allows User to Submit a Job to the Queue
  - EXECUTE Access - Applies to Operations on the Queue - Allows User to Act as Operator for the Queue - The Operator of a Queue Can Affect any Job in the Queue
  - CONTROL Access - Applies to Operations on ACL Itself, or Allows User All Privileges of Object's Owner
  - OPER Privilege Allows Execute Access to All Queues
  - OPER Privilege also Allows Establishment of Queues

- Enhanced Proxy Login Features
  - NETPROXY.DAT - New Proxy File Name
  - NETACP Maintains Copy of NETPROXY in Volatile Database
  - Updates to Permanent Database Propagated to Volatile
  - Multiple Proxy Accounts Allowed (Up to Fifteen)
    - Manager Sets Up More than One Proxy Account for a User
    - One Proxy is Default (ADD (or MODIFY) /PROXY/DEFAULT)
    - User Names Non-default Proxy Account in Access Control String
    - Example Command to use Specified Proxy Account:
      COPY NEWZEL"MYPRXY2"::AFILE.TYPE AFILE.TYPE
  - Ability to Identify Remote Users by the Following Combinations:
    - Node Name and Username (as Before)
    - Node Name and UIC (New)
  - Ability to Enable/Disable Incoming/Outgoing Proxy Access
  - NCP Commands:
    - SET EXECUTOR INCOMING PROXY ENABLED/DISABLED
    - SET EXECUTOR OUTGOING PROXY ENABLED/DISABLED
- Other Relevant DECnet Changes
  - Reverse Path Caching
  - Multiple Ethernet Links - Path Splitting
  - NCP Multiple Command Recall
  - MOM$LOG - Supports Logging of MOM's Use

o Forcing Expired Password Change

- Valid If PWD Expiration Date Specified with /PWDLIFETIME

- User Can be Forced to Change Password at Login Time if Password has Expired.

- Prevents Forgetting to Update Password Immediately

- To Enable/Disable Forced Password Change for an Account, On AUTHORIZE Command ADD or MODIFY, Specify:

  . /FLAGS=NODISFORCE_PWD_CHANGE - (To Enable)

  . /FLAGS=DISFORCE_PWD_CHANGE - (To Disable)

- If Forced Password Change is Disabled, V4.x Behaviour Occurs

- o True Highwater Marking Implemented for Some Files

  - Highwater Marking Still Enabled by Default on Initialize

  - True Highwater Marking Now Implemented, but Only for Sequential Files Opened for Exclusive Access

    - File System Keeps Track of Outer Limit of Written Space
    - Highwater Mark Updated in File Header Whenever Logical End-of-file Mark Updated (Usually on Close Operation)
    - File System Prevents User from Reading Beyond End of HW Mark

  - Files Other Than Sequential, Exclusive-Access Still Use Erase-on-Allocate

  - Prevents Disk Scavenging

EY-2403E-SA-0001

VMS System Security Seminar

Special Addendum

Processing VAX/VMS System Security Files

"Securitrieve"

Reprinted from ***The VAX Professional***
with permission from the publishers:

Professional Press
921 Bethlehem Pike
Spring House, PA 19477
(215) 542-7008

SECURITRIEVE

Using Datatrieve to Manage VMS Security

— by Al Cini

Al Cini is president of Computer Methods Corporation, a software consulting and training company based in Moorestown, New Jersey which specializes in DEC systems.

A VMS System Manager has always had a lot to worry about when it came to authorizing system users, but now with Version 4 of the Operating System things like the Rightslist and network "proxy" accounts make the job even tougher. In large-system environments or DECNET networks, keeping track of who-has-what-privilege can be a nightmare if all you've got to work with is AUTHORIZE. For those VMS system managers who need all the help they can get, help is at hand from an old friend: **Datatrieve!** This article will briefly review the data files which support VMS Version 4 security features, and outline how Datatrieve can be used to process them. For a detailed review of the new "Rightslist" and "DECNET proxy login" capabilities, refer to your **VAX/VMS Guides to System Security and Networking.**

A Little Background

Most of VMS security is managed by records in three data files:

- **SYS$SYSTEM:SYSUAF.DAT** - whenever you add a new user to your system, the AUTHORIZE utility creates a record in this file containing the user's USERNAME, initial password, UIC, quotas, and access restrictions. The macro $UAFDEF (in SYS$LIBRARY:LIB.MLB) defines the formats of these records.

- **SYS$SYSTEM:RIGHTSLIST.DAT** - AUTHORIZE's **ADD/IDENT** command creates identifier records in the Rightslist file, and the **GRANT/IDENT** and **REVOKE/IDENT** commands associate and disassociate these identifiers with UIC's. You can inspect the $KGBDEF macro for a look at its logical record format. [By the way, KGB stands for "Key Grants Block," and not the well-known Eastern-Bloc intelligence organization. For "equal time," another macro - $CIADEF - details the format of the Compound Intrusion Analysis memory data structure. Proof that at least one fairly active sense of humor was involved in coding all this.]

- **SYS$SYSTEM:NETUAF.DAT** - using AUTHORIZE to add and remove DECNET "proxy" records (**ADD/PROXY, REMOVE/PROXY**) correspondingly adds and removes records in this file. Proxy record formats can be found in the $NAFDEF macro.

A fourth file, SYS$SYSTEM:VMSMAIL.DAT, supports the enhanced MAIL utility found in VMS V4; since it has little to do with VMS security, we won't pay any attention to it here.

Good news! These files are all "vanilla" RMS data files (prolog 3 format, taking advantage of its newly-implemented multi-key capability), and are therefore legible to applications programs and Datatrieve!

Bad News! The "layouts" of the files' records are nowhere to be found except in an "encrypted" form within their associated macros. Until now, that is.

User Authorization File Record Format

For each authorized user, a record in the following format is maintained in the User Authorization File. Comments in the listing presented below relate the general purpose and structure of each field:

```
DEFINE RECORD USER_AUTHORIZATION_RECORD DESCRIPTION IS
        /*
        This is a CDDL definition of the VMS V4 User Authorization
        Record (SYSUAF.DAT), obtained from the expansion of the $UAFDEF
        macro.

        This record definition can be used in conjunction with Datatrieve
        or a high-level programming language to process information stored
        in the User Authorization File; NOTE, however, that USER
        AUTHORIZATION RECORDS SHOULD BE UPDATED BY USING THE "AUTHORIZE"
        UTILITY ONLY! The AUTHORIZE utility is smart enough to know how
        to change related structures (such as RIGHTSLIST.DAT).
        */.

USER_AUTHORIZATION_RECORD STRUCTURE.

        ! Internal stuff:
        UAF_RTYPE                       DATATYPE IS UNSIGNED BYTE.
        UAF_VERSION                     DATATYPE IS UNSIGNED BYTE.
        UAF_USRDATOFF                   DATATYPE IS UNSIGNED WORD.

        ! The username:
        UAF_USERNAME                    DATATYPE IS TEXT
                                        SIZE IS 32 CHARACTERS.

        ! The UIC, redefined as its group and member sub-fields:
        VARIANTS.
                VARIANT.
                ! The whole thing:
                UIC STRUCTURE.
                        UAF_UIC         DATATYPE IS UNSIGNED LONGWORD.
                END UIC STRUCTURE.
                END VARIANT.
                VARIANT.
                ! The parts:
                GROUP_MEMBER STRUCTURE.
                        UAF_MEM         DATATYPE IS UNSIGNED WORD.
                        UAF_GRP         DATATYPE IS UNSIGNED WORD.
                END GROUP_MEMBER STRUCTURE.
        END VARIANT.
        END VARIANTS.

        ! More internal stuff:
        UAF_SUB_ID                      DATATYPE IS UNSIGNED LONGWORD.
        UAF_PARENT_ID                   DATATYPE IS UNSIGNED QUADWORD.

        ! The account/owner:
```

```
UAF_ACCOUNT                     DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.
UAF_OWNER                       DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.

! Various defaults:
UAF_DEFDEV                      DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.
UAF_DEFDIR                      DATATYPE IS TEXT
                                SIZE IS 64 CHARACTERS.
UAF_LGICMD                      DATATYPE IS TEXT
                                SIZE IS 64 CHARACTERS.
UAF_DEFCLI                      DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.
UAF_CLITABLES                   DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.

! Internal password fields:
UAF_PWD                         DATATYPE IS UNSIGNED QUADWORD.
UAF_PWD2                        DATATYPE IS UNSIGNED QUADWORD.

! The number of failures since last successful login:
UAF_LOGFAILS                    DATATYPE IS UNSIGNED WORD.

! More internal password fields:
UAF_SALT                        DATATYPE IS UNSIGNED WORD.
UAF_ENCRYPT                     DATATYPE IS UNSIGNED BYTE.
UAF_ENCRYPT2                    DATATYPE IS UNSIGNED BYTE.
UAF_PWD_LENGTH                  DATATYPE IS UNSIGNED BYTE.
*                               DATATYPE IS TEXT
                                SIZE IS 1 CHARACTER.

! Various control and last-access dates:
UAF_EXPIRATION                  DATATYPE IS DATE.
UAF_PWD_LIFETIME                DATATYPE IS DATE.
UAF_PWD_DATE                    DATATYPE IS DATE.
UAF_PWD2_DATE                   DATATYPE IS DATE.
UAF_LASTLOGIN_I                 DATATYPE IS DATE.
UAF_LASTLOGIN_N                 DATATYPE IS DATE.

! Privilege Masks:
!
!  Each of the bits in the following paired longwords signifies the
! . presence or absence of a specific privilege in the user's auth-
!  orized and default privilege masks:
!
!  In the first word of either pair:
!
!   Bit 0  "0n"          CMKRNL
!   Bit 1  "0n"          CMEXEC
!   Bit 2  "0n"          SYSNAM
!   Bit 3  "0n"          GRPNAM
!   Bit 4  "0n"          ALLSPOOL
!   Bit 5  "0n"          DETACH
!   Bit 6  "0n"          DIAGNOSE
!   Bit 7  "0n"          LOG_IO
```

```
!    Bit 8  "On"          GROUP
!    Bit 9  "On"          NOACNT
!    Bit 10 "On"          PRMCEB
!    Bit 11 "On"          PRMMBX
!    Bit 12 "On"          PSWAPM
!    Bit 13 "On"          SETPRI
!    Bit 14 "On"          SETPRV
!    Bit 15 "On"          TMPMBX
!    Bit 16 "On"          WORLD
!    Bit 17 "On"          MOUNT
!    Bit 18 "On"          OPER
!    Bit 19 "On"          EXQUOTA
!    Bit 20 "On"          NETMBX
!    Bit 21 "On"          VOLPRO
!    Bit 22 "On"          PHY_IO
!    Bit 23 "On"          BUGCHK
!    Bit 24 "On"          PRMGBL
!    Bit 25 "On"          SYSGBL
!    Bit 26 "On"          PFNMAP
!    Bit 27 "On"          SHMEM
!    Bit 28 "On"          SYSPRV
!    Bit 29 "On"          BYPASS
!    Bit 30 "On"          SYSLCK
!    Bit 31 "On"          SHARE
!
!  In the second word of either pair:
!
!    Bit 0  "On"          UPGRADE
!    Bit 1  "On"          DOWNGRADE
!    Bit 2  "On"          GRPPRV
!    Bit 3  "On"          READALL
!    Bit 4  "On"          TMPJNL
!    Bit 5  "On"          PRMJNL
!    Bit 6  "On"          SECURITY
!    Bit 7  "On"          ACNT
!    Bit 8  "On"          ALTPRI

! Authorized privileges (specified as defined above):
UAF_PRIV1                       DATATYPE IS UNSIGNED LONGWORD.
UAF_PRIV2                       DATATYPE IS UNSIGNED LONGWORD.

! Default privileges (specified as defined above):
UAF_DEF_PRIV1                   DATATYPE IS UNSIGNED LONGWORD.
UAF_DEF_PRIV2                   DATATYPE IS UNSIGNED LONGWORD.

UAF_MIN_CLASS                   DATATYPE IS TEXT
                                SIZE IS 20 CHARACTERS.
UAF_MAX_CLASS                   DATATYPE IS TEXT
                                SIZE IS 20 CHARACTERS.

! Account flags:
!    Bit 0  "On"   DISCTLY
!    Bit 1  "On"   DEFCLI
!    Bit 2  "On"   LOCKPWD
!    Bit 3  "On"   CAPTIVE
!    Bit 4  "On"   DISACNT
!    Bit 5  "On"   DISWELCOM
```

```
!   Bit 6  "On"   DISMAIL
!   Bit 7  "On"   NOMAIL
!   Bit 8  "On"   GENPWD
!   Bit 9  "On"   PWD_EXPIRED
!   Bit 10 "On"   PWD2_EXPIRED
!   Bit 11 "On"   AUDIT
!   Bit 12 "On"   DISREPORT
!   Bit 13 "On"   DISRECONNECT
UAF_FLAGS                       DATATYPE IS UNSIGNED LONGWORD.

! Primary- and secondary-day hourly access restriction flags:
!
! Each of the following 24-bit fields (divided, as you can see
! into byte-triples) specifies allowable access hours for this
! account, in each of the five access modes.
!
! Each bit corresponds to an hour of the day (Midnight to a
! minute before 1AM is bit 0, 11:00 PM 'til a minute before
! Midnight is bit 23). When the corresponding bit is "ON",
! access is DENIED. Therefore, when these fields are all zeroes,
! access is allowed at all times.
!
! Network primary access hours:
UAF_NETWORK_ACCESS_P1           DATATYPE IS UNSIGNED BYTE.
UAF_NETWORK_ACCESS_P2           DATATYPE IS UNSIGNED BYTE.
UAF_NETWORK_ACCESS_P3           DATATYPE IS UNSIGNED BYTE.

! Network secondary access hours:
UAF_NETWORK_ACCESS_S1           DATATYPE IS UNSIGNED BYTE.
UAF_NETWORK_ACCESS_S2           DATATYPE IS UNSIGNED BYTE.
UAF_NETWORK_ACCESS_S3           DATATYPE IS UNSIGNED BYTE.

! Batch primary access hours:
UAF_BATCH_ACCESS_P1             DATATYPE IS UNSIGNED BYTE.
UAF_BATCH_ACCESS_P2             DATATYPE IS UNSIGNED BYTE.
UAF_BATCH_ACCESS_P3             DATATYPE IS UNSIGNED BYTE.

! Batch secondary access hours:
UAF_BATCH_ACCESS_S1             DATATYPE IS UNSIGNED BYTE.
UAF_BATCH_ACCESS_S2             DATATYPE IS UNSIGNED BYTE.
UAF_BATCH_ACCESS_S3             DATATYPE IS UNSIGNED BYTE.

! Local primary access hours:
UAF_LOCAL_ACCESS_P1             DATATYPE IS UNSIGNED BYTE.
UAF_LOCAL_ACCESS_P2             DATATYPE IS UNSIGNED BYTE.
UAF_LOCAL_ACCESS_P3             DATATYPE IS UNSIGNED BYTE.

! Local secondary access hours:
UAF_LOCAL_ACCESS_S1             DATATYPE IS UNSIGNED BYTE.
UAF_LOCAL_ACCESS_S2             DATATYPE IS UNSIGNED BYTE.
UAF_LOCAL_ACCESS_S3             DATATYPE IS UNSIGNED BYTE.

! Dialup primary access hours:
UAF_DIALUP_ACCESS_P1            DATATYPE IS UNSIGNED BYTE.
UAF_DIALUP_ACCESS_P2            DATATYPE IS UNSIGNED BYTE.
UAF_DIALUP_ACCESS_P3            DATATYPE IS UNSIGNED BYTE.
```

```
! Dialup secondary access hours:
UAF_DIALUP_ACCESS_S1            DATATYPE IS UNSIGNED BYTE.
UAF_DIALUP_ACCESS_S2            DATATYPE IS UNSIGNED BYTE.
UAF_DIALUP_ACCESS_S3            DATATYPE IS UNSIGNED BYTE.

! Remote primary access hours:
UAF_REMOTE_ACCESS_P1            DATATYPE IS UNSIGNED BYTE.
UAF_REMOTE_ACCESS_P2            DATATYPE IS UNSIGNED BYTE.
UAF_REMOTE_ACCESS_P3            DATATYPE IS UNSIGNED BYTE.

! Remote secondary access hours:
UAF_REMOTE_ACCESS_S1            DATATYPE IS UNSIGNED BYTE.
UAF_REMOTE_ACCESS_S2            DATATYPE IS UNSIGNED BYTE.
UAF_REMOTE_ACCESS_S3            DATATYPE IS UNSIGNED BYTE.
*                               DATATYPE IS TEXT
                                SIZE IS 12 CHARACTERS.

! Primary days flags:
!   Bit 0 "On"   -   Monday    is a SECONDARY day
!   Bit 1 "On"   -   Tuesday   is a SECONDARY day
!   Bit 2 "On"   -   Wednesday is a SECONDARY day
!   Bit 3 "On"   -   Thursday  is a SECONDARY day
!   Bit 4 "On"   -   Friday    is a SECONDARY day
!   Bit 5 "On"   -   Saturday  is a SECONDARY day
!   Bit 6 "On"   -   Sunday    is a SECONDARY day
UAF_PRIMEDAYS                   DATATYPE IS UNSIGNED BYTE.
*                               DATATYPE IS TEXT
                                SIZE IS 1 CHARACTER.

! Various account resource quotas and restrictions:
UAF_PRI                         DATATYPE IS UNSIGNED BYTE.
UAF_QUEPRI                      DATATYPE IS UNSIGNED BYTE.
UAF_MAXJOBS                     DATATYPE IS UNSIGNED WORD.
UAF_MAXACCTJOBS                 DATATYPE IS UNSIGNED WORD.
UAF_MAXDETACH                   DATATYPE IS UNSIGNED WORD.
UAF_PRCCNT                      DATATYPE IS UNSIGNED WORD.
UAF_BIOLM                       DATATYPE IS UNSIGNED WORD.
UAF_DIOLM                       DATATYPE IS UNSIGNED WORD.
UAF_TQCNT                       DATATYPE IS UNSIGNED WORD.
UAF_ASTLM                       DATATYPE IS UNSIGNED WORD.
UAF_ENQLM                       DATATYPE IS UNSIGNED WORD.
UAF_FILLM                       DATATYPE IS UNSIGNED WORD.
UAF_SHRFILLM                    DATATYPE IS UNSIGNED WORD.
UAF_WSQUOTA                     DATATYPE IS UNSIGNED LONGWORD.
UAF_DFWSCNT                     DATATYPE IS UNSIGNED LONGWORD.
UAF_WSEXTENT                    DATATYPE IS UNSIGNED LONGWORD.
UAF_PGFLQUOTA                   DATATYPE IS UNSIGNED LONGWORD.
UAF_CPUTIM                      DATATYPE IS UNSIGNED LONGWORD.
UAF_BYTLM                       DATATYPE IS UNSIGNED LONGWORD.
UAF_PBYTLM                      DATATYPE IS UNSIGNED LONGWORD.
UAF_JTQUOTA                     DATATYPE IS UNSIGNED LONGWORD.
UAF_PROXY_LIM                   DATATYPE IS UNSIGNED WORD.
UAF_PROXIES                     DATATYPE IS UNSIGNED WORD.
UAF_ACCOUNT_LIM                 DATATYPE IS UNSIGNED WORD.
UAF_ACCOUNTS                    DATATYPE IS UNSIGNED WORD.
```

```
        ! "Fill" space:
        *                                   DATATYPE IS TEXT
                                            SIZE IS 832 CHARACTERS.

    END USER_AUTHORIZATION_RECORD STRUCTURE.
    END USER_AUTHORIZATION_RECORD RECORD.
```

Note that the UAF record format has changed substantially since V3 of VMS; if you have any authorization record descriptions laying around anywhere, they'll need to be replaced.

Tired of typing? Happily, the next two record formats are much smaller than the user authorization record.

Rightslist Record Format

The Rightslist file contains three kinds of records:

- **Maintenance record** - used for internal control, this record contains special identification information and keeps the most recently assigned "base" numeric identifier value. When new identifiers are created, they are assigned values in ascending order from the "base" identifier, and the base identifier is updated accordingly. AUTHORIZE.EXE offers no direct control over the maintenance record. (The CREATE/RIGHTS command, however, can be used to create a brand new, logically "empty" Rightslist file.)

- **Identifier record** - one of these exists for each UIC and general identifier in the Rightslist. Identifier records are created using the AUTHORIZE utility's ADD/IDENT, REVOKE/IDENT, and REMOVE/IDENT commands.

- **Holder record** - holder records associate one or more general identifiers with a specific UIC identifier, and are managed using the GRANT/IDENT and REVOKE/IDENT commands.

All three Rightslist records share the same format:

```
    DEFINE RECORD RIGHTSLIST_RECORD.
        RIGHTSLIST_RECORD STRUCTURE.
            ! "KGB" stands for "KEY GRANTS BLOCK"

            ! Every identifier in the rightslist consists of a 32-bit integer
            ! numeric code, in one of two formats:
            !  * UIC-format identifiers (bit 31 is 0, viz., are positive) are associate
            !    with users and tie back to SYSUAF. UIC's are generally presented
            !    as paired 16-bit Octal numbers within brackets.
            !
            !  * General-format identifiers (bit 31 is 1, viz., are negative) are
            !    associated with applications and can be GRANTed/REVOKEd to/from
            !    UIC identifiers. General identifiers are typically presented as
            !    Hexadecimal values.
            !
            KGB_IDENTIFIER  DATATYPE IS SIGNED LONGWORD.
```

```
        ! A general identifier may or may not hold the "resource" attribute
        ! (indicated by bit 31 = 1 in the flags longword below). Since a resource
        ! identifier may own disk files, you can grant/revoke them to/from users
        ! for special disk space chargeback applications. Note that BOTH the
        ! identifier record AND the associated holder record must be flagged
        ! as resources to support this.
        KGB_ATTRIBUTES  DATATYPE IS SIGNED LONGWORD.

        ! For holder records, these fields tie back to other UIC identifiers in th
        ! rightslist:
        KGB_HOLDER1     DATATYPE IS SIGNED LONGWORD.
        KGB_HOLDER2     DATATYPE IS SIGNED LONGWORD.

        ! The numeric identifier's corresponding alphanumeric name:
        KGB_NAME        DATATYPE IS TEXT
                        SIZE IS 32 CHARACTERS.
        KGB_LEVEL       DATATYPE IS SIGNED WORD.
        *               DATATYPE IS TEXT
                        SIZE IS 2 CHARACTERS.
        KGB_SYSID1      DATATYPE IS SIGNED LONGWORD.
        KGB_SYSID2      DATATYPE IS SIGNED LONGWORD.

        ! In the maintenance record, this is the next ascending-order numeric
        ! identifier to be assigned for an ADD/IDENT command without a /VALUE=
        ! qualifier:
        KGB_NEXT_ID     DATATYPE IS SIGNED LONGWORD.
    END RIGHTSLIST_RECORD STRUCTURE.
END RIGHTSLIST_RECORD RECORD.
```

DECNET Proxy Record Format

The DECNET Proxy file, NETUAF.DAT, is created using the AUTHORIZE CREATE/PROXY command. It houses a very simple record associated with each defined proxy (ADD/PROXY and REMOVE/PROXY), its format defined as follows:

```
DEFINE RECORD PROXY_RECORD.
   PROXY_RECORD STRUCTURE

        ! The remote username consists of a nodename and username, laid out
        ! below as a single 64-byte field and two separate 32-byte fields:
        VARIANTS.
        VARIANT.
               NAF_REMNAME      DATATYPE IS TEXT
                                SIZE IS 64 CHARACTERS.
        END VARIANT.
        VARIANT.
        NAF_BREAKOUT STRUCTURE.
               NAF_REMNODE      DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.
               NAF_REMUSER      DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.
        END NAF_BREAKOUT STRUCTURE.
        END VARIANT.
        END VARIANTS.
```

```
        ! The corresponding "proxy" local username follows:
        NAF_LOCALUSER           DATATYPE IS TEXT
                                SIZE IS 32 CHARACTERS.

        ! Note that while the record includes the following flags longword and
        ! $NAFDEF defines some proxy record flags, there is no documented
        ! AUTHORIZE command syntax to manipulate them:
        NAF_FLAGS               DATATYPE IS SIGNED LONGWORD.
    END PROXY_RECORD STRUCTURE.
END PROXY_RECORD RECORD.
```

Now What?

Have you ever wanted to print a listing of all authorized users who have the BYPASS privilege? Or of all users with BYPASS who are designated as DECNET proxies? Or of everyone whose account will expire within 30 days? Or of everyone who hasn't used the system during the last month?

You can use Datatrieve to produce these reports much more conveniently than cutting and pasting output from AUTHORIZE, but first you've got a little effort to invest.

"Securitrieve" - Step 1

Take a good look at the record layouts presented above. You'll notice all sorts of fields in the user authorization record which can be used as Datatrieve selection criteria in creative ways, and you'll note logical "bridges" with the Rightslist and proxy files. For example, selected user records can be "joined" with the Rightslist over UIC, and with the proxy file over USERNAME.

Step 1 is simple: get to know the authorization record's fields by studying the Guide to Security, and become familiar with some advanced Datatrieve. If your primary job is System Manager, you may want to corner a programmer for a day or two to help you formulate some handy "canned" Datatrieve security reports, or a few "views" to simplify *ad hoc* browsing.

"Securitrieve" - Step 2

Next, use EDT to key the three record definitions into files, then use CDDL to "compile" them into the Common Data Dictionary. Its a lot of typing, but you only need to do it once, and its well worth the trouble. If you're in a hurry, omit the comments (these are the lines which are preceded by an exclamation "!" point).

"Securitrieve" - Step 2A

This step is optional, but potentially very helpful. Many of the fields in the security files are "bit-mapped" integers (e.g., the default and authorized privilege longwords in SYSUAF), or fields which are best-understood when displayed in Octal or Hexadecimal format rather than as decimal values (such as numeric general identifiers in RIGHTSLIST). Unfortunately, standard Datatrieve lacks a "bit"-managed datatype, and can't format numeric output in anything other than decimal radix. However, Datatrieve can be modified. The procedure is well-documented (you can find it in the manuals) and fairly straightforward.

The following code will add two new Datatrieve functions to your Datatrieve image: FN$FAO, which offers a hook into the generalized VMS "Formatted ASCII Output" system service, and FN$BIT_TEST, which uses a FORTRAN OTS routine to test specific bits

within an integer field. They should be inserted with the DTRFND.MAR module you'll find on your system in DTR$LIBRARY (this procedure assumes you told VMSINSTAL you wanted to keep the library components around when Datatrieve was built on your system; you'll need to go back and re-install Datatrieve if you didn't).

```
; FN$FAO - Convert longword argument to string via $FAO system service
;
; output is a string
; input is an output string descriptor,
; an input string descriptor pointing to the FAO conversion mask,
; and the longword argument to convert.

$DTR$FUN_DEF FN$FAO, LIB$SYS_FAO, 4
        $DTR$FUN_OUT_ARG        TYPE=FUN$K_STATUS
        $DTR$FUN_IN_ARG         TYPE=FUN$K_DESC,  DTYPE=DSC$K_DTYPE_T, ORDER=1
        $DTR$FUN_IN_ARG         TYPE=FUN$K_NULL
        $DTR$FUN_IN_ARG         TYPE=FUN$K_TEXT,  OUT_PUT=TRUE
        $DTR$FUN_IN_ARG         TYPE=FUN$K_VALUE, DTYPE=DSC$K_DTYPE_L, ORDER=2

$DTR$FUN_END_DEF

; FN$BIT_TEST- Test specified integer argument bit
;
; output is a string
; input is an output string descriptor,
; the longword bit number to test,
; and the longword argument whose bit is to be tested.

$DTR$FUN_DEF FN$BIT_TEST, FOR$BJTEST, 2
        $DTR$FUN_OUT_ARG        TYPE = FUN$K_VALUE, DTYPE = DSC$K_DTYPE_L
        $DTR$FUN_IN_ARG         TYPE=FUN$K_REF,     DTYPE=DSC$K_DTYPE_L, ORDER=1
        $DTR$FUN_IN_ARG         TYPE=FUN$K_REF,     DTYPE=DSC$K_DTYPE_L, ORDER=2
$DTR$FUN_END_DEF
```

Use EDT to insert these function definitions at the end of DTRFND.MAR, ahead of the following source line:

```
$DTR$FUN_FINI
```

Now follow the documented procedure for assembling DTRFND, replacing it within the Datarieve object library, and re-linking the Datarieve program image and associated shareable image library.

FN$FAO is called within Datatrieve as follows:

FN$FAO(*ARG1, ARG2*) where:

ARG1 - a standard VMS "Formatted Ascii Output" control string.

ARG2 - An integer value to be formatted accordingly.

FN$FAO returns a character string containing ARG2 formatted as specified by ARG1.

For example:

```
DTR> PRINT FN$FAO("!%O", 10)

 FN$FAO
    12

DTR> PRINT FN$FAO("!%X", 10)

 FN$FAO
    A
```

FN$BIT_TEST is called within Datatrieve as follows:

FN$BIT_TEST(*ARG1, ARG2*) where:

ARG1 - an integer argument representing a pattern of up to 31 boolean-evaluated bits.

ARG2 - an integer argument specifying which of the bits (0 to 31) in ARG1 to test.

FN$BIT_TEST returns a -1 if the bit specified by ARG2 is "on" in ARG1; 0 if "off"

For example:

```
DTR> PRINT FN$BIT_TEST(2, 0)

 FN$BIT_TEST
    0

DTR> PRINT FN$BIT_TEST(2, 1)

 FN$BIT_TEST
   -1
```

NOTE: The command procedure that re-links DTR for use under VMS is missing an important CLUSTER= directive, which causes problems when linking under Version 4 of VMS. Use EDT to insert the underlined directive as required within DTRBLD.COM before executing it to re-build Datatrieve:

```
cluster=cddshr,,,sys$share:cddshr.exe/shareable
cluster=dbmshr,,,sys$share:dbmshr.exe/shareable
cluster=rdbshr,,,sys$share:rdbshr/shareable
cluster=rdbbshr,,,sys$share:rdbbshr/shareable
cluster=rdmshr,,,sys$share:rdmshr/shareable
cluster=lbrshr,,,sys$share:lbrshr.exe/shareable
cluster=scrshr,,,sys$share:scrshr.exe/shareable
cluster=sortshr,,,sys$share:sortshr.exe/shareable   ! < ADD THIS FOR V4
cluster=vmsrtl,,,sys$share:vmsrtl.exe/shareable
```

"Securitrieve" - Step 3

Once the record definitions are entered, you can use Datatrieve to define the appropriate domains (for our purposes, let's assume everything is in the same CDD directory path):

```
$ DTR
VAX Datatrieve V3.1
DEC Query and Report System
Type HELP for help
DTR> DEFINE DOMAIN NETUAF -
CON>    USING PROXY_RECORD ON SYS$SYSTEM:NETUAF.DAT;
DTR> DEFINE DOMAIN SYSUAF -
CON>    USING USER_AUTHORIZATION_RECORD ON SYS$SYSTEM:SYSUAF.DAT;
DTR> DEFINE DOMAIN RIGHTSLIST -
CON>    USING RIGHTSLIST_RECORD ON SYS$SYSTEM:RIGHTSLIST.DAT;
DTR>
```

All of the steps provided above assume that you will be working within your CDD$DEFAULT CDD directory path.

"Securitrieve" - Step 4

You can now "ready" the defined domains and process the associated files (REMEMBER TO READY THESE DOMAINS SHARED).

```
DTR> READY SYSUAF SHARED READ
DTR> READY NETUAF SHARED READ
DTR> READY RIGHTSLIST SHARED READ
```

If any Datatrieve record length error or warning messages accompany these READY commands, review your record definitions to make sure all fields are defined with the proper data types and sizes, and check the domain definitions for typographical errors.

"Securitrieve" - Step 5

You can now use Datatrieve to print all sorts of simple and complex security reports! Here are a few examples:

```
$ DTR32
VAX Datatrieve V3.1
DEC Query and Report System
Type HELP for help
DTR> READY SYSUAF SHARED READ
DTR> READY NETUAF SHARED READ
DTR> READY RIGHTSLIST SHARED READ
DTR> ! Find all "expired" accounts:
DTR> FIND ALL SYSUAF WITH UAF_EXPIRATION NE "" AND UAF_EXPIRATION LE "TODAY"
[1 record found]
DTR> ! Print the expired user names and decimal UIC's:
DTR> PRINT ALL UAF_USERNAME, UAF_UIC OF CURRENT

          UAF                  UAF
     USERNAME                  UIC
```

```
CINI                                    4194305

DTR> ! Print the expired user names and octal-UIC-formatted UICs:
DTR> PRINT ALL UAF_USERNAME, FN$FAO("!%U",UAF_UIC) OF CURRENT

                UAF
              USERNAME                              FN$FAO

CINI                                    [100,1]

DTR> ! Find everyone who has the SETPRV privilege:
DTR> FIND ALL SYSUAF WITH FN$BIT_TEST(UAF_PRIV1, 14) = -1
[19 records found]
DTR> ! Of the SETPRV users, print the usernames of all who have at least
DTR> ! one local DECNET proxy entry:
DTR> PRINT ALL A.UAF_USERNAME OF A IN CURRENT WITH
[Looking for Boolean expression]
CON>    COUNT OF B IN NETUAF WITH B.NAF_LOCALUSER = A.UAF_USERNAME NE 0

                UAF
              USERNAME
CINI
```

A Few Cautionary Notes

Naturally, Datatrieve can as easily update these files as report from them, but BE CAREFUL about adding new records with Datatrieve! The AUTHORIZE utility "coordinates" such changes in the user authorization file with related updates to the Rightslist; Datatrieve won't "synchronize" these files automatically and you may well forget to do so manually! Use Datatrieve with caution to add or remove records in SYSUAF, and NEVER update the Rightslist except via AUTHORIZE (or by calling the appropriate Rightslist system service from within an application program).