OpenVMS Technical Journal V6

Protecting and Monitoring OpenVMS Systems

Protecting and Monitoring OpenVMS Systems

Michael Grinnell – Off-site Software Support Engineer

Gina Jones – Off-site Software Support Engineer

Overview

In the work environments of today, the risk of computer breaches from outside corporate or private firewalls are a constant threat. However, the threat to OpenVMS system security can also come from inside of these same firewalls - threats that can allow unwanted access to OpenVMS systems. Even though OpenVMS was named "Cool and Virtually Unhackable" at the 2001 DEFcon9 convention, lax security methods and irresponsible users can leave OpenVMS systems to become vulnerable to unauthorized access.

Fortunately, system administrators can enhance OpenVMS security by monitoring the integrity of critical system files, applications, and utilities without incurring costs for 3rd party products.

This article discusses two mechanisms that OpenVMS provides which can help administrators monitor file and utility access and prevent unauthorized access. OpenVMS Auditing and Access Control Entries (ACEs) are standard with OpenVMS installations. Using these utilities, system administrators can monitor and secure critical system files and resources, operations that are vital to maintaining a secure OpenVMS operating system.

User Access Security Issues in OpenVMS Environments

On computer systems there are basically two types of users: authorized and unauthorized. Authorized persons are given certain access to perform their jobs/tasks and with that access a certain amount of responsibility. On the other hand, unauthorized users are not intentionally given access to system resources and in general take no responsibility for their actions while accessing the system. Security breaches usually result from one of four types of user actions:

- User irresponsibility refers to situations where a user purposely or accidentally causes noticeable damage. A user accessing or copying a file or key to sell is an example of this type of irresponsibility.
- User probing refers to users who are authorized to access computer resources and exploit insufficiently protected parts of the system. Although some user's intentions may not be malicious in nature, theft of services is a crime. Users with more serious intent may seek confidential information, attempt embezzlement, or even destroy data by probing.
- User penetration refers to a situation where a user breaches the security controls to gain access to a system. Although OpenVMS is virtually "hack proof," inadequate security procedures can allow this most serious type of breach.
- Social engineering refers to intruders gaining access by deceiving legitimate users of the system. They may impersonate users by gaining access to their unsecured system/terminal passwords or by having a user perform actions that may compromise the security of the system.

Using Auditing to Monitor Access to Critical Files

Auditing can be described as recording security-relevant activities as they occur on the system and the review/analysis of the auditing log containing those activities.
Both successful and unsuccessful events can be recorded in the audit log file, SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL. In addition, a terminal can be designated as a security operator terminal where those same events can be displayed. Often, unsuccessful events are more useful in revealing possible security concerns than monitoring successful access.

The operating system audits the following security events by default and displays those events to the SECURITY.AUDIT$JOURNAL file or to an operator enabled terminal:

- Authorization database changes
- Intrusion attempts
- Login failures
- Use of DCL command SET AUDIT
- Events triggered by Audit or Alarm ACEs

Example of Audit Settings
The following command displays the security events currently enabled on an OpenVMS system.

```
$ show audit
System security alarms currently enabled for:
  ACL
  Authorization
  Audit:        illformed
  Breakin:      dialup,local,remote,network,detached
  Logfailure:   batch,dialup,local,remote,network,subprocess,detached

System security audits currently enabled for:
  ACL
  Authorization
  Audit:        illformed
  Breakin:      dialup,local,remote,network,detached
  Logfailure:   batch,dialup,local,remote,network,subprocess,detached
```

Additional events can be selected for auditing, such as volume mounts, elevated privilege use, and successfully logins, by using the SET AUDIT command.

Example of SET AUDIT Command
The SET AUDIT command enables auditing to capture unsuccessful and successful login information in audit file and on an OPCOM enabled terminal:

```
$ set audit/audit/alarm/enable=login=all
$ show audit
System security alarms currently enabled for:
  ACL
  Authorization
  Audit:        illformed
  Breakin:      dialup,local,remote,network,detached
  Login:        batch,dialup,local,remote,network,subprocess,detached
  Logfailure:   batch,dialup,local,remote,network,subprocess,detached

System security audits currently enabled for:
  ACL
  Authorization
  Audit:        illformed
  Breakin:      dialup,local,remote,network,detached
  Login:        batch,dialup,local,remote,network,subprocess,detached
  Logfailure:   batch,dialup,local,remote,network,subprocess,detached
```

Sample Auditing Outputs of Logins and Logfailures

Example of Unsuccessful Login

```
%%%%%%%%%%%  OPCOM   8-SEP-2004 12:38:16.07  %%%%%%%%%%%
Message from user AUDIT$SERVER on TIGRES
```

```
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 1025
Auditable event:      Remote interactive login failure
Event time:           8-SEP-2004 12:38:16.06
PID:                  000000E0
Process name:         _TNA7:
Username:             SYSTEM
Terminal name:        TNA7:, _TNA7:, Host: 192.168.1.100
Port:                 3191
Remote node id:       1677830336
Remote node fullname: 192.168.1.100
Remote username:      TELNET_C0A80164
Status:               %LOGIN-F-INVPWD, invalid password
```

Example of Successful Login

```
%%%%%%%%%%%  OPCOM   8-SEP-2004 12:39:50.52  %%%%%%%%%%%
Message from user AUDIT$SERVER on TIGRES
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 1025
Auditable event:        Remote interactive login
Event time:             8-SEP-2004 12:39:50.52
PID:                    000000E1
Process name:           _TNA8:
Username:               SYSTEM
Process owner:          [SYSTEM]
Terminal name:          _TNA8:, Host: 192.168.1.100 Port: 3192
Image name:             TIGRES$DKA0:[SYS0.SYSCOMMON.][SYSEXE]LOGINOUT.EXE
Remote node id:         1677830336
Remote node fullname:   192.168.1.100
Remote username:        TELNET_C0A80164
```

Using Access Control Entries to Limit Access to Files

UIC (User Identification Code) protection of files is often sufficient to deter users from accessing areas of the system they should not be accessing. However, there are times when UIC protection is not adequate. To provide individual customized access protection to files and objects, Access Control Entries (ACE) may be needed.

In addition to monitoring system events, such as login failures, administrators can use an ACE to monitor individual objects, such as files and print queues.

When users attempt to access a protected object, the OpenVMS operating system compares the user profile of the user process with the security profile of the object. The sequence of those checks is as follows:

1. Evaluate the Access Control List (ACL).

If the object has an ACL (Access Control List) the system attempts to match an ACE entry from that ACL list with the user's rights identifiers. If a match is found, the user is granted or denied access and further checking of the ACL ceases.
In the following example, the user TEST holds the identifier named LARRY, which is also present on the PROJECT.TXT file. In this example, the user TEST would be allowed full access to the file:

```
$ set default sys$system
$ run authorize
UAF> show test
Username: TEST
```

```
Identifier                        Value            Attributes
  LARRY                           %X8001003C

File with Identifier:
PROJECT.TXT;1      [SYSTEM]         (RWED,RWED,,)
(IDENTIFIER=LARRY,ACCESS=READ+WRITE+EXECUTE+DELETE)
```

Note: When you provide an ACE, do not forget to deny access via UIC protection; otherwise, you are defeating the purpose of the ACE.

2. Evaluate the protection code.

If the ACE does not grant access, the operating system then evaluates the protection code. The user is granted or denied access based on the UIC protection mask. The protection mask consists of the following categories: SYSTEM, OWNER, GROUP and WORLD. Within those categories, the granted access can be RWED (READ, WRITE, EXECUTE and DELETE) or any combination.

In the following example note the access on the following file:

```
LOGIN.COM;4          [SMITH]                         (RWED,RWED,RE,)
```

The SMITH account is the owner of the file. SYSTEM has RWED. The Owner of the file is SMITH, who also has RWED. The group has RE. The world has no access. Each access field is separated by a comma.

3. Checking for special privileges.

If access was not granted by an ACE or the UIC protection mask, privileges are then evaluated. Users with elevated system privileges may have the ability to access objects regardless of protection offered by an ACE or the UIC protection. The bypass privilege (BYPASS), group privilege (GRPPRV), read all privilege (READALL), or system privilege (SYSPRV) amplifies the holder's access to objects.

Note: Administrators must be careful when granting elevated privileges that may allow users access to critical objects. This can compromise system integrity.

Using Auditing to Monitor Queues and Logical Name Tables

For some object classes, such as queues and logical name tables, a user may gain access based on alternate privileges. For example, the queue object allows full access to all queues for users with the operator privilege (OPER), and the logical name table object allows access to the system table for users with the system name privilege (SYSNAM).
To monitor queues and logical name tables, you can set an alarm ACE on those objects.

Example of Queue Security Auditing

```
$ set audit/audit/alarm/enable=access=(success,failure)/class=queue
$ show audit
System security alarms currently enabled for:
  ACL
  Authorization
  Audit:         illformed
  Breakin:       dialup,local,remote,network,detached
  Logfailure:    batch,dialup,local,remote,network,subprocess,detached
  QUEUE access:
    Failure:     read,submit,manage,delete,control
    Success:     read,submit,manage,delete,control

System security audits currently enabled for:
  ACL
  Authorization
  Audit:          illformed
  Breakin:        dialup,local,remote,network,detached
  Logfailure:     batch,dialup,local,remote,network,subprocess,detached

QUEUE access:
    Failure:      read,submit,manage,delete,control
    Success:      read,submit,manage,delete,control

$ reply/enable=security
$ initialize/queue/batch  test_queue
$ submit/noprint sys$login:login.com

%%%%%%%%%%%  OPCOM  23-NOV-2004 13:19:01.05  %%%%%%%%%%%
Message from user AUDIT$SERVER on TIGRES
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 10
25
Auditable event:   Object access
Event time:        23-NOV-2004 13:19:01.05
PID:               0000008D
Source PID:        000000D8
Username:          SYSTEM
Process owner:     [SYSTEM]
Object class name: QUEUE
Object name:       SYS$BATCH
Access requested:  READ
Status:            %SYSTEM-S-NORMAL, normal successful completion

%%%%%%%%%%%  OPCOM  23-NOV-2004 13:19:01.09  %%%%%%%%%%%
Message from user AUDIT$SERVER on TIGRES
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 1025
Auditable event:   Object access
Event time:        23-NOV-2004 13:19:01.08
PID:               0000008D
Source PID:        000000D8
Username:          SYSTEM
Process owner:     [SYSTEM]
Object class name: QUEUE
Object name:       SYS$BATCH
Access requested:  SUBMIT
Status:            %SYSTEM-S-NORMAL, normal successful completion

Job LOGIN (queue SYS$BATCH, entry 634) started on SYS$BATCH

%%%%%%%%%%%  OPCOM  23-NOV-2004 13:23:52.86  %%%%%%%%%%%
Message from user AUDIT$SERVER on TIGRES
Security alarm (SECURITY) on TIGRES, system id: 1025
Auditable event:  Detached process login
Event time:       23-NOV-2004 13:23:52.85
PID:              000000E0
Username:         SYSTEM
Process owner:    [SYSTEM]
Image name:       TIGRES$DKA0:[SYS0.SYSCOMMON.][SYSEXE]LOGINOUT.EXE
```

Example of Logical Name Table Security Auditing

```
$ set audit/audit/alarm/enable=access=(success,failure)-
/class=logical_name_table
```

```
$ show audit
System security alarms currently enabled for:
  ACL
  Authorization
  Audit:          illformed
  Breakin:        dialup,local,remote,network,detached
  Logfailure:     batch,dialup,local,remote,network,subprocess,detached
  LOGICAL_NAME_TABLE access:
    Failure:      read,write,create,delete,control
    Success:      read,write,create,delete,control

System security audits currently enabled for:
  ACL
  Authorization
  Audit:         illformed
  Breakin:       dialup,local,remote,network,detached
  Logfailure:    batch,dialup,local,remote,network,subprocess,detached
  LOGICAL_NAME_TABLE access:
    Failure:     read,write,create,delete,control
    Success:     read,write,create,delete,control

$ show log/table=lnm$system_table

%%%%%%%%%%%  OPCOM  15-DEC-2004 08:54:25.55  %%%%%%%%%%%
Message from user AUDIT$SERVER on TIGRES
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 10
25
Auditable event:    Object access
Event time:         15-DEC-2004 08:54:25.53
PID:                0000000C4
Process name:       SYSTEM
Username:           SYSTEM
Process owner:      [SYSTEM]
Terminal name:      TNA3:
Object class name:  LOGICAL_NAME_TABLE
Object name:        LNM$SYSTEM_TABLE
Access requested:   READ
Status:             %SYSTEM-S-NORMAL, normal successful completion

%%%%%%%%%%%  OPCOM  15-DEC-2004 08:54:25.55  %%%%%%%%%%%
Message from user AUDIT$SERVER on TIGRES
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 10
25
Auditable event:   Object access
Event time:        15-DEC-2004 08:54:25.53
PID:               000000C4
Process name:      SYSTEM
Username:          SYSTEM
Process owner:     [SYSTEM]
Terminal name:     TNA3:
Object class name: LOGICAL_NAME_TABLE
Object name:       LNM$SYSTEM_TABLE
Access requested:  READ
Status:            %SYSTEM-S-NORMAL, normal successful completion
```

Using an Alarm ACE on Files

An alarm ACE can be used to monitor access to files and other objects.

In the following example, an ACE on the project.txt file will display the access output to the enabled OPCOM terminal and will write that same information to the SECURITY.AUDIT$JOURNAL file**.**

Note: You must include success or failure or both (s+f) on the alarm ACE.

```
$ set security/acl=(alarm=security,access=r+w+e+d+s+f) project.txt
$ set security/acl=(audit=security,access=r+w+e+d+s+f) project.txt
$ directory/security project.txt
Directory DKA0:[000000.USERS]
PROJECT.TXT;1        [TEST]           (RWED,RWED,RE,)
(ALARM=SECURITY,ACCESS=READ+WRITE+EXECUTE+DELETE+SUCCESS+FAILURE)
(AUDIT=SECURITY,ACCESS=READ+WRITE+EXECUTE+DELETE+SUCCESS+FAILURE)
```

Enabling OPCOM Terminals to Monitor Security Alarms

From a privileged account, you can enable OPCOM to receive security messages using the following command:

```
$ reply/enable=security
```

The following is the output of an OPCOM enabled terminal display when an authorized user accesses the project.txt file.

```
$ type dka0:[users]project.txt
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 1025
Auditable event:       Object access
Event time:            14-SEP-2004 11:15:47.24
PID:                   000000DF
Process name:          TEST
Username:              TEST
Process owner:         [TEST]
Terminal name:         TNA8:
Image name:            TIGRES$DKA0:[SYS0.SYSCOMMON.][SYSEXE]TYPE.EXE
Object class name:     FILE
File name:             TIGRES$DKA0:[USERS]PROJECT.TXT;1
File ID:               (8550,27,0)
Access requested:      READ
Sequence key:          0000C2CC
Status:                %SYSTEM-S-NORMAL, normal successful completion
```

Example of OPCOM Display

The following is the output of an OPCOM enabled terminal display when an unauthorized attempts to access the project.txt file.

```
$ type dka0:[users]project.txt
%TYPE-W-OPENIN, error opening DKA0:[USERS]PROJECT.TXT;1 as input -RMS-E-
PRV, insufficient privilege or file protection violation
Security alarm (SECURITY) and security audit (SECURITY) on TIGRES, system
id: 1025
Auditable event:        Object access
Event information:      file access request (IO$_ACCESS or IO$_CREATE)
Event time:             14-SEP-2004 11:17:17.74
PID:                    000000E0
Process name:           _TNA9:
Username:               GUEST
Process owner:          [GUEST]
Terminal name:          TNA9:
```

```
Image name:             TIGRES$DKA0:[SYS0.SYSCOMMON.][SYSEXE]TYPE.EXE
Object class name:      FILE
Object owner:           [TEST]
Object protection:      SYSTEM:RWED, OWNER:RWED, GROUP:RE, WORLD:
File name:              _TIGRES$DKA0:[USERS]PROJECT.TXT;1
File ID:                (8550,27,0)
Access requested:       READ
Sequence key:           0000C4EA
Status:                 %SYSTEM-F-NOPRIV, insufficient privilege or
object protection violation
```

Because AUDIT=SECURITY is enabled, you can also review the same information as above by analyzing the SECURITY.AUDIT$JOURNAL file from a privileged account.

Example of SECURITY.AUDIT$JOURNAL File
The following is an abbreviated display of the above alarms as they appear in the SECURITY.AUDIT$JOURNAL file.

```
$ set def sys$manager
$ analyze/audit/since=14-sep-2004:11:14
Date / Time            Type    Subtype    Node  Username    ID    Term
-----------------------------------------------------------------------
14-SEP-2004 11:15:47.24 ACCESS  OBJ_ACCESS  TIGRES TEST  000000DF TNA8:
14-SEP-2004 11:17:17.74 ACCESS  OBJ_ACCESS  TIGRES GUEST 000000E0 TNA9:
```

Note: Adding the /full qualifier to the above command provides the same output as seen in the OPCOM display above.

Removing ACEs, Audits, and Alarms from Objects

Use one of the following commands to remove the ACE from a file. The first command removes a single ACE. The second command removes both the audit ACE and the alarm ACE.

```
$ dir/sec project.txt
PROJECT.TXT;1        [TEST]                       (RWED,RWED,RE,)
(AUDIT=SECURITY,ACCESS=READ+WRITE+EXECUTE+DELETE+SUCCESS+FAILURE)
(ALARM=SECURITY,ACCESS=READ+WRITE+EXECUTE+DELETE+SUCCESS+FAILUR)
$ set security/acl=(alarm=security,access=r+w+e+d+s+f)/delete project.txt

$ dir/sec project.txt
PROJECT.TXT;1        [TEST]                       (RWED,RWED,RE,)
(AUDIT=SECURITY,ACCESS=READ+WRITE+EXECUTE+DELETE+SUCCESS+FAILURE)
$ set security/acl/delete=all project.txt

$ dir/sec project.txt
Directory DKA0:[000000.USERS]
PROJECT.TXT;1        [TEST]                       (RWED,RWED,RE,)
```

Example of Removing Security Alarm and Audit from Queue Object:

```
$ set audit/audit/alarm/disable=access=(success,failure)/class=queue
```

Example of Removing Security Alarm and Audit from Logical Name Table Object:

```
$ set audit/audit/alarm/disable=access=(success,failure)-
/class=logical_name_table
```

Summary

You can use the OpenVMS Auditing Utility and using Access Control Entries to monitor access to files and objects. There are other objects and system resources that can be monitored and restricted by using these two utilities. Please reference the OpenVMS Guide to System Security for additional information on using OpenVMS auditing and Access Control Lists (ACL) to help ensure the security of your OpenVMS system.

For more information

For more information about using these security methods, refer to the HP OpenVMS *Guide to System Security.*