VAM 3.1 Administration & User's Guide

September 2018

This manual provides the system manager with the procedures for installing, managing, and using the VAM family of software products.

Operating System/Version: OpenVMS Alpha V8.2 or later

OpenVMS Itanium V8.2 or later

Software Version: VMS Authentication Module 3.1

Process Software
Framingham, Massachusetts
USA

The material in this document is for informational purposes only and is subject to change without notice. It should not be construed as a commitment by Process Software. Process Software assumes no responsibility for any errors that may appear in this document.

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Third-party software may be included in your distribution of VMS Authentication Module, and subject to their software license agreements. See www.process.com/products/vam/3rdparty.html for complete information.

If the examples of URLs, domain names, internet addresses, and web sites we use in this documentation reflect any that actually exist, it is not intentional and should not to be considered an endorsement, approval, or recommendation of the actual site, or any products or services located at any such site by Process Software. Any resemblance or duplication is strictly coincidental.

Preface

About VMS Authentication Module

The VMS Authentication Module (VAM) provides users of OpenVMS systems controlled access to both user-written applications and the OpenVMS system overall using LDAP and/or RADIUS. It can be incorporated into an OpenVMS-based platform in two ways:

- Via an API that the user incorporates into a specific application to control access to that application.
- On a system-wide basis via use of the LGI callouts for OpenVMS `LOGINOUT.EXE`
- On a system-wide basis via the use of the OpenVMS ACME (Authentication and Credential Management Extension) interface.

Chapters three through six describe the two mechanisms and how they are implemented.

Introducing This Guide

This guide describes the VMS Authentication Module (VAM) software. It covers the following topics: software installation, configuration, and server monitoring and control.

What You Need to Know Beforehand

Before using VAM, you should be familiar with:

- Computer networks in general
- OpenVMS operating system and file system
- The TCP/IP stack (MultiNet, TCPware, or HP's OpenVMS TCP/IP software) you're using

How This Guide Is Organized

This guide has the following contents:

- Chapter 1, *Before You Begin*, explains what you need to prepare for an installation.

- Chapter 2, *Installing and Configuring VAM*, provides a step-by-step procedure for executing the software installation and configuring general VAM options.
- Chapter 3, *Using LDAP and VAM*, explains how to configure VAM for using LDAP authentication.
- Chapter 4, *Using RADIUS and VAM*, explains how to configure VAM for using RADIUS authentication.
- Chapter 5, *Using LOCALUAF and VAM*, explains how to configure VAM for using the local UAF file for authentication.
- Chapter 7, *Using the VAM API*, describes how to integrate the VAM API into a user-written application.

Accessing the VAM Public Mailing List

Process Software maintains two public mailing lists for VAM customers.

The Info-VAM mailing list is a forum for discussion among VAM system managers and programmers. Questions and problems regarding VAM can be posted for a response by any of the subscribers. To subscribe to Info-VAM, send a mail message with the word `SUBSCRIBE` in the body to `Info-VAM-request@lists.process.com`

The VAM-Announce mailing list is a one-way communication (from Process Software to you) used for the posting of announcements relating to VAM (patch releases, product releases, etc.). To subscribe to VAM-Announce, send a mail message with the word `SUBSCRIBE` in the body to `VAM-Announce-request@lists.process.com`

Obtaining Customer Support

You can use the following customer support services for information and help about VAM and other Process Software products if you subscribe to our Product Support Services. (If you bought VAM products through an authorized Process Software reseller, contact your reseller for technical support.) Contact Technical Support directly using the following methods:

Electronic Mail

E-mail relays your question to us quickly and allows us to respond, as soon as we have information for you. Send e-mail to `support@process.com`. Be sure to include your:

- Name
- Telephone number
- Company name

- Maintenance agreement number
- Product name and version number
- Operating system version number
- A detailed problem description

Telephone

If calling within the United States or Canada, call Process Software Technical Support toll-free at (800) 394-8700. If calling from outside the United States or Canada, dial +1 (508) 628-5074. Please be ready to provide your name, company name, maintenance contract number, and telephone number.

Web

There is a variety of useful technical information available on our web site, www.process.com

Conventions Used

Convention	Meaning
host	Any computer system on the network. The local host is your computer. A remote host is any other computer.
`monospaced type`	System output or user input. User input is **`in reversed bold type.`** Example: `Is this configuration correct?` **`YES`** Monospaced type also indicates user input where the case of the entry should be preserved.
italic type	Variable value in commands and examples. For example, *username* indicates that you must substitute your actual username. Italic text also identifies documentation references.

[*directory*]	Directory name in an OpenVMS file specification. Include the brackets in the specification.
[*optional-text*]	(Italicized text and square brackets) Enclosed information is optional. Do not include the brackets when entering the information. Example: `START/IP line address [info]` This command indicates that the *info* parameter is optional.
{value | value}	Denotes that you should use only one of the given values. Do not include the braces or vertical bars when entering the value.
Note	Information that follows is particularly noteworthy.
Caution	Information that follows is critical in preventing a system interruption or security breach.
key	Press the specified key on your keyboard.
Ctrl+key	Press the control key and the other specified key simultaneously.
Return	Press the Return or Enter key on your keyboard.

1. Before You Begin

Introduction

This chapter introduces you to and prepares you for the VMS Authentication Module (VAM) product installation, configuration, startup, and testing. It is for the OpenVMS system manager or technician responsible for product installation and configuration.

Steps to Get VAM Up and Running

To get VAM up and working, you must perform the following steps:

1. Load the license pack.
2. Install the software. See Chapter 2, *Installing and Configuring VAM.*
3. Configure the VAM environment. See Chapter 2, *Installing and Configuring VAM.*
4. Configure the OpenVMS system to use ACME (if ACME is used). See Chapter 6, *Using VAM with ACME.*

Prepare for Installation

VAM installation involves using the `VMSINSTAL` procedure. Preparing for installation involves:

- Understanding the hardware and software requirements
- Determining if you have sufficient disk space and global pages for the installation
- Determining where to install the software

Hardware Requirements

VAM has no special hardware requirements beyond those stated in the Software Product Description for TCPware, MultiNet or TCP/IP Services.

Software Requirements

VAM supports OpenVMS VAX version 7.3 or later; OpenVMS Alpha 7.3-2 or later; OpenVMS IA64 version 8.4 or later; MultiNet version 5.5 or later, TCPware version 5.9 or later, and TCP/IP Services version 5.5 and later.

When using the VAM ACME agents (LDAP or RADIUS), only OpenVMS versions 8.3 and later are supported.

If upgrading from one major version of the operating system to a new major version (e.g., from OpenVMS AXP V7.3-2 to OpenVMS AXP V8.3), VAM must be reinstalled to ensure the correct version of the VAM software is installed.

Disk Space and Global Pages

Disk space and global page requirements are documented in the release notes.

General Requirements

Check at this point that you:

- Have OPER, SYSPRV, or BYPASS privileges
- Can log in to the system manager's account
- Are the only user logged in (recommended)
- Backed up your system disk on a known, good, current, full backup (recommended)
- Currently have MultiNet, TCPware, or TCP/IP Services running.

Where to Install VAM

Install VAM in a location depending on the following:

- Generally, on your system disk, but you can install VAM anywhere, just answer the question when it appears. This is also where you would keep your "common" files. Node-specific files should always be on your system disk.
- If the machine is in a single platform cluster, on a common disk.
- If the machine is in a mixed platform cluster, once on the Alpha system disk (or disks), once on the IA64 system disk (or disks), and once on the VAX common system disk.

2. Installing and Configuring VAM

Introduction

This chapter takes you through the VMS Authentication Manager (VAM) product installation procedure and certain post-installation tasks. It is for the OpenVMS system manager, administrator, or technician responsible for product installation.

To prepare for installation, see Chapter 1, *Before You Begin*.

> **Note:** Once you have installed VAM, you need to reinstall it after you have done a major OpenVMS upgrade.

To install VAM:

1. Load the software.
2. Run the `VMSINSTAL` procedure.
3. Install other products, if needed, and perform post-installation tasks.

Load the Software

VAM is available for download from the Process Software FTP site. Information on downloading the software will be supplied to licensed customers by Process Software.

The VAM software must be installed from the system manager's account.

If you install VAM on a VMS cluster that has a common system disk, install the software on only one node in the cluster. Be sure to configure VAM on all systems in a VMS cluster that has a common system disk, even though it only needs to be installed once.

VAM is installed by invoking VMSINSTAL, the OpenVMS installation program for layered products. VMSINSTAL prompts you for any information it needs.

Sample Installation

```
$ @sys$update:vmsinstal VAM031 dka100:

        OpenVMS  Software Product Installation Procedure V8.4

It is 26-May-2022 at 14:09.

Enter a question mark (?) at any time for help.

* Are you satisfied with the backup of your system disk [YES]? y


The following products will be processed:

  VAM V3.1

        Beginning installation of VAM V3.1 at 14:09

%VMSINSTAL-I-RESTORE, Restoring product save set A ...

                    VMS Authentication Module (R)

ALL RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES

This licensed material is the valuable property of Process Software.
Its use, duplication, or disclosure is subject to the restrictions set
forth in the License Agreement.

Other use, duplication or disclosure, unless expressly provided for in
the license agreement, is unlawful.

* What device do you want to install VMS Authentication Module on
[SYS$SYSDEVICE:]: y
* Do you want to purge files replaced by this installation [YES]? y

The installation will now proceed with no further questions.


******************************************************************

    To complete this installation, you must refer to the documentation
    and the Release Notes for post-installation instructions.
```

```
************************************************************************

%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target
directories...

        Installation of VAM V3.1 completed at 14:09

    Adding history entry in VMI$ROOT:[SYSUPD]VMSINSTAL.HISTORY

    Creating installation data file: VMI$ROOT:[SYSUPD]VAM031.VMI_DATA


        VMSINSTAL procedure done at 14:10

$
```

Installing VAM for the First Time on a Common VMScluster System Disk

No special preparation is required after installing VAM on one node of a VMScluster with a common system disk.

Installing VAM on Mixed Platform Clusters

VAM has no files which can be shared between cluster systems of different architectures.

Post-Installation Steps

The following sections describe the post-installation setup required to enable the various forms of authentication. Specific configuration of the authentication methods (e.g., LDAP and RADIUS) are covered in subsequent chapters.

For both the VAM callable module and the VAM OpenVMS `LOGINOUT` callouts, the file *`install_device`*`:[VAM]VAM_CONFIG.TEMPLATE` must be copied (if it doesn't already exist) to *`install_device`*`:[VAM]VAM_CONFIG.DAT`. This file contains the configurable options for VAM, and may be edited as needed by the system manager.

> **Note:** If you are planning on configuring VAM LDAP or RADIUS to use the VMS ACME system, refer to Chapter 6, *Using VAM with ACME* for additional required steps.

Post-Installation File Protections

The following files must have at least the following protection and ownership. Failure to have these protections will result in authentication attempts failing.

```
VAM_CONFIG.DAT    [SYSTEM]          (RWED,RWED,,)
SDCONF.REC        [SYSTEM]          (RWED,RWED,,)
```

Post-Installation Using the VAM Callable Module

To use the VAM callable module, the system manager must add the line

```
@install_device:[VAM]VAM_STARTUP
```

to the `SYSTARTUP_VMS.COM` file.

Beyond that, no further configuration on the client system is required.

The user will be responsible for using the provided VAM API to integrate VAM into the desired application(s).

Post-Installation Using the VAM OpenVMS LOGINOUT Callouts

The OpenVMS system requires further configuration to enable the `LOGINOUT` callouts.

- Edit `VAM:VAM_CONFIG.DAT` and set the appropriate configuration keywords as desired.
- The dynamic `SYSGEN` parameter `LGI_CALLOUTS` must be set to "`1`": Note that the `LGI_CALLOUTS` parameter is reset to "`0`" each time VMS is booted, so it must be reset after each system boot.
- Next, the system manager must determine which authentication methods (LDAP and/or RADIUS) users are to be required to use. See chapters 3 and 4 for information on configuring the LGI callouts for these methods.

> **Note:** Including the LGI parameter on the VAM_STARTUP command line will enable both the VAM LGI callouts and the VAM callable module.

Configuration Keywords When Using LOGINOUT Callouts

The following keywords, found in VAM:CONFIG.DAT, are used to control access using the OpenVMS LOGINOUT callouts.

LGI_AUTH_METHODS
Contains a priority-ordered list of the authentication methods to be used. For example, "LDAP,RADIUS" will cause the VAM LGI interface to attempt first LDAP and then RADIUS authentication when called.

FALLTHROUGH_TO_VMS
If set to 1, allows VAM to fall through to using normal VMS authentication if the LDAP and/or RADIUS servers are all unreachable.

PROMPT_FOR_FT_PWD
If set to 0 and when the FALLTHROUGH_TO_VMS keyword is set to 1, the password entered during the LDAP or RADIUS authentication attempt will be used to authenticate against the local VMS User Authentication File (UAF). If set to 1 (the default), the user will be prompted for a VMS password to authenticate locally using the VMS UAF.

General Logical Names

These logical names are defined on all VAM systems. They are defined in VAM:VAM_SPECIFIC_STARTUP.COM when the VAM_STARTUP command procedure is executed.

VAM
This logical points to the *install_device*:[VAM] directory.

VAM_ROOT

This logical points to *install_device*:[VAM.]. It may be used, for example, to specify the log file directory: VAM_ROOT:[LOG].

VAM_LOG
This logical points to the *install_device*[VAM.LOG] directory.

Logging Control Logicals

The following logical names are used to affect logging for the VAM software. The logicals are located in the VAM_SPECIFIC_STARTUP command procedure and are normally commented out. This logging is used to debug VAM installations, and should generally be used only when recommended by Process Software.

VAM_LOGFILE
This logical determines the location and name of the file used to log VAM transactions and errors.

VAM_CURRENT_TRACE_LEVEL
This logical determines the level of detail in the VAM log. The level is a combination of the following bit masks:

TRACE_EXECUTION (1) - traces general steps the VAM module is performing.

TRACE_EXECUTION_DEEP (2) - verbose tracking of the VAM module processing.

TRACE_INFO (4) - Tracks informational messages generated by the VAM module

TRACE_ERROR (8) - Logs errors encountered by the VAM module

3. Using LDAP and VAM

Introduction

The VMS Authentication Module (VAM) provides users of OpenVMS systems controlled access to both user-written applications and the OpenVMS system overall using LDAP. It can be incorporated into an OpenVMS-based platform in three ways:

- Via an API that the user incorporates into a specific application to control access to that application. The VAM API is described in detail in chapter 7, *Using the VAM API*.
- On a system-wide basis via use of the LGI callouts for OpenVMS `LOGINOUT.EXE`.
- On a system-wide basis via the use of a VAM agent for the OpenVMS ACME system on OpenVMS V8.3 and higher. See Chapter 6, *Using VAM with ACME* for details on configuring your system to use VAM with ACME.

SSH logins are not affected by the VAM LGI callouts.

The system console (`OPA0:`) is never required to use the LDAP LGI Callout interface, in order to prevent being locked-out of the system in the event of a network failure that prevents the VMS system from talking with the LDAP server system(s).

> **Note:** This chapter assumes the user is familiar with LDAP in general; of the specifics of the user's LDAP installation; and if using TLS/SSL, of certificates and how to obtain and use them. Due to the breadth and depth of the topics above, this chapter will not attempt to present a tutorial on those topics.

Post-Installation Steps

The following sections describe the post-installation setup required to enable the various forms of authentication.

VAM uses configuration keywords, set in the `VAM:VAM_CONFIG.DAT` file, to determine the location of the LDAP server, the filter to be used for lookups, etc. In this way, it presents the

maximum flexibility for integration into the user's existing LDAP environment. The VAM LDAP support must be configured for all three modes of operation (callable module, LGI callout and ACME).

The VAM LDAP LGI Callouts

VAM may be incorporated into the OpenVMS login mechanism to control access to the entire system. VAM provides an OpenVMS shareable image, which the system manager can incorporate, using supported OpenVMS mechanisms, into the OpenVMS `LOGINOUT` mechanism. This image uses the LDAP protocols to supplement the standard OpenVMS login processing and provides the necessary user authentication to access the system as part of the login process.

This section assumes the user has basic knowledge of how LDAP directories are constructed and work.

Sample VAM LDAP Login

The following example shows a login to a system. It assumes the configuration keyword `LDAP_PW_PROMPT` has been set to "`LDAP Password: `":

```
$ SET HOST BOSTON1

 Welcome to OpenVMS (TM) IA64 Operating System, Version V8.4

Username: johndoe
LDAP Password: ********


        Welcome to OpenVMS IA64 V8.4

    Last interactive login on Monday, 23-JAN-2022 12:04:50.21
    Last non-interactive login on Friday,  2-DEC-2022 07:33:34.74

            You have 1 new Mail message.

BOSTON1_$
```

Controlling Access to the Callout

The system manager configures the system to use the LGI callouts. This may be done in two ways:

- Define the configuration keyword `REQUIRE_LDAP`. If set, all users are required to use LDAP authentication.
- Add the rights identifier `VAM_LGI_LDAP` to the system rights database. This identifier may then be granted to those users who will be required to use LDAP authentication.

VAM LDAP Configuration Keywords

Access to LDAP via VAM requires setting several configuration options in the configuration file `VAM:VAM_CONFIG.DAT`. This section describes those keywords and their usage.

LDAP_CERT

This configuration item is used when performing encrypted LDAP sessions. It is set to the file name of the PEM-formatted base64 (x.509) certificate containing the root certification chain for the trusted certification authority (CA) that will be used to establish the bonafides of the VAM system.

ALLOW_DECNET_LOGIN

If set to a non-zero value, DECnet CTERM (`RTAnn:`) devices are required to log in using LDAP.

ALLOW_DECTERM_LOGIN

If set to a non-zero value, DECterm (`FTAnn:`) devices are required to log in using LDAP.

LDAP_ALLOW_NULL_PASSWORD

If set to one, this keyword allows the use of blank passwords when prompting for the LDAP password.

LDAP_NOPASSWORD_SYNC

If set to 1, this will prevent VAM from updating the user's password and password change data in the VMS UAF file after a successful LDAP login. By default, VAM synchronizes this information in the UAF file to ensure that LDAP and VMS passwords are kept in sync.

LDAP_PW_PROMPT

If set, this keyword defines the prompt to use when prompting for the LDAP password, if the default of `"Password: "` isn't desired.

LDAP_TIMELIMIT

This configuration item sets the maximum length of time an LDAP search will be allowed to take. The value is in seconds. If not specified, the default is 5 seconds.

LDAP_DEBUG

This keyword defines the debug level to use for the LDAP transactions. This can result in large amounts of debug information being sent to the log file, and should not normally be used or needed.

LDAP_COMMON_USERNAME

Defines the common username for a "many-to-one" mapping of LDAP usernames to a single VMS username. This is applicable to LGI callout sessions only.

Configuring VAM LDAP Server Search Criteria

VAM provides the ability to perform multiple searches on multiple LDAP servers. This is provided through the use of *stanzas*, which consist of an `LDAP_SERVER` section which describes a specific server (e.g., the server name and port), followed by one or more `LDAP_SEARCH` sections that describe the individual searches to be performed on that server.

Specifying Servers Using the LDAP_SERVER Keywords

The following configuration keywords are used to configure access to an LDAP server. These keywords are set in the file `VAM:VAM_CONFIG.DAT`.

LDAP_SERVER ***URI***

This is the fully-qualified domain name of the LDAP server to be used in Uniform Resource Locator (URI) format. If prefaced by `ldap`, the URI indicates an unencrypted session will be done via port 389. If prefaced by `ldaps`, the URI indicates an encrypted session will be done via port 636. The port may also be explicitly specified in the URI.

For example:

```
ldap_server    ldaps://ldap.example.com:636/
```

Defines a server called ldap.example.com. Port 636 will be used to communicate to the server, and the session will be encrypted.

LDAP_USE_TLS

If your LDAP server supports LDAPS (LDAP-over-TLS), setting the value of this keyword to 1 will instruct VAM to attempt to use LDAPS for user authentication. If an LDAPS connection cannot be established, a standard LDAP connection will be used to authenticate the user.

Setting the value of this keyword to 2 will force an LDAPS connection. If an LDAPS connection cannot be established, the user will receive an error and will not be able to log in.

The value may never be used when using the LDAPS form of the URI for a server to specify that the session should be encrypted.

Specifying Searches Using the LDAP_SEARCH Keywords

The following configuration keywords are used to configure searches on an LDAP server within the configuration stanza for that server. These keywords are set in the file `VAM:VAM_CONFIG.DAT`.

LDAP_AUTH_FILTER

Specifies the LDAP search filter used to find the directory entry for a user who is authenticating to the web user interface.

Both `LDAP_BASE_DN` and `LDAP_AUTH_FILTER` allow the following expansion tags to be used in their values:

Tag	Description
`%u`	The user's login name
`%d`	The user's login domain

For example, a site might set the values of `LDAP_BASE_DN` and `LDAP_AUTH_FILTER` as:

```
ldap_base_dn            o=%d
ldap_auth_filter        (&(objectclass=person)(uid=%u))
```

If a user logged in as bob@example.com, the values of these configuration variables would be expanded to:

```
ldap_base_dn:     o=example.com
ldap_auth_filter: (&(objectclass=person)(uid=bob))
```

LDAP_AUTH_SERVER

Specifies the name of the LDAP host to search for authentication information. There is no default value.

LDAP_BASE_DN

Specifies the entry in the LDAP directory under which searches occur (sometimes also known as the search base). Consult your LDAP server's documentation set for more information specific to your implementation.

LDAP_BASE_DN supports the same tag expansions as LDAP_AUTH_FILTER.

LDAP_SEARCHACCT_DN
VAM must query the LDAP server to find the Distinguished Name of the user attempting to log in before the user can be authenticated. By default, this initial query will be done anonymously. Some directory servers (notably Microsoft's Active Directory) do not allow anonymous queries.

LDAP_SEARCHACCT_DN
Specifies the Distinguished Name of a user with search privileges on the directory server that VAM will connect as. By default, the value is NULL which indicates an anonymous login.

LDAP_SEARCHACCT_PASSWORD
Specifies the password for the search user whose Distinguished Name is specified in. LDAP_SEARCHACCT_DN. By default, the value is NULL which indicates an anonymous login.

Fetching User Attributes

VAM provides the ability to fetch a list of named attributes for a user that are stored in an LDAP directory. The search for attributes is performed on the same server on which the user has been successfully authenticated.

The form of the attribute information returned depends on the VAM interface being used. When using the VMS LOGINOUT callouts, the information will be returned as a series of logical names created in the process's job logical name table. The form of each logical name is "VAM_ATTR_*attribute_name*"; for example, VAM_ATTR_logonCount would hold the logonCount attribute that was fetched for a user.

When using the VAM API, the user specifies the UserAttributes argument to the VMSAuthenticate call. This is pointer to a struct attr structure pointer. A linked list of attributes and their values is returned in the UserAttributes argument. This structure is described in the description of the VMSAuthenticate call in Chapter 7.

To configure VAM to fetch attributes, the following keywords are used in the VAM_CONFIG.DAT file:

LDAP_ATTRIBUTE

Specifies an attribute to fetch. Each LDAP_ATTRIBUTE line is of the form "attribute_name,attribute_type". Multiple attribute lines are permitted.

The *attribute_name* is case-sensitive, and must be the same case as the attribute as stored in the LDAP directory.

Permitted values for *attribute_type* are:

- ATTRIBUTE_STRING for values that are stored in the LDAP directory as character strings. The value is returned as a null-terminated string.
- ATTRIBUTE_BINARY for values that are stored in the LDAP directory as binary values. The value is returned as a decimal number represented by a null-terminated string.

For example:

```
ldap_attribute      MyNamedAttribute,attribute_string
```

will cause the character string attribute MyNamedAttribute to be fetched.

LDAP_ATTRIBUTE_BASE_DN
Specifies the entry in the LDAP directory under which the search for the LDAP attributes occurs (sometimes also known as the search base). Consult your LDAP server's documentation set for more information specific to your implementation.

LDAP_ATTRIBUTE_BASE_DN supports the same tag expansions as LDAP_BASE_DN.

LDAP_ATTRIBUTE_FILTER
Specifies the LDAP search filter used to find the attribute entry for a user who is authenticating to the web user interface.

LDAP_ATTRIBUTE_BASE_DN supports the same tag expansions as LDAP_AUTH_FILTER.

Using TLS/SSL with VAM

TLS/SSL may be used to provide secure message transfer between VAM and the LDAP server. This is recommended as LDAP transactions by default are unencrypted and may contain clear-text username/password tuples. Thus, failure to use TLS/SSL can open a network security hole.

To enable TLS/SSL support:

- The trusted root certificate chain for the CA used to sign the LDAP server's certificate must be obtained. This certificate must be a PEM-formatted base64 (X.509) file.
- The VAM_CONFIG.DAT file must be edited to set the LDAP_CERT keyword. This keyword must point to the filename of the trusted root certificate chain.
- Ensure the server URI(s) correctly use ldaps in the URI

Note that the `ldapsearch` and `openssl` utilities (supplied in the VAM distribution) may be used to help verify the certificate chain and search criteria.

Sample LDAP Configuration

The following is an excerpt from a `VAM:VAM_CONFIG.DAT` file that illustrate a sample VAM LDAP configuration.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!               LDAP Configuration Keywords
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! If the next keyword is defined, then all users will be required
! to use LDAP authentication when using the LGI$ callouts.
! This will override the checks for the LGI_LDAP
! rights identifier to determine who is required to use LDAP.
!
REQUIRE_LDAP             1
!
! The next keyword, if set to 1, will prevent VAM from updating the
! user's password and password change data in the VMS UAF file after a
! successful LDAP login.
!
LDAP_NOPASSWORD_SYNC     0
!
! Set the max time limit (in seconds) for LDAP searches.  Defaults
! to 5 seconds if not defined.
!
! The next keyword defines the prompt to use when prompting for the
! LDAP password, if the default of "Password: " isn't desired.
!
LDAP_PW_PROMPT                  "LDAP Password:
!
! The next keyword is used to define the VMS username to which
! LDAP usenames will map upon successful authentication,
! providing a "many-to-one" external username to VMS username
! mapping.
!
!LDAP_COMMON_USERNAME            johndoe"
!
! If set to one, the next keyword allows the use of blank passwords
! when prompting for the LDAP password.
!
LDAP_ALLOW_NULL_PASSWORD        0

LDAP_TIMELIMIT    10
!
! Define the name of the PEM-formatted base64 file containing the
! root certificate chain for the trusted CA for LDAP sessions
```

```
!
LDAP_CERT     MYSYS$DKA100:[CERTS]CA_ROOT_CERTS.PEM
!
! Define keywords for LDAP attributes to be fetched.  Note that
! these are case-sensitive.
!
LDAP_ATTRIBUTE logonCount,attribute_binary
LDAP_ATTRIBUTE cn,attribute_string
!
! Define the search criteria for searching for attributes.
!
LDAP_ATTRIBUTE_BASE_DN     "CN=Users,dc=example,dc=com"
LDAP_ATTRIBUTE_FILTER
"(&(objectclass=userAttrs)(sAMAccountName=%u))"
!
! The next keywords define the parameters for performing LDAP
! authentication, for both the LGI interface and the programmatic
! interface.  They should be set to values appropriate to your
location.
!
! Multiple servers may be specified.  Each server section starts with
! an "LDAP_SERVER" label, and within each server section, searches
specific
! to that server are then defined in LDAP_SEARCH sections.
!
! Note that the port portion of the URI is optional.  If not
specified,
! the port will defined to 389 for ldap and 636 for ldaps.
!!
LDAP_SERVER ldap://ldap.example.com
LDAP_SEARCH
   LDAP_AUTH_FILTER
"(&(objectclass=user)(sAMAccountName=%u))"
   LDAP_BASE_DN
“cn=Users,dc=marketing,dc=example,dc=com"
   LDAP_SEARCHACCT_DN
“cn=Admin,CN=Users,dc=marketing,dc=example,dc=com"
   LDAP_SEARCHACCT_PASSWORD    "secretpassword"
LDAP_SEARCH
   LDAP_AUTH_FILTER
"(&(objectclass=user)(sAMAccountName=%u))"
   LDAP_BASE_DN
"cn=Users,dc=accounting,dc=example,dc=com"
   LDAP_SEARCHACCT_DN
"cn=Mgr,CN=Users,dc=accounting,dc=example,dc=com"
   LDAP_SEARCHACCT_PASSWORD    “secretpassword"
LDAP_SEARCH
   LDAP_AUTH_FILTER
"(&(objectclass=user)(sAMAccountName=%u))"
   LDAP_BASE_DN                "cn=Users,dc=sales,dc=example,dc=com"
   LDAP_SEARCHACCT_DN
"cn=JohnDoe,CN=Users,dc=sales,dc=example,dc=com"
```

```
   LDAP_SEARCHACCT_PASSWORD    "secretpassword"
!
LDAP_SERVER ldaps://eng.example.com
LDAP_SEARCH
   LDAP_AUTH_FILTER
"(&(objectclass=user)(sAMAccountName=%u))"
   LDAP_BASE_DN               "cn=Users,dc=QA,dc=example,dc=com"
   LDAP_SEARCHACCT_DN
"cn=Admin,CN=Users,dc=QA,dc=example,dc=com"
   LDAP_SEARCHACCT_PASSWORD   "secretpassword"
LDAP_SEARCH
   LDAP_AUTH_FILTER
"(&(objectclass=user)(sAMAccountName=%u))"
   LDAP_BASE_DN               "cn=Users,dc=dev,dc=example,dc=com"
   LDAP_SEARCHACCT_DN
"cn=SYSMAN,CN=Users,dc=dev,dc=example,dc=com"
   LDAP_SEARCHACCT_PASSWORD   "secretpassword"
LDAP_SEARCH
   LDAP_AUTH_FILTER
"(&(objectclass=user)(sAMAccountName=%u))"
   LDAP_BASE_DN               "cn=Users,dc=techpubs,dc=example,dc=com"
   LDAP_SEARCHACCT_DN
"cn=SYSMAN,CN=Users,dc=techpubs,dc=example,dc=com"
   LDAP_SEARCHACCT_PASSWORD   "secretpassword"
```

Using a Common Username with LDAP

Using a common username with LDAP allows the mapping of some or all LDAP usernames to a single VMS username when using the LGI callouts.

The purpose of this feature is to have a common application on a VMS system where, for example, 1000 users would all use it, but it's not important to identify the users uniquely. Those 1000 users could each have an LDAP sign on, but it's neither practical nor necessary to have a dedicated VMS user account for each of them. Hence, the "many-to-one" mapping.

To use a common username:

- The common username must have `VAM_LGI_LDAP` as a rights identifier, or the configuration keyword `REQUIRE_LDAP` needs to be set to 1.
- The configuration keyword `LDAP_COMMON_USERNAME` must be set.

If any of the LDAP accounts have a VMS UAF record, that record is ignored.

Note that the configuration above would cause all LDAP usernames to use the common username. If there are accounts where you don't want this behavior, you must grant the

VAM_LGI_LDAP_UNIQUE_USERNAME rights identifier to each account that you don't want to use the common username. Consequently, each of those accounts must have a valid VMS UAF record.

VAM LDAP Support Tools

The following unsupported tools, provided in the OpenLDAP distribution, are supplied in the VAM directory. These tools are supplied as a convenience to the user and are not supported by Process Software.

Documentation for these tools may be found at www.openldap.org. The supplied tools include:

```
ldapcompare
ldapdelete
ldapmodify
ldapmodrdn
ldappasswd
ldapsearch
ldapwhoami
openssl
```

4. Using RADIUS and VAM

Introduction

The VMS Authentication Module (VAM) provides users of OpenVMS systems controlled access to both user-written applications and the OpenVMS system overall using RADIUS. It can be incorporated into an OpenVMS-based platform in three ways:

- Via an API that the user incorporates into a specific application to control access to that application. The VAM API is described in detail in Chapter 7, "*Using the VAM API"*.
- On a system-wide basis via use of the LGI callouts for OpenVMS `LOGINOUT.EXE`.
- On a system-wide basis via the use of a VAM agent for the OpenVMS ACME system on OpenVMS V8.3 and higher. See Chapter 6, *"Using VAM with ACME*" for details on configuring your system to use VAM with ACME.

SSH logins are not affected by the VAM LGI callouts.

The system console (`OPA0:`) is never required to use the RADIUS LGI Callout interface, to prevent being locked-out of the system in the event of a network failure that prevents the VMS system from talking with the RADIUS server system(s).

> **Note:** This chapter assumes the user is familiar with RADIUS in general and of the specifics of the user's RADIUS server installation. Due to the breadth and depth of how a RADIUS server may be configured, this chapter will not attempt to present a tutorial on those topics.

Post-Installation Steps

The following sections describe the post-installation setup required to enable the various forms of authentication.

VAM uses configuration keywords, set in the `VAM:VAM_CONFIG.DAT` file, to determine the location of the RADIUS server, the filter to be used for lookups, etc. In this way, it presents the

maximum flexibility for integration into the user's existing RADIUS environment. The VAM RADIUS support must be configured for all three modes of operation (callable module, LGI callout and ACME).

The VAM RADIUS LGI Callouts

VAM may be incorporated into the OpenVMS login mechanism to control access to the entire system. VAM provides an OpenVMS shareable image, which the system manager can incorporate, using supported OpenVMS mechanisms, into the OpenVMS `LOGINOUT` mechanism. This image uses the RADIUS protocol to supplement the standard OpenVMS login processing and provides the necessary user authentication to access the system as part of the login process.

Sample VAM RADIUS Login

The following example shows a login to a system. It assumes the configuration keyword `RADIUS_PW_PROMPT` has been set to "`RADIUS Password: `":

```
$ SET HOST BOSTON1

 Welcome to OpenVMS (TM) IA64 Operating System, Version V8.4

Username: johndoe
RADIUS Password: ********


        Welcome to OpenVMS IA64 V8.4

    Last interactive login on Monday, 13-AUG-2022 12:04:50.21
    Last non-interactive login on Friday,  2-DEC-2021 07:33:34.74

            You have 1 new Mail message.

BOSTON1_$
```

Controlling Access to the Callout

The system manager configures the system to use the LGI callouts. This may be done in two ways:

- Define the configuration keyword `REQUIRE_RADIUS`. If set, all users are required to use RADIUS authentication.
- Add the rights identifier `VAM_LGI_RADIUS` to the system rights database. This identifier may then be granted to those users who will be required to use RADIUS authentication.

VAM RADIUS Configuration Keywords

Access to RADIUS via VAM requires setting several configuration options in the configuration file `VAM:VAM_CONFIG.DAT`. This section describes those keywords and their usage.

ALLOW_DECNET_LOGIN
If set to a non-zero value, DECnet CTERM (`RTAnn:`) devices are required log in using RADIUS

ALLOW_DECTERM_LOGIN
If set to a non-zero value, DECterm (`FTAnn:`) devices are required log in using RADIUS

RADIUS_KEY
This keyword defines the key to use when transacting with the RADIUS server. This is case-sensitive, and must be absolutely identical to the corresponding key on the RADIUS server. For example:

```
radius_key            TopSecretKey
```

RADIUS_NOPASSWORD_SYNC
If set to 1, this will prevent VAM from updating the user's password and password change data in the VMS UAF file after a successful RADIUS login. By default, VAM synchronizes this information in the UAF file to ensure that RADIUS and VMS passwords are kept in sync.

RADIUS_PORT
This keyword defines the port on the RADIUS server system to use. It will default to 1812 if not specified.

RADIUS_PW_PROMPT
This keyword defines the prompt to use when prompting for the RADIUS password, if the default of "`Password: `" isn't desired

RADIUS_SERVER
This is the fully-qualified domain name of the RADIUS server to be used. For example:

```
radius_server    radius.example.org
```

Defines a server called `radius.example.org`

RADIUS_TIMEOUT
This configuration item sets the maximum length of time an RADIUS transaction will be allowed to take. The value is in seconds. If not specified, the default is 5 seconds.

RADIUS_COMMON_USERNAME
Defines the common username for a "many-to-one" mapping of RADIUS usernames to a single VMS username. This is applicable to LGI callout session only.

Sample RADIUS Configuration

The following is an excerpt from a `VAM:VAM_CONFIG.DAT` file that illustrate a sample VAM RADIUS configuration.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!               RADIUS Configuration Keywords
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! If the next keyword is set to 1, then all users will be required
! to use RADIUS authentication when using the LGI$ callouts.
! This will override the checks for the LGI_RADIUS
! rights identifier to determine who is required to use RADIUS.
!
REQUIRE_RADIUS          1
!
! The next keyword defines the RADIUS server to use.
!
RADIUS_SERVER           radius.example.com
!
! The next keyword defines the key to use when transacting with
! the RADIUS server.  This is case-sensitive, and must be
! absolutely identical to the corresponding key on the RADIUS
! server.
!
RADIUS_KEY              topsecret
!
! The next keyword, if set to 1, will prevent VAM from updating the
! user's password and password change data in the VMS UAF file after a
! successful RADIUS login.
!!
RADIUS_NOPASSWORD_SYNC          0

! The next keyword defines the prompt to use when prompting for the
! RADIUS password, if the default of "Password: " isn't desired.
!
RADIUS_PW_PROMPT        "RADIUS Password: "
!
! Set the max time limit (in seconds) for RADIUS responses. Defaults
```

```
! to 5 seconds if not defined.
!
RADIUS_TIMEOUT    10
!!
! The next keyword is used to define the VMS username to which
! RADIUS usenames will map upon successful authentication,
! providing a "many-to-one" external username to VMS username
! mapping.
!
!RADIUS_COMMON_USERNAME          johndoe
```

Using a Common Username with RADIUS

Using a common username with RADIUS allows the mapping of some or all RADIUS usernames to a single VMS username when using the LGI callouts.

The purpose of this feature is to have a common application on a VMS system where, for example, 1000 users would all use it, but it's not important to identify the users uniquely. Those 1000 users could each have a RADIUS sign on, but it's neither practical nor necessary to have a dedicated VMS user account for each of them. Hence, the "many-to-one" mapping.

To use a common username:

- The common username must have `VAM_LGI_RADIUS` as a rights identifier, or the configuration keyword `REQUIRE_RADIUS` needs to be set to 1.
- The configuration keyword `RADIUS_COMMON_USERNAME` must be set.

If any of the RADIUS accounts have a VMS UAF record, that record is ignored.

Note that the configuration above would cause all RADIUS usernames to use the common username. If there are accounts where you don't want this behavior, you must grant the `VAM_LGI_RADIUS_UNIQUE_USERNAME` rights identifier to each account that you don't want to use the common username. Consequently, each of those accounts must have a valid VMS UAF record.

5. Using LOCALUAF and VAM

Introduction

The VMS Authentication Module (VAM) provides users of OpenVMS systems controlled access to user-written applications via an API that the user incorporates into a specific application to control access to that application. The VAM API is described in detail in Chapter 7, "*Using the VAM API*".

> **Note:** LOCALUAF processing is not offered for the use of the LGI callouts for OpenVMS `LOGINOUT.EXE`, as this would be redundant with what is offered by OpenVMS.

Post-Installation Steps

The following sections describe the post-installation setup required to enable the various forms of authentication.

LOCALUAF authentication is supported only for using the VAM callable module. To use the VAM callable module, the system manager must add the line

```
@install_device:[VAM]VAM_STARTUP
```

to the `SYSTARTUP_VMS.COM` file.

Beyond that, no further configuration on the client system is required.

The user will be responsible for using the provided VAM API to integrate VAM into the desired application(s).

Controlling LOCALUAF Access to the Application

Some installations may have several applications protected via VAM and using LOCALAUF processing, but which they want to further restrict access to. For example, the company may

want the PAYROLL application restricted to only people from the payroll department, while the INVENTORY application might be restricted to salespeople.

VAM provides a mechanism for restricting access in VAM-enabled applications by using VMS rights identifiers. When adding the VAM interface to an application, the application programmers may add the *identifier* field to the VMSAuthenticate() function call (see Chapter 7, *Using the VAM API*, for information on calling VMSAuthenticate). VAM then attempts to match *identifier* with a rights ID in the UAF record for the username specified in the call to VMSAuthenticate. If a match is made, access is allowed; otherwise, access is denied.

If *identifier* is not specified or is blank when calling VMSAuthenticate, then *identifier* will default to VAM_UAF_ID. Therefore, the VAM_UAF_ID rights ID must be granted to all VAM users using LOCALUAF processing if *identifier* is not specified in the call to VMSAuthenticate.

For example, ABC Corporation has three VAM-enabled applications using LOCALUAF processing: payroll, inventory and personnel. User John Doe will be allowed to access only INVENTORY, while Jane Doe will be allowed to access PERSONNEL and PAYROLL. To set these accounts up, the following steps may be used:

```
$ run sys$system:authorize
UAF> add/identifier payroll
%UAF-I-RDBADDMSG, identifier PAYROLL value %X80010003 added to rights
database
UAF> add/identifier inventory
%UAF-I-RDBADDMSG, identifier INVENTORY value %X80010004 added to
rights database
UAF> add/identifier personnel
%UAF-I-RDBADDMSG, identifier PERSONNEL value %X80010005 added to
rights database
UAF> grant/identifier inventory johndoe
%UAF-I-GRANTMSG, identifier INVENTORY granted to JOHNDOE
UAF> grant/identifier payroll janedoe
%UAF-I-GRANTMSG, identifier PAYROLL granted to JANEDOE
UAF> grant/identifier personnel janedoe
%UAF-I-GRANTMSG, identifier PERSONNEL granted to JANEDOE
UAF>  Exit
%UAF-I-NOMODS, no modifications made to system authorization file
%UAF-I-NAFNOMODS, no modifications made to network proxy database
%UAF-I-RDBDONEMSG, rights database modified
$
```

Then, when adding VAM to, for example, the payroll application, the call to VMSAuthenticate would be:

```
status = VMSAuthenticate("LOCALUAF", username, 0, &IOCallback,
                         &InfoCallback, &TimeoutCallback,
                         &ScreenClearCallback, 0, "payroll", 0, 0);
```

6. Using VAM with ACME

Introduction

The VMS Authentication Module (VAM) provides users of OpenVMS V8.3 and higher the ability to perform LDAP and RADIUS authentication via the VMS ACME subsystem.

SSH logins will not use ACME.

This chapter assumes the user is familiar with ACME in general.

After Installing VAM

After installing and configuring VAM, the latest ACMELOGIN kit for VMS must be installed. This provides ACME-enabled `LOGINOUT` and `SETP0` images to use the VAM ACME image(s). These images use ACME to perform logins to the system and use the `VMS SET PASSWORD` command, respectively. To install these images:

- Download the latest ACMELDAP ECO kit from HP.
- Execute the `ZIPEXE` file to uncompress the ACMELDAP PCSI kit.
- Extract the backup file `ACME_DEV_KITS.BCK` from the PCSI file.
- Extract the ACMELOGIN kit from `ACME_DEV_KITS.BCK`
- Install the ACMELOGIN PCSI kit just extracted.

Setting up User Accounts to Use VAM ACME

User accounts that will use VAM ACME must have the following set up:

- In `VAM:VAM_CONFIG.DAT`, the proper `REQUIRE` keyword (e.g., `REQUIRE_LDAP` or `REQUIRE_RADIUS`*)* must be set up. Use of the rights list identifier in the user's UAF record (e.g., `VAM_LGI_LDAP` or `VAM_LGI_RADIUS`) isn't supported.
- Each user account that will use VAM ACME must have the `EXTAUTH` flag set in the account's UAF record.

Starting the VAM ACME Agent

A VAM ACME agent is enabled by adding the `ACME`*`protocol`* keywords to the `VAM_STARTUP.COM` procedure when it's executed to start VAM. For example:

```
$ @SYS$SYSDEVICE:[VAM]VAM_STARTUP ACMELDAP
```

or

```
$ @SYS$SYSDEVICE:[VAM]VAM_STARTUP ACMERADIUS
```

These commands will cause the following to be performed:

- The VAM ACME persona extension (`PSC_PERSONA_EXT.EXE`) will be loaded into the VMS kernel. This enables the `SET PASSWORD` processing.
- The VMS ACME server will be stopped, restarted with the privileges required to execute the VAM ACME agents, and both the default VMS agent and the LDAP or RADIUS VAM ACME agent will be loaded and enabled.

The file names for the Process-supplied agents are:

```
VMS$PSC_LDAP_DOI_ACMESHR.EXE
VMS$RADIUS_LDAP_DOI_ACMESHR.EXE
```

Displaying VAM ACME Agents

To display the loaded ACME agents, use the `SHOW SERVER ACME` command:

```
$ SHOW SERVER ACME
ACME Information on node BOSTON1  10-JUL-2022 13:54:58.30  Uptime 0
00:00:24

ACME Server id: 2  State: Processing New Requests
   Agents Loaded:        2   Active:      2
   Thread Maximum:       4   Count:       4
   Request Maximum:    252   Count:       0

ACME Agent id: 1  State: Active
   Name: "VMS"
   Image: "DISK$SYS:[VMS$COMMON.SYSLIB]VMS$VMS_ACMESHR.EXE;1"
   Identification: "VMS ACME built 27-SEP-2006"
   Information: "No requests completed since the last startup"
   Domain of Interpretation: Yes
   Execution Order:      2

ACME Agent id: 2  State: Active
   Name: "PSC_LDAP_DOI"
   Image: "DISK$SYS:[VAM]MS$PSC_LDAP_DOI_ACMESHR.EXE;7"
   Identification: "PSC_LDAP DOI"
   Information: "PSC_LDAP_DOI Agent is initialized"
```

```
   Domain of Interpretation: Yes
   Execution Order:      1
```

Restrictions using VAM ACME

Some restrictions exist when using VAM ACME. The following sections detail these restrictions.

Multiple Agent Support

Unlike using VAM with the LGI callouts, only a single VAM ACME agent (LDAP or RADIUS) may be loaded and active at any time.

ACME-Specific Configuration Keywords

The following keywords apply specifically to VAM ACME configurations:

`PREAUTH_RETURNS_FAILURE`

If set to 1, and when using the VAM ACME agents, controls whether the VAM LDAP and RADIUS agents return `AUTHFAILURE` when a pre-authenticated authentication (e.g., a batch job) is attempted. This defaults to 0 (continues processing, skipping the various authentication checks the VAM agents do).

Unsupported VAM Configuration Keywords

The following LDAP-related and RADIUS-related configuration keywords are not supported by VAM ACME:

- `FALLTHROUGH_TO_VMS`
- `PROMPT_FOR_FT_PWD`
- `LDAP_NOPASSWORD_SYNC`
- `LDAP_ALLOW_NULL_PASSWORD`
- `LDAP_COMMON_USERNAME`
- `RADIUS_COMMON_USERNAME`

7. Using the VAM API

Introduction

VAM provides an API for allowing user-written applications to use LDAP, RADIUS or local UAF authentication for controlling access to the application. This can be implemented by a business that uses normal operating-system access for its internal functions, but which may need further authentication for specific applications that interface to counterparts on remote systems. In this case, VAM provides an additional layer of security for access to that application.

This chapter describes how to use the VAM API when using VAM as a front-end for an application.

The VAM API

The API allows VAM to be incorporated into user-written applications to control access to those applications. The API allows authentication via LDAP, RADIUS, or via the local system UAF.

This can be used by a business that uses normal operating-system access for its internal functions, but which may need further controlled access to specific applications that interface to counterparts on remote systems. In this case, VAM provides an additional layer of security for access to that application.

The API Authentication Philosophy

The authentication process begins with the user calling the `VMSAuthenticate` function, providing the username for the user, the type of authentication to perform and callbacks necessary to carry on a further dialog if needed.

For LDAP processing, the authentication routines carry on the dialog with the LDAP server, handling all the internal processing necessary. The user must configure VAM to specify the correct LDAP server and search criteria (e.g., the LDAP filter to use) to be used for the queries.

For LOCALUAF processing, the authentication routines perform many of the same checks that the VMS LOGINOUT processing does in order to validate the user.

The authentication routines will not carry on the actual dialog with the user. The user program, by supplying the dialog callbacks, will be required to do the actual dialog, using prompts

supplied by the authentication routines. In this way, the user may tailor this to the user's specific environment (video terminal, DECwindows application, etc).

When prompting for data via the dialog callbacks, the user is responsible for disabling terminal echo prior to reading the information, and re-enabling it after reading the information, and may be responsible (depending on the type of authentication being performed) for performing edits on the input data (such as being of proper length and type).

The basic processing will be as follows:

- The user is prompted for the username within the context of the user program.
- The user program calls `VMSAuthenticate()` to initialize processing. The first parameter to this function determines the authentication mechanism to use (LDAP, RADIUS or LOCALUAF).
- `VMSAuthenticate` may use the callback routines to obtain more information from the user or to display information to the user.
- `VMSAuthenticate` will return to the original caller with a status indicating whether the user has been authenticated.

Compiling a VAM Application

When compiling a source module that will call the `VMSAuthenticate` function, include the file `VAM:VMSAUTHENTICATE.H`.

Linking A VAM Application

To link an application which uses the VAM API, include the file `VAM:[VAM]VAM_LINK.OPT`. For example:

```
$ LINK MYAPPLICATION,VAM:[VAM]VAM_LINK.OPT/OPTION
```

VAM Application Special Note

VAM-enabled programs must be installed with CMKRNL privileges. If this is not done, they will be unable to successfully parse the VAM configuration file. For example:

```
$ INSTALL ADD MYPROGRAM.EXE /PRIV=CMKRNL
```

VAM API Functions

The following sections describe each of the VAM API calls. It includes not only the `VMSAuthenticate` function, but also the callback functions that are supplied by the user.

VMSAuthenticate

The user application calls `VMSAuthenticate` to perform authentication. `VMSAuthenticate` must be supplied with an identifier that defines what type of authentication will take place and a username. A password may be supplied; however, it may be ignored. The application must provide four callbacks to interact with the user.

`VMSAuthenticate` is a synchronous function; as such, it will not return until authentication completes successfully or fails.

Format

```
int VMSAuthenticate
(
  char *AuthenticationType,
  char *Username,
  char *Password,
  int *(*IOCallback)(),
  int *(*InfoCallback)(),
  void *(*TimeoutCallback)(),
  void *(*ScreenClearCallback)(),
  int *UserData,
  char *Identifier,
  struct vam_attr **UserAttributes,
  0
);
```

Inputs

- *`AuthenticationType`* - String (null-terminated) containing the type of authentication desired. Currently, must be `LDAP`, `RADIUS`, or `LOCALUAF`.
- *`Username`* - String (null-terminated) containing the username to be checked. The username is case-sensitive.
- *`Password`* - String (null-terminated) containing the password to be checked. Ignored when *`AuthenticationType`* is `LDAP` or `RADIUS`. Required when *`AuthenticationType`* is `LOCALUAF`.
- *`IOCallback`* - Pointer to the user-defined callback to be called when a prompt/response dialog must be performed with the user.
- *`InfoCallback`* - Pointer to the user-defined callback to be called when an informational message must be displayed to the user.
- *`TimeoutCallback`* - Pointer to the user-defined callback to be called when a prompt timeout occurs.
- *`ScreenClearCallback`* - Pointer to the user-defined callback to be called when the screen is to be cleared after a prompt.
- *`UserData`* - Pointer to a user-defined data area. The contents and size of this data area are to be defined by the user, and may contain any context information desired by the user (for example, to

identify the user or terminal being authenticated). This pointer will be passed to all user-defined callback routines.

- *Identifier* - String (null-terminated) that contains the name of the application. This will be used to match a VMS rights identifier when *AuthenticationType* is LOCALUAF. If this field is not specified for LOCALUAF processing, the identifier VAM_UAF_ID is used by default.
- *UserAttributes* - The address of a struct vam_attr pointer. When attributes for a user are fetched, a pointer to a linked list of vam_attr structures will be returned in this variable.
- The final parameter (denoted by "0" above) is reserved for future use but must be specified.

The vam_attr structure is defined in the VMSAUTHENTICATE.H file, and has the following form:

```
struct vam_attr
{
  char *attribute_name;
  char *value;
  struct vam_attr *next;
};
```

The fields within this structure are:

attribute_name - pointer to a character string that will contain the name of the attribute that was specified in the ATTRIBUTE keyword in the VAM:VAM_CONFIG.DAT file.

value - pointer to a character string that contains the value fetched for attribute_name. This will be NULL if no value was fetched.

next - pointer to the next attribute structure. This will be zero if the end of the attribute chain has been reached.

Outputs

None.

Returns

SS$_NORMAL

Authentication successful.

SS$_ABORT

Authentication was aborted by the server

SS$_BADPARAM

- No username was supplied

- Authentication type was not `LDAP`, `RADIUS`, or `LOCALUAF`
- All callbacks were not supplied

`SS$_CANCEL`

Authentication was aborted by the user

`SS$_NOLICENSE`

A valid license was not loaded.

> **Note:** When performing authentication, the return status will never tell the user program (provided the arguments to the routine call were correct) why the authentication failed, only that it did fail. Providing this information to a user could provide an attacker with a clue as to what to try next.

IOCallback

This user-application-supplied routine is called when a prompt/response dialog (consisting of exactly one prompt and expecting exactly one response) must be conducted with the user. The callback will be called with the information necessary to prompt for and return the required information (for example, the prompt to use, the length characteristics of the expected response, and if the response should be echoed to the terminal screen). The callback is expected to prompt for the data and return the null-terminated data in the response field. The callback is responsible (when directed by the `EchoFlag`) for turning echo to the terminal off before prompting for the data, and turning echo back on after getting the data from the caller.

Format

```
int IOCallback
(
  char *     Prompt,
  char *     Response,
  int        MinRespLen,
  int        MaxRespLen,
  int        RespType,
  int        EchoFlag,
  int        Timeout,
  int   *    UserData
);
```

Inputs

- `Prompt` - String (null-terminated) that contains the prompt to display.
- `MinRespLen` - Minimum length of expected response.
- `MaxRespLen` - Maximum length of expected response.
- `RespType` - Type of data desired for the expected response, where 0 = numeric (0-9) and 1 = alphanumeric.
- `EchoFlag` - Set to 1 if the response should be echoed to the screen.
- `Timeout` - Time (in seconds) to display the prompt.
- `UserData` - Pointer to a user-defined data area. The contents and size of this data area are to be defined by the user, and may contain any context information desired by the user (for example, to identify the user or terminal being authenticated).

Outputs

- `Response` - character string (null-terminated) that contains the response returned by the user.

Returns

1 = successfully completed

0 = call was aborted. This will cause the authentication session to be terminated, with `VMSAuthenticate` returning a status of `SS$_CANCEL`.

InfoCallback

This user application-supplied callback routine is used when an informational message must be displayed by the user application with no response required (save for possibly an "OK" button in, for example, a DECwindows application).

Format

```
int InfoCallback(char *Prompt, int Timeout, int *UserData);
```

Inputs

- *`Prompt`* - Character string (null-terminated) that contains the prompt to display.
- *`Timeout`* - Time (in seconds) to display the prompt
- *`UserData`* - Pointer to a user-defined data area. The contents and size of this data area are to be defined by the user, and may contain any context information desired by the user (for example, to identify the user or terminal being authenticated).

Outputs

None.

Returns

1 = successfully completed

0 = call was aborted. This will cause the authentication session to be terminated, with `VMSAuthenticate()` returning a status of `SS$_CANCEL`.

TimeoutCallback

This user application-supplied callback routine is invoked when a timeout for a prompt has been exceeded. The user application is required to terminate the I/O operation that it invoked. This timer is started just prior to calling the user-supplied `IOCallback` or `InfoCallback` routines.

> **Note:** The user program must not disable AST's via the VMS `$SETAST` system service. If this is done, timeouts won't be enforced.

Format

```
void TimeoutCallback(int *UserData);
```

Inputs

`UserData` - Pointer to a user-defined data area. The contents and size of this data area are to be defined by the user, and may contain any context information desired by the user (for example, to identify the user or terminal being authenticated).

Outputs

None.

Returns

None.

ScreenClearCallback

This user application-supplied callback routine is used when the screen should be cleared subsequent to a call to `IOCallBack` or `InfoCallback`.

Format

```
int ScreenClearCallback(int *UserData);
```

Inputs

`UserData` - Pointer to a user-defined data area. The contents and size of this data area are to be defined by the user, and may contain any context information desired by the user (for example, to identify the user or terminal being authenticated).

Outputs

None.

Returns

None.

8. Using VAM with SSH

Introduction

VAM may be used with the SSH server offerings from Process Software, found in MultiNet, TCPware and SSH for OpenVMS. The VAM security modules are implemented in the SSH2 server in the form of plug-ins using keyboard-interactive authentication, and require a valid VAM license to use. The SSH client used must support keyboard-interactive authentication.

> **Note:** This chapter assumes the user is familiar with configuring the SSH offerings from Process Software.

Configuring VAM in SSH

The following sections describe the post-installation setup required to enable the various forms of authentication.

Configuring VAM

In general, VAM is configured for SSH support via the use of the `VAM_CONFIG.DAT` file. However, due to restrictions of the SSH environment, not all VAM configuration keywords are honored by SSH. These unused configuration keywords are:

- `LDAP_NO_PASSWORD_SYNC`
- `LGI_AUTH_METHODS`
- `ALLOW_DECNET_LOGIN`
- `ALLOW_DECTERM_LOGIN`
- `LDAP_COMMON_USERNAME`
- `SECURID_COMMON_USERNAME`
- `PROMPT_FOR_FT_PWD`
- `FALLTHROUGH_TO_VMS`

Configuring SSH

The `SSH2_DIR:SSHD2_CONFIG` file must be modified to enable keyboard-interactive support and the proper plugin support.

The following example illustrates enabling LDAP support:

```
AllowedAuthentications          keyboard-interactive
AuthKbdInt.Required             plugin
AuthKbdInt.Plugin               ldapplugin
```

9. Using VAM with MultiNet FTP

Introduction

When using VAM with LGI callouts and the MultiNet FTP server, the server must be configured in a specific way. This chapter documents how the server must be configured.

The following is an example of configuring MultiNet FTP server to perform authentication via VAM using the LGI callouts. Once VAM has been configured, MultiNet FTP must be configured as follows:

```
$ mult configure/server
MultiNet Server Configuration Utility V5.6
[Reading in configuration from MULTINET:SERVICES.MASTER_SERVER]
SERVER-CONFIG>select ucxqio
[The Selected SERVER entry is now UCXQIO]
SERVER-CONFIG>set flag start_aux_server
[UCXQIO flags set to <START_AUX_SERVER>]
SERVER-CONFIG>set process ucxqio
[VMS process name set to UCXQIO]
SERVER-CONFIG>write
[Writing configuration to
MULTINET_COMMON_ROOT:[MULTINET]SERVICES.MASTER_SERVER.EXE;2]
SERVER-CONFIG>restart
%RUN-S-PROC_ID, identification of created process is 20800565
SERVER-CONFIG>Ctrl+Z
[Configuration not modified, so no update needed]
$
```

Appendix A. Adding ACME to an OpenVMS 8.4+ System

The following is an example of configuring a VMS 8.4 system to perform authentication via VAM using ACME.

```
$ run VMS84A_ACMELDAP-V0400.ZIPEXE
UnZipSFX 5.42 of 14 January 2001, by Info-ZIP (ZipBugs@lists.wku.edu).
  inflating: dec-axpvms-vms84a_acmeldap-v0400--4.pcsi$compressed
  inflating: dec-axpvms-vms84a_acmeldap-v0400--4.pcsi$compressed_esw
$ product list vms84a_acmeldap
Performing product kit validation of signed kits ...
%PCSI-I-VALPASSED, validation of EXAMPLE$DKA100:[SYS0.][SYSMGR]DEC-
AXPVMS-VMS84A_AC
MELDAP-V0400--4.PCSI$COMPRESSED;1 succeeded

The following product has been selected:
    DEC AXPVMS VMS84A_ACMELDAP V4.0        Patch (remedial update)

Do you want to continue? [YES] RETURN

EXAMPLE$DKA100:[SYS0.][SYSMGR]DEC-AXPVMS-VMS84A_ACMELDAP-V0400--
4.PCSI$COMPRESSED;1
    Format:   Compressed
    Kit Size: 8002 blocks and 10544 blocks when decompressed
    Manifest: Kit indicates that a manifest file was created for it
              Associated manifest file was used to successfully
validate kit

------------------------------------------------------------------------
CONTENTS OF KIT USING RELATIVE FILE SPECIFICATION
------------------------------------------------------------------------
[000000]DEC-AXPVMS-VMS84A_ACMELDAP-V0400--4.PCSI$TLB
[SYSUPD]PCSI_PRECONFIGURE.COM
[SYS$LDR]ACME.EXE
[SYSEXE]ACME_SERVER.EXE
[SYSEXE]SETSHOSERVER.EXE
[SYSHLP]ACME_DEV_README.TXT
[SYSHLP]VMS84A_ACMELDAP-V0100.RELEASE_NOTES
[SYSHLP]VMS84A_ACMELDAP-V0200.RELEASE_NOTES
[SYSHLP]VMS84A_ACMELDAP-V0300.RELEASE_NOTES
```

```
[SYSHLP]VMS84A_ACMELDAP-V0400.RELEASE_NOTES
[SYSLIB]LDAP$SHR.EXE
[SYSLIB]VMS$VMS_ACMESHR.EXE
[SYSUPD]ACME_DEV_KITS.BCK
[SYSUPD]POST_ABORT.COM
[SYSUPD]PCSI_POSTINSTALL.COM
[000000]DEC-AXPVMS-VMS84A_ACMELDAP-V0400--4.PCSI$DESCRIPTION
------------------------------------------------------------------------
-
$ product extract file vms83a_acmeldap/select=ACME_DEV_KITS.BCK

Performing product kit validation of signed kits ...
%PCSI-I-VALPASSED, validation of EXAMPLE$DKA100:[SYS0.][SYSMGR]DEC-
AXPVMS-VMS84A_AC
MELDAP-V0400--4.PCSI$COMPRESSED;1 succeeded

The following product has been selected:
    DEC AXPVMS VMS84A_ACMELDAP V4.0        Patch (remedial update)

Do you want to continue? [YES] RETURN

Portion done: 0%...100%

$ dir ACME_DEV_KITS.BCK
Directory SYS$SYSROOT:[SYSMGR]

ACME_DEV_KITS.BCK;1

Total of 1 file.
$ backup/list ACME_DEV_KITS.BCK/save
Listing of save set(s)

Save set:          ACME_DEV_KITS.BCK
Written by:        ECOKITBLD
UIC:               [000011,017761]
Date:              26-MAY-2011 01:37:23.46
Command:           BACKUP/LOG ACME_LDAP_DOCS.BCK,DEC-AXPVMS-
VMS84A_ACMELDAP_STD-
V0103--4.PCSI$COMPRESSED,DEC-AXPVMS-VMS84A_ACMELDAP_STD-V0103--
4.PCSI$COMPRESSED
_ESW,DEC-AXPVMS-VMS84A_ACMELOGIN-V0102--
4.PCSI$COMPRESSED,PWRK$MSV1_0_ACMESHR_AL
PHA.EXE,DEC-AXPVMS-VMS84A_ACMELOGIN-V0102--4.PCSI$COMPRESSED_ESW
ACME_DEV_KITS.B
CK/SAV
Operating system:  OpenVMS Alpha version V8.4
BACKUP version:    AXP84R001
CPU ID register:   80000000
Node name:         _ALTOS::
Written on:        _DSA70:
Block size:        32256
```

```
Group size:        10
Buffer count:      880

[KIT_BUILD.PCSI.ALPHA.84.ACMELDAP.V0300.A.ACME_DEV_KITS]ACME_LDAP_DOCS
.BCK;1
        3150  18-FEB-2009 15:21
[KIT_BUILD.PCSI.ALPHA.84.ACMELDAP.V0300.A.ACME_DEV_KITS]DEC-AXPVMS-
VMS84A_ACMELDAP_STD-V0103--4.PCSI$COMPRESSED;1
        143  23-FEB-2009 17:32
[KIT_BUILD.PCSI.ALPHA.84.ACMELDAP.V0300.A.ACME_DEV_KITS]DEC-AXPVMS-
VMS84A_ACMELDAP_STD-V0103--4.PCSI$COMPRESSED_ESW;1
        18  23-FEB-2009 17:32
[KIT_BUILD.PCSI.ALPHA.84.ACMELDAP.V0300.A.ACME_DEV_KITS]DEC-AXPVMS-
VMS84A_ACMELOGIN-V0102--4.PCSI$COMPRESSED;1
        1012  25-MAY-2009 09:03
[KIT_BUILD.PCSI.ALPHA.84.ACMELDAP.V0300.A.ACME_DEV_KITS]PWRK$MSV1_0_AC
MESHR_ALPHA.EXE;1
        311  12-OCT-2005 00:19
[KIT_BUILD.PCSI.ALPHA.84.ACMELDAP.V0300.A.ACME_DEV_KITS]DEC-AXPVMS-
VMS84A_ACMELOGIN-V0102--4.PCSI$COMPRESSED_ESW;1
        18  25-MAY-2009 09:03

Total of 6 files, 4652 blocks

End of save set
$ backup/log ACME_DEV_KITS.BCK/save/select=DEC-AXPVMS-
VMS84A_ACMELOGIN-V0102--4.PCSI$COMPRESSED* *.*
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DEC-AXPVMS-
VMS84A_ACMELOGIN-V0102--4.PCSI$COMPRESSED;1
%BACKUP-S-CREATED, created SYS$SYSROOT:[SYSMGR]DEC-AXPVMS-
VMS84A_ACMELOGIN-V0102--4.PCSI$COMPRESSED_ESW;1
$ product install VMS84A_ACMELOGIN

Performing product kit validation of signed kits ...
%PCSI-I-VALPASSED, validation of EXAMPLE$DKA100:[SYS0.][SYSMGR]DEC-
AXPVMS-VMS84A_ACMELOGIN-V0102--4.PCSI$COMPRESSED;1 succeeded

The following product has been selected:
    DEC AXPVMS VMS84A_ACMELOGIN V1.2       Patch (remedial update)
Do you want to continue? [YES] RETURN

 Configuration phase starting ...

You will be asked to choose options, if any, for each selected product
and for any products that may be installed to satisfy software
dependency requirements.

Configuring DEC AXPVMS VMS84A_ACMELOGIN V1.2: ACMELOGIN Patch Kit

    COPYRIGHT 1982-2009
    Hewlett-Packard Development Company, L.P.
```

```
Recovery data will be saved which will allow you to un-install
this kit.  In the past, kit installations provided some level
of recovery capability by renaming all replaced files to
file_name.ext_OLD.  If you wish, you can continue to do this.

Note that this will triple the disk space required for
this kit - one for the installed files, once for the saved
recovery data and once for the file_name.ext_OLD files.

Do you wish to have replaced files renamed
to file_name.ext_OLD [NO] ?: RETURN

Files will not be renamed

* This product does not have any configuration options.

    << System Disk Backup >>

    This kit will make functional changes to your system.
    Before installing this kit you should make a backup
    copy of your system disk.  If you do not make a copy
    of your system disk you will not be able to restore
    your system to a pre-kit installation state.



    Do you want to continue? [YES] RETURN

Execution phase starting ...

The following product will be installed to destination:
    DEC AXPVMS VMS84A_ACMELOGIN V1.2       DISK$AXPVMS84:[VMS$COMMON.]

Portion done: 0%...40%...50%...90%...100%

The following product has been installed (and a recovery data set
created):
    DEC AXPVMS VMS84A_ACMELOGIN V1.2       Patch (maintenance update)

DEC AXPVMS VMS84A_ACMELOGIN V1.2: ACMELOGIN Patch Kit

    VMS84A_ACMELOGIN-V0102 Release notes available


    For details on ACME installation and operation, please refer to
    SYS$HELP:ACME_DEV_README.TXT.


    IMPORTANT: Post-installation tasks
```

```
    Post-installation tasks are required for cluster configurations.
If
    this system is part of a cluster configuration, you must install
the
    new LOGINOUT.EXE and SETP0.EXE images on the other systems in the
cluster.
    For each system in the cluster, issue the following commands:

          $ INSTALL REPLACE LOGINOUT.EXE
          $ INSTALL REPLACE SETP0.EXE
$
```

Appendix B. Sample Program using the VAM API

The following is a sample program using the VAM API. This program may be found in the VAM directory as VAM_AUTHORIZE.C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ssdef.h>
#include <iodef.h>
#include <starlet.h>
#include <descrip.h>
#include <lib$routines.h>
#include <libclidef.h>
#include <signal.h>

#include "vmsauthenticate.h"

int term_channel;
int qio_efn;
struct
{
  short iostat;
  short term_offset;
  short term;
  short term_size;
} iosb;

int exit_val;
void exit_handler(int);

int (*VMSAUTHENTICATE)();
int load_vamshr_shareable(void);

main()
{
  int status;
  char username[50];
  char identifier[50];
  $DESCRIPTOR(namdesc, "TT:");
  unsigned long ctrl_mask = LIB$M_CLI_CTRLT | LIB$M_CLI_CTRLY;

  status = load_vamshr_shareable();
  if (!(status & 1))
```

```
  {
    printf("\n\nCan't load VAMSHR sharelable (%x),  exiting\n\n", status);
    exit(0x10000004);
  }

  lib$disable_ctrl(&ctrl_mask, 0);

  /* open the terminal and get the event flag to use */
  sys$assign(&namdesc, &term_channel, 0, 0);
  lib$get_ef(&qio_efn);

  status = IOCallback("Username: ", username, 0, 32, ALPHA_RESPONSE, 1, 0,
0);
  if (!status)
  {
    printf("\n\nAuthentication aborted, exiting\n\n");
    exit(0x10000004);
  }

  status = VMSAUTHENTICATE("SECURID", username, 0, &IOCallback,
&InfoCallback, &TimeoutCallback,  &ScreenClearCallback, 0, identifier, 0,
0);
  sys$dassgn(term_channel);
  printf("\n");

  lib$enable_ctrl(&ctrl_mask, 0);

  /* all done */
  exit((status & 1) ? 1 : 0x10000004);
}


int IOCallback(char *Prompt, char *Response, int MinRespLen, int MaxRespLen,
               int RespType, int EchoFlag, int Timeout, int *UserData)
{
  int i;
  int status;
  unsigned int readfunc;

  /* display the prompt if needed */
  if (Prompt && strlen(Prompt))
    printf("\n%s", Prompt);

  /* Now read from the terminal */
  readfunc = IO$_READVBLK;
  if (!EchoFlag)
    readfunc |= IO$M_NOECHO;

  for (i = 0; i < NSIG; i++)
    signal(i, exit_handler);
```

```
  status = sys$qiow(qio_efn, term_channel, readfunc, &iosb, 0, 0, Response,
MaxRespLen, 0, 0, 0, 0);

  for (i = 0; i < NSIG; i++)
    signal(i, SIG_DFL);

  if (getenv("VAM_AUTHORIZE_DEBUG"))
  {
    printf("\nstatus = %d, iosb.iostat = %d, iosb.term = %d\n\n",
           status, iosb.iostat, iosb.term);
  }

  if (iosb.iostat == SS$_CONTROLY)
    exit (0x10000004);

  if (!(status & 1) || !(iosb.iostat & 1))
  {
    if ((status & 1) && (iosb.iostat == SS$_ENDOFFILE))
    {
      strcpy(Response, "\n");
      iosb.term_offset = 1;
    }
    else
    {
      printf("\n\nFailed to read response, exiting");
      return 0;
    }
  }

  /* null-terminate the input */
  Response[iosb.term_offset] = 0;

  /* all done */
  return 1;
}


int InfoCallback(char *Prompt, int Timeout, int *Userdata)
{
  printf("\n%s", Prompt);
  return 1;
}


void ScreenClearCallback(int *Userdata)
{
}


void TimeoutCallback(int *UserData)
{
  InfoCallback("Timeout exceeded", 5, 0);
```

```
}


void exit_handler(int sig)
{
  int i;

  if (getenv("VAM_AUTHORIZE_DEBUG"))
  {
    if (sig != SIGINT)
      printf("Signal %d caught\n", sig);
    else
      printf("Abort (CTRL-C or CTRL-Y) caught\n");
  }

  for (i = 0; i < NSIG; i++)
    signal(i, SIG_DFL);

  exit (0x10000004);
}

int load_vamshr_shareable(void)
{
  $DESCRIPTOR (image_d, "VAMSHR");
  $DESCRIPTOR (name_d, "VMSAUTHENTICATE");
  int status;

  lib$establish(lib$sig_to_ret);

  status = lib$find_image_symbol (&image_d, &name_d, &VMSAUTHENTICATE);
  return(status);
}
```