CHAPTER 39

Authenticate Program

Support for Version 6 Only

Version 7 and later uses the **sasauth** program that is automatically invoked to validate the server userid and password. However, information about the **authenticate** program is included here for Version 6 users.

Authenticating Userid and Password Pairs

For connections to a UNIX remote host when using the TCP/IP access method, SAS/CONNECT and SAS/SHARE use the default authentication program to verify the remote host userid and to verify that the password is correct for the specified userid.

The UNIX spawner program uses the native UNIX authentication mechanism, by default, to validate a userid and password pair. Alternatively, the user can invoke the UNIX spawner program with the -AUTHPROG option and an argument that specifies the name of the customized authentication program. See “Starting the UNIX Spawner Program” on page 479 for details about invoking the UNIX spawner with the -AUTHPROG option.

A SAS/SHARE server implicitly invokes the default authentication program when a user accesses a SAS/SHARE server that is running in secure mode. To secure a server, the server administrator sets the TCPSEC environment variable to _SECURE_ (**%let tcpsec=_secure_;**) and sets the options OAVALID and UAVALID to YES (**proc server UAVALID=YES OAVALID=YES;**).

By default, both UAVALID and OAVALID options are set to NO on all UNIX platforms. Therefore, you must explicitly set these options to YES to allow only validated client connections to the server.

Both SAS/CONNECT and SAS/SHARE users can use the sample utility program, *!sasroot* **/utilities/bin/authenticate** , which is shipped with SAS software (Release 6.11 TS020 or a subsequent release), or they can use a customized Authenticate program.

Guidelines for Writing and Storing an Authentication Program

By default, the TCP/IP access method uses an external program named **authenticate** to validate the userid and password pair. The program must take two arguments, *username* and *password*, and it must then verify that the password is correct for the specified user name. If the password is valid, the program exits with a zero return code. If the password is invalid, the program exits with a non-zero return code.

It is recommended that you write attempts, successes, and failures from the Authenticate program to a log. Also, it is recommended that you fail the authentication for any step in the process that has a problem.

After you finish testing the program, move it to the *!sasroot* **/utilities/bin** directory where SAS expects the program to be located.

Obtaining Password Information

Methods for obtaining password information vary by type of UNIX system. Many UNIX systems use conventional password files that contain the encrypted password. Other UNIX systems use a "shadow" password file. Encrypted passwords are stored in a separate file that is readable only by a user that has root privileges.

The password files and the types of UNIX systems that use them are:

conventional password file **/etc/password**
SunOS 4.1, HP-UX, AIX

shadow password file **/etc/shadow**
SVR4-compliant systems (SVR4 is an abbreviation for System V Revision 4.)

Note: Examples of SVR4-compliant systems are Solaris 2 , MIPS ABI, and Intel ABI UNIX. △

Note: The AIX system also uses shadow passwords but in a different way than SVR4-compliant systems. △

The sample programs in the *!sasroot* **/utilities/src** directory contain instructions that obtain the encrypted password from both the conventional password file and the shadow password file. See the following authentication program examples for details about setting up and running these programs.

Note: The password that you set up and the one that was used to log on to the system do not have to be the same. Any user-supplied method of password validation is allowed. △

Authentication Program Examples

The *!sasroot* **/utilities/src** directory contains documented examples of the following authentication programs:

auth.conv.c
obtains the encrypted password from the conventional password file.

auth.shadow.c
obtains the encrypted password from the shadow file.

Compiling the Authenticate Program

In most cases, you can compile the working examples with the following commands:

```
%
cd !sasroot/utilities/src

%
cc -o authenticate
 authentication-program
```

Typically, the **cc** command is the name of the C language compiler, but the command that you use on your system may be different. You do not need to set high optimization or to use an ANSI standard compiler to build the program because it already uses the standard C library functions for most of the work. *authentication-program* is either **auth.conv.c**, which uses the conventional password file **/etc/passwd**, or **auth.shadow.c**, which uses the shadow password file **/etc/shadow**.

Changing the Permissions in the Executable File

After you compile the authentication program, you must change the permissions in the executable file so that it runs with root privileges.

Example 1:

For an SVR4-compliant system that uses the **/etc/shadow** file, change the file's ownership to root. Root must have a setuid (**s**) privilege.

```
%
chown root authenticate

%
chmod +s authenticate
```

Example 2:

The standard AIX and SVR4 implementations of shadow passwords are different. The AIX system user must compile the **auth.conv.c** file and change the resulting executable to setuid root, as follows:

```
%
chown root authenticate

%
chmod ogu+s authenticate
```

Other UNIX systems may use different methods to enable programs to run with root privileges.

Testing the Authentication Program

You can perform all testing of the authentication program outside the SAS/CONNECT environment because the programs are stand-alone. The simplest way to test the programs is to check the UNIX status variable in the UNIX shell. For example, using the C shell, you might test the **authenticate** program as follows:

```
%
authenticate bass
 valid-password
%
echo $status

0
```

```
%
```

You must supply a valid password for the userid, in this case, **bass**. The password is valid because the exit status is 0.

In the following test, the password is invalid because the exit status is non-zero.

```
%
authenticate bass
 invalid-password
%
echo $status

1
%
```

After you test the program and are satisfied that it works correctly, move the program to the *!sasroot* **/utilities/bin** directory where SAS expects the program to be located.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 643.

Communications Access Methods for SAS/CONNECT and SAS/SHARE Software, Version 8

ISBN 1–58025–479–9

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999