# Shadow Password Suite

John F. Haugh II
*River Parishes Programming*
*2302 Emmett Parkway*
*Austin, TX 78727*
*jfh@rpp386.cactus.org*

## ABSTRACT

The Shadow Password Suite consists of a collection of library modules and administrative utilities which provide for enhanced system security. The Shadow Password Suite originated from the desire to increase password security by implementing the */etc/shadow* file and has since evolved into a complete suite targeted at overall security as it relates to identification and authentication. By using conditional compilation directives and a configuration file, the provided functionality may be altered to work with a wide variety of systems.

The basic library modules increase password security by removing the encrypted password data from the traditional */etc/passwd* file and placing this data in the newer */etc/shadow* file. Additional modules provide a more modularized access method to the system identification information making utilities easier to write and more reliable. Full support is provided by the library for adding, deleting, and updating entries in the system files. The */etc/group*, */etc/passwd* and */etc/shadow* files are supported as are hashed database look-aside versions of each. A fourth file, */etc/gshadow*, has been added to store the encrypted group password and provide for the concept of a group administrator. It is supported similarly to the other three files.

The administrative utilities provide a command line interface to the system identification files. The traditional user and group utilities are provided. User and group accounts may be added, deleted and updated from the command line. Each utility optionally logs these changes using the *syslog()* facility. All authentication utilities log authentication failures using the *syslog()* facility as well. Additionally, the login command records authentication failures in the file */etc/ftmp* in the same format as used by the *who* command.

## 1. Introduction

The Shadow Password Suite began as a library providing the routines required for the */etc/shadow* file by the Unix[1] System V Release 3.2 operating system. The suite has since developed into a collection of library functions and system administration tools. As the */etc/shadow* file format has changed, the suite has been updated to remain compatible. The current shadow password file format consists of the user name, encrypted password, password aging information and an account expiration date.

The *login*, *passwd*, and *su* utilities are provided to allow the user to make use of the functions provided to access the */etc/shadow* file. Work-alikes for the UNIX System V Release 4 utilities *useradd, userdel, usermod, groupadd, groupdel,* and *groupmod*, as well as the BSD utilities *chsh* and *chfn*, are also provided. The *chsh* and *chfn* utilities may be invoked with the *-s* and *-f* options, respectively, to the *passwd* utility. All of these utilities are capable of logging their actions with the *syslog()* facility and have been enhanced to take advantage of features which are unique to this suite.

The group and dialup password features are now supported with the *gpasswd* and *dpasswd* utilities. The two utilities allow the system administrator to alter these two password features. The *gpasswd* utility may be invoked indirectly by using the *-g* option to the *passwd* utility.

Password aging data may be updated with the *chage* utility. This utility provides support for changing account password aging data, which requires manual editing of the password file on older systems. This command may also be used to display the aging information for a user so that they may determine when their account or password is due to expire.

## 2. Shadow Library Functions

The Shadow Password library contains replacements for all of the standard password and group file C library functions. Additional functions provide for database-like access to the underlying files.

### 2.1 Standard Functions

The *getpwnam(), getpwuid(), getpwent(),* etc. functions are provided. The functions which access the system information files by name or ID support the use of hashed look-aside files for lookup. The user may select either the 7th Edition UNIX DBM database library or the newer NDBM database by use of conditional compilation macros.

These functions are written to support the major variants for the format of the */etc/passwd* file format. By using conditional compilation directives, the user may select support for password aging and filesystem quotas.

### 2.2 File Access and Update Functions

---

[1] UNIX is a registered trademark of AT&T.

Functions for updating the system information files have been added. These allow the programmer to update one or more records in each file in an atomic manner. The programmer begins by locking and opening the file. The programmer may then search for entries by name and will be given a pointer to a copy of the requested data. Changes to the data may then be made via an update function. When the programmer has completed all desired changes, they close the file and unlock it. If the file is unlocked without being closed first, all changes are abandoned.

### 2.2.1 Locking a File

The lock function creates a file which indicates that the given file has been opened for exclusive access. When a lock file is found by a subsequent call to the lock function, the validity of the lock file is determined to insure that a lock file has not been left by an earlier process which no longer holds the lock. The lock functions are

```
int pw_lock ();
int spw_lock ();
int gr_lock ();
int sgr_lock ();
```

A zero value is returned if the file could not be locked and a non-zero value is returned if the file was locked.

### 2.2.2 Opening a File

The open function opens the underlying file and loads all of the records in the file into allocated storage. The library verifies that the file is locked if the open mode includes write access. Each entry in the file is parsed and a structure is allocated to hold the data. The original line is also saved so that the line will not have to be regenerated if it is not altered. Lines which could not be parsed successfully are flagged and no data structure is created. Flagged lines are not used for locate, update, or delete operations. Malformed lines must be removed from the data files manually. The open functions are

```
int pw_open (int mode);
int spw_open (int mode);
int gr_open (int mode);
int sgr_open (int mode);
```

The mode parameter is either O_RDONLY or O_RDWR. This mode will be used for the system open function. A zero value is returned on failure and a non-zero value is returned on success.

### 2.2.3 Locate a File Entry

The locate function scans the list of data structures. This list includes entries that were created with the open function when the file was initially opened and entries that were added with the update function, but not entries that were removed with the delete function. A pointer to the first matching entry is returned. The locate functions are

```
const struct passwd * pw_locate (char * name);
const struct spwd * spw_locate (char * name);
const struct group * gr_locate (char * name);
const struct sgrp * sgr_locate (char * name);
```

The name parameter is the user or group name of the entry to be located. A NULL pointer will be returned if no matching entry is located. The returned pointer references the object which is stored in the internal list and must not be altered by the programmer. The programmer must copy the data from the referenced object if the data is to be altered.

### 2.2.4 Update a File Entry

The update function determines if the parameter refers to an existing file entry. If the entry exists, the copy is updated to reflect the changes. Otherwise the new entry is added to the end of the list. The update functions are

```
int pw_update (struct passwd * entry);
int spw_update (struct spwd * entry);
int gr_update (struct group * entry);
int sgr_update (struct sgrp * entry);
```

A zero value is returned on failure and a non-zero value is returned on success.

### 2.2.5 Delete a File Entry

The delete function locates the named entry and marks the entry as being deleted. This list member will be ignored for the purpose of all future locate requests and will not be written out when the file is closed. The file must have been opened for read-write access for this function to be used. The cursor will be updated to refer to the next object in the file. The delete functions are

```
int pw_remove (char * name);
int spw_remove (char * name);
int gr_remove (char * name);
int sgr_remove (char * name);
```

A zero value is returned on failure and a non-zero value is returned on success.

### 2.2.6 Locate the Next File Entry

The next function returns a pointer to the next entry in the list. A cursor is maintained which points to the last entry which was returned by the delete, locate, next, or update functions. A pointer to the next list member is returned, and the cursor is updated. In this manner, the entire file may be scanned entry by entry. When there are no remaining list entries, a NULL pointer is returned. A subsequent call to the next function after NULL is returned will return a pointer to the first object in the list. The object referenced by the returned pointer must not be modified by the programmer. The programmer must make a copy of the data if it is to be altered. The next functions are

```
const struct passwd * pw_next ();
const struct spwd * spw_next ();
const struct group * gr_next ();
const struct sgrp * sgr_next ();
```

### 2.2.7 Rewind the File

The rewind function updates the file cursor so that the cursor points to the first entry in the list. The next function will return a pointer to the first entry if it is called immediately after the file is rewound. The rewind functions are

```
int pw_rewind ();
int spw_rewind ();
int gr_rewind ();
int sgr_rewind ();
```

### 2.2.8 Close the File

The close function creates a new file containing all entries in the data structure list which are not marked as being deleted. This includes entries added with the update function and entries marked as invalid when the file was initially loaded by the open function. Allocated data structures will be freed after all of the entries have been written out. The close functions are

```
int pw_close ();
int spw_close ();
int gr_close ();
int sgr_close ();
```

A zero value is returned on failure and a non-zero value on success. A copy of the original file will be saved using the original file name with a hyphen appended as the save file name.

### 2.2.9 Unlock the File

The unlock function removes the lock file making the file available to other processes. Any changes to the file will be abandoned if the close function is not invoked prior to invoking the unlock function. Allocated data structures will be freed if they were not freed by an earlier call to the close function. The unlock functions are

```
int pw_unlock ();
int spw_unlock ();
int gr_unlock ();
int sgr_unlock ();
```

A zero value is returned on failure, and non-zero is returned on success.

## 2.3 Database Access and Update Functions

The traditional mechanism for updating the DBM files has been to use the *mkpasswd* utility to recreate the entire file after it has been altered. New functions have been added which allow the programmer to update the database files as each entry in the text file is altered. These functions allow the

programmer to add, delete, or update individual entries in the database files. The update and delete functions perform their operations immediately.

### 2.3.1 Locate a Database Entry

The various get functions will invoke the *fetch()* function to retrieve a requested entry from the database. There are no special functions specifically for this purpose. The functions which reference the DBM files are

```
struct passwd *getpwnam (char *name);
struct passwd *getpwuid (uid_t uid);
struct spwd *getspnam (char *name);
struct group *getgrnam (char *name);
struct group *getgrgid (gid_t gid);
struct sgrp *getsgnam (char *name);
```

### 2.3.2 Update a Database Entry

The update function uses the database *store()* function to add or update the supplied record in the database file. A pointer to the new record is provided to the update function. The update functions are

```
int pw_dbm_update (struct passwd *pwd);
int sp_dbm_update (struct spwd *spwd);
int gr_dbm_update (struct group *grp);
int sg_dbm_update (struct sgrp *sgrp);
```

A zero value is returned on failure. A non-zero value is returned on success.

### 2.3.3 Delete a Database Entry

The delete function uses the database *delete()* function to remove the matching record. Because the password and group files have entries which are indexed both by name and numerical identifier, the name and identifier must be present in a structure. The shadow password and shadow group files only need the name to locate the requested entry. The delete functions are

```
int pw_dbm_remove (struct passwd *entry);
int sp_dbm_remove (char *name);
int gr_dbm_remove (struct group *entry);
int sg_dbm_remove (char *name);
```

A zero value is returned on failure and a non-zero value is returned on success.

## 3. Utilities

The Shadow Password Suite contains the standard utilities as well as a number of additional utilities to support new or previously unsupported existing functionality.

### 3.1 User Account Maintenance Commands

The *chage, chfn, chsh, passwd, useradd, userdel,* and *usermod* commands use the functions which were described earlier to manage user account information.

The DBM files are updated as changes are made. Files which have entries altered will be saved as described for the close functions. The commands log the changes which were made and the user that was affected. The *syslog()* facility is used to perform this function.

An extension to the standard system authentication method allows the system administrator to define a program to perform user authentication on a per-user basis. Each user may have a special program which verifies the user's identity in a program specific manner. There are no constraints on what mechanisms the authentication program uses. The authentication method is defined to the system by use of the *useradd* and *usermod* commands.

The *passwd* supports administrator defined authentication methods. The system will invoke the defined authentication program for any account with an administrator defined method. The method will be invoked once to verify the user's identity and again to request the user enter a new authentication value.

### 3.2 Group Account Maintenance Commands

The *gpasswd, groupadd, groupdel,* and *groupmod* commands use the database functions for the group files. The DBM files are updated as changes are made. These commands function similar to the user account maintenance commands.

### 3.3 User Authentication Commands

The *login* and *su* commands verify that the named user is properly authenticated.

The *login* command makes use of the file */etc/login.defs* to indicate how various features are supported. The system administrator is able to configure support for virtually all of the command's behavior. Configurable functions include

-Dialup passwords
-Login failure logging
-Display last login time
-Display status of mailbox
-Enable login time and port access controls
-Enable root login port access controls
-Display message of the day files
-Enable setting of TERM environmental variable
-Enable non-administrator login restrictions
-Enable user to suppress login messages
-Set a default TZ environmental variable
-Set a default HZ environmental variable
-Set a default PATH environmental variable
-Set default terminal control characters
-Set default terminal group and permissions

The system administrator is able to edit the file and have the changes take effect immediately without having to recompile the system software.

The *su* command makes use of the */etc/login.defs* file to control use of the *syslog()* facility and logging to a text file. The environmental variables TZ and HZ

will also be set from the */etc/login.defs* file if the user is not preserving the current environment.

The *id* command provides the user with the current user and group identifiers.

The *dpasswd* command maintains the dialup password files. This is used by the *login* command to control access to the system when the user attempts to gain access via a dialup port.

## 3.3 Group Authentication Commands

The *newgrp* and *sg* commands verify that a user is permitted to change their current real and effective group identifier.

The *newgrp* command replaces the current shell with a shell which has the requested group identifier. The command verifies that the user has the appropriate authorization by checking the contents of the group file for memberership. If the user is not a member, the user will be prompted for a password. The group membership list and the encrypted group password is maintained with the *gpasswd* command.

The *sg* command provides the ability to execute one or more commands with a new group identifier without replacing the current shell. The *sg* command authenticates the user in a manner identical to the *newgrp* command. This mechanism allows users to more easily change between group identifiers when the system does not support the concurrent group set functionality. The user has complete access to all groups to which they are granted membership by the */etc/group* file.

The *groups* command provides the user with a list of groups to which the user has password free access. Systems which do not support concurrent group sets report the membership that is defined by the */etc/group* file, not the real or effective group identifiers.

## 3.4 Administration Commands

### 3.4.1 File Maintenance and Conversion Commands

Conversion between shadowed password files and non-shadowed files is supported with the *pwconv* and *pwunconv* commands. These commands allow the administrator to convert all of the entries in either format file and produce the other format files as output. There is some loss of information when converting from shadowed to non-shadowed files as the */etc/shadow* file contains information which is not present in the */etc/passwd* file. There is also a loss of resolution as shadowed dates are stored in units of days and non-shadowed dates are stored in units of weeks.

The DBM files may be completely reconstructed with the *mkpasswd* command. The command reads the contents of the various text files and produces DBM files as output. The use of DBM files improves system performance by reducing the access time required to locate specific entries. No changes are required to begin or end use of these files. The command is capable of creating DBM files for the following files

-/etc/passwd
-/etc/group
-/etc/shadow
-/etc/gshadow

### 3.4.2 Single-User Authentication

Access to the system in single-user or system maintenance mode may be controlled by use of the *sulogin* command. It requests the root password from the user before creating a login shell. Adding this command to the */etc/inittab* or similar file allows the system administrator to prevent access by unauthorized users when the system is restarted. An example of an entry to perform this function is

**co:Ss:respawn:/etc/sulogin /dev/console**

This entry will cause the command to be repeatedly executed until the user responds to the password prompt with end of file. The *init* command will be signalled to cause the system to enter multi-user mode after an end of file is entered.

### 3.4.3 Enforcing User Access Times

The */etc/porttime* file is used by the *login* command when login time and port access checks have been enabled. A user may login during permitted times and remain signed on after the permitted time period has expired. The *logoutd* command provides an enforcement tool for preventing users from having access during unauthorized times.

The logout daemon is started from the */etc/rc* file and runs once a minute. The */etc/utmp* file is examined and each signed on user is checked against the contents of the */etc/porttime* file to insure that the user is permitted to be signed on at the current time on the current port.

The user will be given advance warning prior to being logged out. The entire process group will be terminated. On systems where the login process ID is available, the process group will be sent a SIGHUP signal, followed shortly thereafter by a SIGTERM signal. Systems which support the *vhangup()* function will use that mechanism instead.

### 3.4.4 Controlling and Reporting Login Failures

The ability to set and report login failure limits is provided by the *faillog* command. This command allows the system administrator to set per-user limits on the number of login failures which will be permitted before an account is disabled. Users are permitted to view the failure information for their own account, but are not able to change the parameters which control when their account is disabled.

Each user account is controlled independently. The number of failures which is permitted before the account is disabled is stored the */usr/adm/faillog* file along with the terminal device where the attempt was made and the number of failures since the last successful sign on. A value of 0 indicates that an unlimited number of failures is permitted. This may be used to prevent denial of service attacks for crucial accounts.

The failure limit is set by specifying the user name and the requested limit. Or, the user name may be omitted and the change will be made for all current users. This function is restricted to the system administrator.

The current failure status may be reset per-user or for all users. This allows the system administrator to clear all indications of login failures and re-enable accounts which have become disabled.

The print function reports login failures which have not been followed by successful logins. Specific user names may be given to determine the last failed login when there has been a successful login after the most recent failure.

## 4. Special Files

The Shadow Password Suite defines 3 news files. They are */etc/porttime*, */etc/gshadow* and */usr/adm/faillog*.

### 4.1 /etc/porttime File

This file defines the times of day, days of week, and terminal ports which a user or list of users may use. The format of the file is a text file with one entry per line. Comment lines are introduced with a pound character. Each entry contains the colon-separated following fields

- comma-separated list of terminal ports
- comma-separated list of user names
- comma-separated list of daytimes

The terminal port list consists of device names with the leading */dev/* prefix removed. An asterisk indicates that all terminal devices match this entry.

Each named user will match this entry when attempting to login use a listed port. An asterisk indicates that all users match this entry.

The daytimes are given as one or more days followed by a hyphenated range of times. The day codes are Su (Sunday), Mo (Monday), Tu (Tuesday), We (Wednesday), Th (Thursday), Fr (Friday), Sa (Saturday), Wk (Monday thru Friday) and Al (all seven days). The range of times is given as two four-digit times. The first time may be greater than the second time, indicating that the time period starts at the initial time value, extends to midnight, then resumes in the morning and extends until the final time.

### 4.2 /etc/gshadow File

The */etc/gshadow* file contains the encrypted group password and group administrator information as well as the group membership. The file consists of lines of text with each line representing one group. Each group entry contains

- group name
- encrypted group password
- comma-separated list of group administrators
- comma-separated list of group members

The group name corresponds to the name in the */etc/group* file. The encrypted group password replaces the corresponding field in the old group file and performs a similar function. The comma-separated list of group administrators contains the names of users who are permitted to add members to the group membership list or change the group password. The comma-separated list of group members contains the names of users who are permitted to switch to the group without providing the correct password. Any user not in the group membership list will be required to provide the correct password when using the *newgrp* or *sg* commands.

### 4.3 /usr/adm/faillog File

The */usr/adm/faillog* file contains the control information for locking out an account with an excessive number of login failures. The file consists of binary data representing C language data structures. Each entry is indexed by numerical user identifier. The contents of each entry is

- (short) count of failures since the last successful login
- (short) limit of login failures before the account is disabled
- (char[12]) device name of the login port where the last failure occurred
- (time_t) time of the last login failure

This file is supported by the *faillog* utility. A value of zero for the failure limit will prevent an account from being locked out due to excessive failures. The failures will continue to be logged.

## 5. Package Configuration

The Shadow Password Suite is configured at compilation time with the include file *config.h*. This file controls the basic software support that is present on the system or the amount of functionality that the system administrator wishes to include.

### 5.1 LOGINDEFS Definition

This define controls the location of the configuration file that is used by the *login* command. The default location is */etc/login.defs*. This file contains the control values for the administrator configurable functions. It is a user editable file. Each entry consists of an identifier and a string giving the value for the parameter.

### 5.2 SHADOWPWD Definition

This define controls support for the */etc/shadow* file. This macro must be defined as most of the commands are dependent on the functions used to access that file.

### 5.3 SHADOWGRP Definition

This define controls support for the */etc/gshadow* file. This is an optional feature that provides for group administrators and a means of concealing group passwords.

### 5.4 DOUBLESIZE Definition

This define controls support for 16 character passwords. This feature extends the encrypted password by appending 11 additional characters of ciphertext to the end of the 13 character salt and ciphertext standard. For users with passwords 9 characters and longer, the stored encrypted password will be 24 characters in length. For users with passwords 8 characters and shorter, the stored encrypted password is the standard 13 character string.

### 5.5 AGING Definition

This define controls support for */etc/passwd* style password aging. This format uses a four character string appended to the encrypted password to indicate the age and expiration information for the user's password. This macro must be defined for account expiration to function. Systems which do not wish to enforce account and password expiration values may leave this macro undefined.

### 5.6 DBM and NDBM Definitions

These defines select the database functions to be used to support the database files. The system administrator may define either of these macros, but not both. If there is no desire to use the database files, the system administrator should define neither of the two macros.

### 5.7 USE_SYSLOG Definition

This define controls the use of the *syslog()* function. Programs and functions which log their actions with *syslog()* use this macro to control inclusion of the function calls. This macro should be defined if at all possible as the programs report intrusion attempts as they are detected.

### 5.8 RLOGIN Definition

This define controls the inclusion of code to perform remote network logins. This macro controls the support for *rlogind* and *telnetd* TCP/IP services.

### 5.9 DIR_XENIX, DIR_BSD, and DIR_SYSV Definitions

These defines control the type of directory access routines which are used by the system. These functions are used by the programs which traverse user directories while changing user identifiers or ownership on files, or programs which copy or remove user files when a user home directory is moved or deleted.

## 6. Documentation

Nroff -man macro documentation files are provided for all of the programs that are provided as part of the Shadow Password Suite. The files which are unique to the package are documented as are any programming interfaces.

## 7. Availability

This software is available from USENET archive sites which store the comp.sources.misc newsgroup. Distribution is permitted for all non-commercial purposes.