NAME

pam_acct_mgmt, **pam_authenticate**, **pam_chauthtok**, **pam_close_session**, **pam_end**, **pam_get_data**, **pam_get_item**, **pam_get_user**, **pam_getenv**, **pam_getenvlist**, **pam_open_session**, **pam_putenv**, **pam_set_data**, **pam_set_item**, **pam_setcred**, **pam_start**, **pam_strerror** **-** Pluggable Authentication Modules Library

LIBRARY

Pluggable Authentication Module Library (libpam, -lpam)

SYNOPSIS

#include <security/pam_appl.h>

int
pam_acct_mgmt(*pam_handle_t *pamh, int flags*);

int
pam_authenticate(*pam_handle_t *pamh, int flags*);

int
pam_chauthtok(*pam_handle_t *pamh, int flags*);

int
pam_close_session(*pam_handle_t *pamh, int flags*);

int
pam_end(*pam_handle_t *pamh, int status*);

int
pam_get_data(*const pam_handle_t *pamh, const char *module_data_name, const void **data*);

int
pam_get_item(*const pam_handle_t *pamh, int item_type, const void **item*);

int
pam_get_user(*pam_handle_t *pamh, const char **user, const char *prompt*);

*const char **
pam_getenv(*pam_handle_t *pamh, const char *name*);

*char ***

pam_getenvlist(*pam_handle_t *pamh*);

int
pam_open_session(*pam_handle_t *pamh*, *int flags*);

int
pam_putenv(*pam_handle_t *pamh*, *const char *namevalue*);

int
pam_set_data(*pam_handle_t *pamh*, *const char *module_data_name*, *void *data*,
  *void (*cleanup)(pam_handle_t *pamh, void *data, int pam_end_status)*);

int
pam_set_item(*pam_handle_t *pamh*, *int item_type*, *const void *item*);

int
pam_setcred(*pam_handle_t *pamh*, *int flags*);

int
pam_start(*const char *service*, *const char *user*, *const struct pam_conv *pam_conv*,
  *pam_handle_t **pamh*);

*const char **
pam_strerror(*const pam_handle_t *pamh*, *int error_number*);

DESCRIPTION

The Pluggable Authentication Modules (PAM) library abstracts a number of common authentication-related operations and provides a framework for dynamically loaded modules that implement these operations in various ways.

Terminology

In PAM parlance, the application that uses PAM to authenticate a user is the server, and is identified for configuration purposes by a service name, which is often (but not necessarily) the program name.

The user requesting authentication is called the applicant, while the user (usually, root) charged with verifying his identity and granting him the requested credentials is called the arbitrator.

The sequence of operations the server goes through to authenticate a user and perform whatever task he requested is a PAM transaction; the context within which the server performs the requested task is called a session.

The functionality embodied by PAM is divided into six primitives grouped into four facilities: authentication, account management, session management and password management.

Conversation

The PAM library expects the application to provide a conversation callback which it can use to communicate with the user. Some modules may use specialized conversation functions to communicate with special hardware such as cryptographic dongles or biometric devices. See pam_conv(3) for details.

Initialization and Cleanup

The **pam_start**() function initializes the PAM library and returns a handle which must be provided in all subsequent function calls. The transaction state is contained entirely within the structure identified by this handle, so it is possible to conduct multiple transactions in parallel.

The **pam_end**() function releases all resources associated with the specified context, and can be called at any time to terminate a PAM transaction.

Storage

The **pam_set_item**() and **pam_get_item**() functions set and retrieve a number of predefined items, including the service name, the names of the requesting and target users, the conversation function, and prompts.

The **pam_set_data**() and **pam_get_data**() functions manage named chunks of free-form data, generally used by modules to store state from one invocation to another.

Authentication

There are two authentication primitives: **pam_authenticate**() and **pam_setcred**(). The former authenticates the user, while the latter manages his credentials.

Account Management

The **pam_acct_mgmt**() function enforces policies such as password expiry, account expiry, time-of-day restrictions, and so forth.

Session Management

The **pam_open_session**() and **pam_close_session**() functions handle session setup and teardown.

Password Management

The **pam_chauthtok**() function allows the server to change the user’s password, either at the user’s request or because the password has expired.

Miscellaneous

The **pam_putenv**(), **pam_getenv**() and **pam_getenvlist**() functions manage a private environment list in which modules can set environment variables they want the server to export during the session.

The **pam_strerror**() function returns a pointer to a string describing the specified PAM error code.

RETURN VALUES

The following return codes are defined by <*security/pam_constants.h*>:

[PAM_ABORT]	General failure.
[PAM_ACCT_EXPIRED]	User account has expired.
[PAM_AUTHINFO_UNAVAIL]	Authentication information is unavailable.
[PAM_AUTHTOK_DISABLE_AGING]	Authentication token aging disabled.
[PAM_AUTHTOK_ERR]	Authentication token failure.
[PAM_AUTHTOK_EXPIRED]	Password has expired.
[PAM_AUTHTOK_LOCK_BUSY]	Authentication token lock busy.
[PAM_AUTHTOK_RECOVERY_ERR]	Failed to recover old authentication token.
[PAM_AUTH_ERR]	Authentication error.
[PAM_BAD_CONSTANT]	Bad constant.
[PAM_BAD_FEATURE]	Unrecognized or restricted feature.
[PAM_BAD_HANDLE]	

	Invalid PAM handle.
[PAM_BAD_ITEM]	Unrecognized or restricted item.
[PAM_BUF_ERR]	Memory buffer error.
[PAM_CONV_ERR]	Conversation failure.
[PAM_CRED_ERR]	Failed to set user credentials.
[PAM_CRED_EXPIRED]	User credentials have expired.
[PAM_CRED_INSUFFICIENT]	Insufficient credentials.
[PAM_CRED_UNAVAIL]	Failed to retrieve user credentials.
[PAM_DOMAIN_UNKNOWN]	Unknown authentication domain.
[PAM_IGNORE]	Ignore this module.
[PAM_MAXTRIES]	Maximum number of tries exceeded.
[PAM_MODULE_UNKNOWN]	Unknown module type.
[PAM_NEW_AUTHTOK_REQD]	New authentication token required.
[PAM_NO_MODULE_DATA]	Module data not found.
[PAM_OPEN_ERR]	Failed to load module.
[PAM_PERM_DENIED]	Permission denied.

[PAM_SERVICE_ERR]
Error in service module.

[PAM_SESSION_ERR]
Session failure.

[PAM_SUCCESS] Success.

[PAM_SYMBOL_ERR]
Invalid symbol.

[PAM_SYSTEM_ERR]
System error.

[PAM_TRY_AGAIN]
Try again.

[PAM_USER_UNKNOWN]
Unknown user.

SEE ALSO

openpam(3), pam_acct_mgmt(3), pam_authenticate(3), pam_chauthtok(3), pam_close_session(3), pam_conv(3), pam_end(3), pam_get_data(3), pam_getenv(3), pam_getenvlist(3), pam_get_item(3), pam_get_user(3), pam_open_session(3), pam_putenv(3), pam_setcred(3), pam_set_data(3), pam_set_item(3), pam_start(3), pam_strerror(3)

STANDARDS

X/Open Single Sign-On Service (XSSO) - Pluggable Authentication Modules, June 1997.

AUTHORS

The OpenPAM library and this manual page were developed for the FreeBSD Project by ThinkSec AS and Network Associates Laboratories, the Security Research Division of Network Associates, Inc. under DARPA/SPAWAR contract N66001-01-C-8035 ("CBOSS"), as part of the DARPA CHATS research program.

The OpenPAM library is maintained by Dag-Erling Smørgrav <*des@des.no*>.