NAME

pam_unix − Module for traditional password authentication

SYNOPSIS

pam_unix.so [...]

DESCRIPTION

This is the standard Unix authentication module. It uses standard calls from the system's libraries to retrieve and set account information as well as authentication. Usually this is obtained from the /etc/passwd and the /etc/shadow file as well if shadow is enabled.

The account component performs the task of establishing the status of the user's account and password based on the following *shadow* elements: expire, last_change, max_change, min_change, warn_change. In the case of the latter, it may offer advice to the user on changing their password or, through the **PAM_AUTHTOKEN_REQD** return, delay giving service to the user until they have established a new password. The entries listed above are documented in the **shadow**(5) manual page. Should the user's record not contain one or more of these entries, the corresponding *shadow* check is not performed.

The authentication component performs the task of checking the users credentials (password). The default action of this module is to not permit the user access to a service if their official password is blank.

A helper binary, **unix_chkpwd**(8), is provided to check the user's password when it is stored in a read protected database. This binary is very simple and will only check the password of the user invoking it. It is called transparently on behalf of the user by the authenticating component of this module. In this way it is possible for applications like **xlock**(1) to work without being setuid−root. The module, by default, will temporarily turn off SIGCHLD handling for the duration of execution of the helper binary. This is generally the right thing to do, as many applications are not prepared to handle this signal from a child they didn't know was **fork()**d. The **noreap** module argument can be used to suppress this temporary shielding and may be needed for use with certain applications.

The maximum length of a password supported by the pam_unix module via the helper binary is *PAM_MAX_RESP_SIZE* − currently 512 bytes. The rest of the password provided by the conversation function to the module will be ignored.

The password component of this module performs the task of updating the user's password. The default encryption hash is taken from the **ENCRYPT_METHOD** variable from */etc/login.defs*

The session component of this module logs when a user logins or leave the system.

Remaining arguments, supported by others functions of this module, are silently ignored. Other arguments are logged as errors through **syslog**(3).

OPTIONS

debug
Turns on debugging via **syslog**(3).

audit
A little more extreme than debug.

quiet
Turns off informational messages namely messages about session open and close via **syslog**(3).

nullok
The default action of this module is to not permit the user access to a service if their official password is blank. The **nullok** argument overrides this default.

try_first_pass
Before prompting the user for their password, the module first tries the previous stacked module's password in case that satisfies this module as well.

use_first_pass
The argument **use_first_pass** forces the module to use a previous stacked modules password and will never prompt the user − if no password is available or the password is not appropriate, the user will be

denied access.

nodelay
This argument can be used to discourage the authentication component from requesting a delay should the authentication as a whole fail. The default action is for the module to request a delay−on−failure of the order of two second.

use_authtok
When password changing enforce the module to set the new password to the one provided by a previously stacked **password** module (this is used in the example of the stacking of the **pam_cracklib** module documented below).

authtok_type=*type*
This argument can be used to modify the password prompt when changing passwords to include the type of the password. Empty by default.

nis
NIS RPC is used for setting new passwords.

remember=*n*
The last *n* passwords for each user are saved in /etc/security/opasswd in order to force password change history and keep the user from alternating between the same password too frequently. The MD5 password hash algorithm is used for storing the old passwords. Instead of this option the **pam_pwhistory** module should be used.

shadow
Try to maintain a shadow based system.

md5
When a user changes their password next, encrypt it with the MD5 algorithm.

bigcrypt
When a user changes their password next, encrypt it with the DEC C2 algorithm.

sha256
When a user changes their password next, encrypt it with the SHA256 algorithm. If the SHA256 algorithm is not known to the **crypt**(3) function, fall back to MD5.

sha512
When a user changes their password next, encrypt it with the SHA512 algorithm. If the SHA512 algorithm is not known to the **crypt**(3) function, fall back to MD5.

blowfish
When a user changes their password next, encrypt it with the blowfish algorithm. If the blowfish algorithm is not known to the **crypt**(3) function, fall back to MD5.

rounds=*n*
Set the optional number of rounds of the SHA256, SHA512 and blowfish password hashing algorithms to *n*.

broken_shadow
Ignore errors reading shadow information for users in the account management module.

minlen=*n*
Set a minimum password length of *n* characters. The default value is 6. The maximum for DES crypt−based passwords is 8 characters.

obscure
Enable some extra checks on password strength. These checks are based on the "obscure" checks in the original shadow package. The behavior is similar to the pam_cracklib module, but for non−dictionary−based checks. The following checks are implemented:

Palindrome
Verifies that the new password is not a palindrome of (i.e., the reverse of) the previous one.

Case Change Only
Verifies that the new password isn't the same as the old one with a change of case.

Similar
Verifies that the new password isn't too much like the previous one.

Simple
Is the new password too simple? This is based on the length of the password and the number of different types of characters (alpha, numeric, etc.) used.

Rotated
Is the new password a rotated version of the old password? (E.g., "billy" and "illyb")

no_pass_expiry
When set ignore password expiration as defined by the *shadow* entry of the user. The option has an effect only in case *pam_unix* was not used for the authentication or it returned authentication failure meaning that other authentication source or method succeeded. The example can be public key authentication in *sshd*. The module will return **PAM_SUCCESS** instead of eventual **PAM_NEW_AUTHTOK_REQD** or **PAM_AUTHTOK_EXPIRED**.

Invalid arguments are logged with **syslog**(3).

MODULE TYPES PROVIDED

All module types (**account**, **auth**, **password** and **session**) are provided.

RETURN VALUES

PAM_IGNORE
Ignore this module.

EXAMPLES

An example usage for /etc/pam.d/login would be:

```
# Authenticate the user
auth      required   pam_unix.so
# Ensure users account and password are still active
account   required   pam_unix.so
# Change the user's password, but at first check the strength
# with pam_cracklib(8)
password  required   pam_cracklib.so retry=3 minlen=6 difok=3
password  required   pam_unix.so use_authtok nullok md5
session   required   pam_unix.so
```

SEE ALSO

login.defs(5), **pam.conf**(5), **pam.d**(5), **pam**(7)

AUTHOR

pam_unix was written by various people.