NAME

pam_pwquality − PAM module to perform password quality checking

SYNOPSIS

pam_pwquality.so [...]

DESCRIPTION

This module can be plugged into the **password** stack of a given service to provide some plug-in strength-checking for passwords. The code was originally based on pam_cracklib module and the module is backwards compatible with its options.

The action of this module is to prompt the user for a password and check its strength against a system dictionary and a set of rules for identifying poor choices.

The first action is to prompt for a single password, check its strength and then, if it is considered strong, prompt for the password a second time (to verify that it was typed correctly on the first occasion). All being well, the password is passed on to subsequent modules to be installed as the new authentication token.

The checks for strength are:

Palindrome
: Is the new password a palindrome?

Case Change Only
: Is the new password the the old one with only a change of case?

Similar
: Is the new password too much like the old one? This is primarily controlled by one argument, **difok** which is a number of character changes (inserts, removals, or replacements) between the old and new password that are enough to accept the new password.

Simple
: Is the new password too small? This is controlled by 6 arguments **minlen**, **maxclassrepeat**, **dcredit**, **ucredit**, **lcredit**, and **ocredit**. See the section on the arguments for the details of how these work and there defaults.

Rotated
: Is the new password a rotated version of the old password?

Same consecutive characters
: Optional check for same consecutive characters.

Too long monotonic character sequence
: Optional check for too long monotonic character sequence.

Contains user name
: Check whether the password contains the user’s name in some form.

Dictionary check
: The *Cracklib* routine is called to check if the password is part of a dictionary.

These checks are configurable either by use of the module arguments or by modifying the */etc/security/pwquality.conf* configuration file. The module arguments override the settings in the configuration file.

OPTIONS

debug
: This option makes the module write information to **syslog** (3) indicating the behavior of the module (this option does not write password information to the log file).

authtok_type=*XXX*
: The default action is for the module to use the following prompts when requesting passwords: `"New UNIX password: "`and `"Retype UNIX password: "`. The e xample word *UNIX* can be replaced with this option, by default it is empty.

retry=*N*
Prompt user at most *N* times before returning with error. The default is *1*.

difok=*N*
This argument will change the default of *1* for the number of changes in the new password from the old password.

The special value of *0* disables all checks of similarity of the new password with the old password except the new password being exactly the same as the old one.

minlen=*N*
The minimum acceptable size for the new password (plus one if credits are not disabled which is the default). In addition to the number of characters in the new password, credit (of +1 in length) is given for each different kind of character (*other*, *upper*, *lower* and *digit*). The default for this parameter is *8*. Note that there is a pair of length limits also in *Cracklib*, which is used for dictionary checking, a ‘‘way too short’’ limit of *4* which is hard coded in and a build time defined limit (*6*) that will be checked without reference to **minlen**.

dcredit=*N*
(N >= 0) This is the maximum credit for having digits in the new password. If you have less than or *N* digits, each digit will count +1 towards meeting the current **minlen** value. The default for **dcredit** is *0* which means there is no bonus for digits in password.

(N < 0) This is the minimum number of digits that must be met for a new password.

ucredit=*N*
(N >= 0) This is the maximum credit for having upper case letters in the new password. If you have less than or *N* upper case letters, each upper case letter will count +1 towards meeting the current **minlen** value. The default for **ucredit** is *0* which means there is no bonus for upper case letters in password.

(N < 0) This is the minimum number of upper case letters that must be met for a new password.

lcredit=*N*
(N >= 0) This is the maximum credit for having lower case letters in the new password. If you have less than or *N* lower case letters, each lower case letter will count +1 towards meeting the current **minlen** value. The default for **lcredit** is *0* which means there is no bonus for lower case letters in password.

(N < 0) This is the minimum number of lower case letters that must be met for a new password.

ocredit=*N*
(N >= 0) This is the maximum credit for having other characters in the new password. If you have less than or *N* other characters, each other character will count +1 towards meeting the current **minlen** value. The default for **ocredit** is *0* which means there is no bonus for other characters in password.

(N < 0) This is the minimum number of other characters that must be met for a new password.

minclass=*N*
The minimum number of required classes of characters for the new password. The four classes are digits, upper and lower letters and other characters. The difference to the **credit** check is that a specific class if of characters is not required. Instead *N* out of four of the classes are required. By default the check is disabled.

maxrepeat=*N*
Reject passwords which contain more than *N* same consecutive characters. The default is 0 which means that this check is disabled.

maxsequence=*N*
Reject passwords which contain monotonic character sequences longer than *N*. The default is 0 which means that this check is disabled. Examples of such sequence are ’12345’ or ’fedcb’. Note that most such passwords will not pass the simplicity check unless the sequence is only a minor part of the

password.

maxclassrepeat=*N*
Reject passwords which contain more than *N* consecutive characters of the same class. The default is 0 which means that this check is disabled.

gecoscheck=*N*
If nonzero, check whether the individual words longer than 3 characters from the **passwd** (5) GECOS field of the user are contained in the new password. The default is 0 which means that this check is disabled.

dictcheck=*N*
If nonzero, check whether the password (with possible modifications) matches a word in a dictionary. Currently the dictionary check is performed using the *cracklib* library. The default is 1 which means that this check is enabled.

usercheck=*N*
If nonzero, check whether the password (with possible modifications) contains the user name in some form. The default is 1 which means that this check is enabled. It is not performed for user names shorter than 3 characters.

usersubstr=*N*
If greater than 3 (due to the minimum length in usercheck), check whether the password contains a substring of at least *N* length in some form. The default is 0, which means this check is disabled.

enforcing=*N*
If nonzero, reject the password if it fails the checks, otherwise only print the warning. The default is 1 which means that the weak password is rejected (for non-root users).

badwords=*<list of words>*
The words more than 3 characters long from this space separated list are individually searched for and forbidden in the new password. By default the list is empty which means that this check is disabled.

dictpath=*/path/to/dict*
This options allows for specification of non-default path to the cracklib dictionaries.

enforce_for_root
The module will return error on failed check even if the user changing the password is root. This option is off by default which means that just the message about the failed check is printed but root can change the password anyway. Note that root is not asked for an old password so the checks that compare the old and new password are not performed.

local_users_only
The module will not test the password quality for users that are not present in the */etc/passwd* file. The module still asks for the password so the following modules in the stack can use the **use_authtok** option. This option is off by default.

use_authtok
This argument is used to *force* the module to not prompt the user for a new password but use the one provided by the previously stacked **password** module.

MODULE TYPES PROVIDED

Only the **password** module type is provided.

RETURN VALUES

PAM_SUCCESS
The new password passes all checks.

PAM_AUTHTOK_ERR
No new password was entered, the username could not be determined or the new password fails the strength checks.

PAM_AUTHTOK_RECOVERY_ERR
: The old password was not supplied by a previous stacked module or got not requested from the user. The first error can happen if **use_authtok** is specified.

PAM_SERVICE_ERR
: A internal error occurred.

EXAMPLES

For an example of the use of this module, we show how it may be stacked with the password component of **pam_unix** (8).

```
#
# These lines stack two password type modules. In this example the
# user is given 3 opportunities to enter a strong password. The
# "use_authtok" argument ensures that the pam_unix module does not
# prompt for a password, but instead uses the one provided by
# pam_pwquality.
#
password required pam_pwquality.so retry=3
password required pam_unix.so use_authtok
```

Another example is for the case that you want to use sha256 password encryption:

```
#
# These lines allow modern systems to support passwords of at least 14
# bytes with extra credit of 2 for digits and 2 for others the new
# password must have at least three bytes that are not present in the
# old password
#
password required pam_pwquality.so \
              difok=3 minlen=15 dcredit=2 ocredit=2
password required pam_unix.so use_authtok nullok sha256
```

And here is another example in case you don't want to use credits:

```
#
# These lines require the user to select a password with a minimum
# length of 8 and with at least 1 digit number, 1 upper case letter,
# and 1 other character
#
password required pam_pwquality.so \
              dcredit=-1 ucredit=-1 ocredit=-1 lcredit=0 minlen=8
password required pam_unix.so use_authtok nullok sha256
```

SEE ALSO

pwscore (1), **pwquality.conf** (5), **pam_pwquality** (8), **pam.conf** (5), **PAM** (8)

AUTHORS

Tomas Mraz <tmraz@redhat.com>

Original author of **pam_cracklib** module Cristian Gafton <gafton@redhat.com>