NAME

pam_passwdqc - Password quality-control PAM module

SYNOPSIS

[*service-name*] *module-type control-flag pam_passwdqc* [*options*]

DESCRIPTION

The **pam_passwdqc** module is a simple password strength checking module for PAM. In addition to checking regular passwords, it offers support for passphrases and can provide randomly generated passwords.

The **pam_passwdqc** module provides functionality for only one PAM category: password changing. In terms of the *module-type* parameter, this is the "password" feature.

The **pam_chauthtok**() service function will ask the user for a new password, and verify that it meets certain minimum standards. If the chosen password is unsatisfactory, the service function returns PAM_AUTHTOK_ERR.

The following options may be passed to the authentication module:

min=*N0*,*N1*,*N2*,*N3*,*N4*

(**min**=**disabled**,24,12,8,7) The minimum allowed password lengths for different kinds of passwords/passphrases. The keyword **disabled** can be used to disallow passwords of a given kind regardless of their length. Each subsequent number is required to be no larger than the preceding one.

N0 is used for passwords consisting of characters from one character class only. The character classes are: digits, lower-case letters, upper-case letters, and other characters. There is also a special class for non-ASCII characters which could not be classified, but are assumed to be non-digits.

N1 is used for passwords consisting of characters from two character classes, which do not meet the requirements for a passphrase.

N2 is used for passphrases. A passphrase must consist of sufficient words (see the **passphrase** option below).

N3 and *N4* are used for passwords consisting of characters from three and four character classes, respectively.

When calculating the number of character classes, upper-case letters used as the first character and digits used as the last character of a password are not counted.

In addition to being sufficiently long, passwords are required to contain enough different characters for the character classes and the minimum length they have been checked against.

max=*N*

(**max**=40) The maximum allowed password length. This can be used to prevent users from setting passwords which may be too long for some system services. The value 8 is treated specially: if **max** is set to 8, passwords longer than 8 characters will not be rejected, but will be truncated to 8 characters for the strength checks and the user will be warned. This is for compatibility with the traditional DES password hashes, which truncate the password at 8 characters.

It is important that you do set **max**=8 if you are using the traditional hashes, or some weak passwords will pass the checks.

passphrase=*N*

(**passphrase**=3) The number of words required for a passphrase, or 0 to disable passphrase support.

match=*N*

(**match**=4) The length of common substring required to conclude that a password is at least partially based on information found in a character string, or 0 to disable the substring search. Note that the password will not be rejected once a weak substring is found; it will instead be subjected to the usual strength requirements with the weak substring removed.

The substring search is case-insensitive and is able to detect and remove a common substring spelled backwards.

similar=**permit**|**deny**

(**similar**=**deny**) Whether a new password is allowed to be similar to the old one. The passwords are considered to be similar when there is a sufficiently long common substring and the new password with the substring removed would be weak.

random=*N*[,**only**]

(**random**=42) The size of randomly-generated passwords in bits, or 0 to disable this feature. Passwords that contain the offered randomly-generated string will be allowed regardless of other possible restrictions.

The **only** modifier can be used to disallow user-chosen passwords.

enforce=**none**|**users**|**everyone**

(**enforce**=**everyone**) The module can be configured to warn of weak passwords only, but not actually enforce strong passwords. The **users** setting will enforce strong passwords for non-root users only.

non-unix

Normally, **pam_passwdqc** uses getpwnam(3) to obtain the user's personal login information and use that during the password strength checks. This behavior can be disabled with the **non-unix** option.

retry=*N*

(**retry**=3) The number of times the module will ask for a new password if the user fails to provide a sufficiently strong password and enter it twice the first time.

ask_oldauthtok[=**update**]

Ask for the old password as well. Normally, **pam_passwdqc** leaves this task for subsequent modules. With no argument, the **ask_oldauthtok** option will cause **pam_passwdqc** to ask for the old password during the preliminary check phase. If the **ask_oldauthtok** option is specified with the **update** argument, **pam_passwdqc** will do that during the update phase.

check_oldauthtok

This tells **pam_passwdqc** to validate the old password before giving a new password prompt. Normally, this task is left for subsequent modules.

The primary use for this option is when **ask_oldauthtok**=**update** is also specified, in which case no other modules gets a chance to ask for and validate the password. Of course, this will only work with UNIX passwords.

use_first_pass, **use_authtok**

Use the new password obtained by modules stacked before **pam_passwdqc**. This disables user interaction within **pam_passwdqc**. The only difference between **use_first_pass** and **use_authtok** is that the former is incompatible with **ask_oldauthtok**.

SEE ALSO

getpwnam(3), pam.conf(5), pam(3)

AUTHORS

The **pam_passwdqc** module was written by Solar Designer <*solar@openwall.com*>. This manual page,

derived from the author’s documentation, was written for the FreeBSD Project by ThinkSec AS and NAI Labs, 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.