

Choosing a Good Password with Npasswd

By Boris Loza

If a computer system's security is one of your concerns, you need not be reminded that one of the first points of attack from hackers is the user's password. Even though you may think you have taught users how to select good passwords, you can still come across many guessed passwords after running some password strength checking software.

Each time we've checked our user's accounts, we've come up with as many as 10 percent easily-cracked passwords. This was a real nightmare until we decided to find a way to prevent users from using easy-to-guess passwords.

Avoiding easy-to-guess passwords

Unfortunately, SunOS doesn't supply system utilities that restrain the ways users select their passwords. What we needed was a utility that would help our users to easily choose a secure password.

We have found several utilities that can do this job for us: `passwd+`, `anlpasswd`, `password coach`, `obvious-pw`, `npasswd`, `gen-password` and `cracklib`. Development and technical support for some of these utilities have been discontinued, and some are too old or not well documented.

Taking all this into consideration, we decided to try `npasswd`, developed by Clyde Hoover from the University of Texas at Austin. Various people have assisted its development. Alec Muffet, for example, provided `npasswd` developers with his famous Crack's "modules ready to use and a copious collection of word lists".

`Npasswd` replaces the `passwd` utility for UNIX. You can download its latest release (version 2.05) from <http://www.utexas.edu/cc/unix/software/npasswd/download1.html>. You'll find documentation at <http://www.utexas.edu/cc/unix/software/npasswd/doc/>, or in the `doc` directory in the source distribution, or in `<your_install_path>/lib/passwd/doc` when the installation is complete.

Guessability tests

Hoover designed `npasswd` to provide a series of guessability tests. The user's password is accepted only when it passes all of the tests. You can customize which tests `npasswd` uses and in what order these tests should be applied with the configuration file. Some tests are mandatory, and others optional. The password tests are History and Lexical.

History

This test is optional. Password candidates are compared to the passwords in the user's history and rejected if found. When a password change is done, the new encrypted password is stored in the history database.

Lexical

This test is mandatory. The lexical check includes:

- Enforcing a minimum length of six characters.
- Checking for non-printable or forbidden characters.
- Denying excessive adjacent repeated characters.
- Encouraging a diversity of character classes (mixed case, numbers, punctuation).
- Looking for easily guessed patterns (Social Security numbers, telephone numbers, etc.).

Criteria that can be modified

The following criteria can be modified in the configuration file:

- **Local** (optional). This is the hook for the insertion of site-specific tests. The standard code checks if the candidate is a variation of the system hostname, aliases, or entries in the user's `.rhosts` file.
- **Password** (mandatory). The candidate is examined to see if it is derived from the user's previous password information.
- **Dictionary** (mandatory). `Npasswd` checks the candidate against words in various dictionaries. The candidate is rejected if it can be derived from any word in the dictionaries. A variant of the `Crack` password guessing code is used for this purpose.

Installing npasswd

To install `npasswd`, run `gunzip` and `tar xvf` on the `npasswd-2_05_tar.tar` file. Then, change to the directory `npasswd-2.05` and run `Configure` – the script to set the options to be incorporate in `npasswd`. The meaning of the various options can be found in the `npasswd` documentation. You can safely hit `[Enter]` on most of the default options. Answer `[n]` for the `Replace system programs?` prompt. We'll explain why later.

In our case, we chose to install `npasswd` under the `/usr/local/` directory. This path would be our relative path to all `npasswd` files. After `Configure` was complete, we typed `make`, and then logged on as `root` and typed `make install`. Note that the installation process doesn't create the password's history database. Use the `history_admin` utility to create and manage the database. That's it! You'll find the `npasswd` object file found under the `/usr/local/lib/passwd/` directory.

Editing the passwd.conf file

Now we can edit a `passwd.conf` file. This file is used to tell the `npasswd` what to allow and what to check for in a new user password. You can modify this file to meet your own security policy. For example, add the following lines:

```
Passwd.MinPassword      8
passwd.CharClasses      3
passwd.History depth    5
passwd.MaxRepeat        1
```

tells `npasswd` that the user password must be at least eight characters long, checks for at least three different character classes in user passwords, remembers the last five passwords that a user employed, and allows only one repeated character. For all `npasswd` options, see [Table A](#). You'll also find a complete list of the configuration options in the `passwd.conf` file itself.

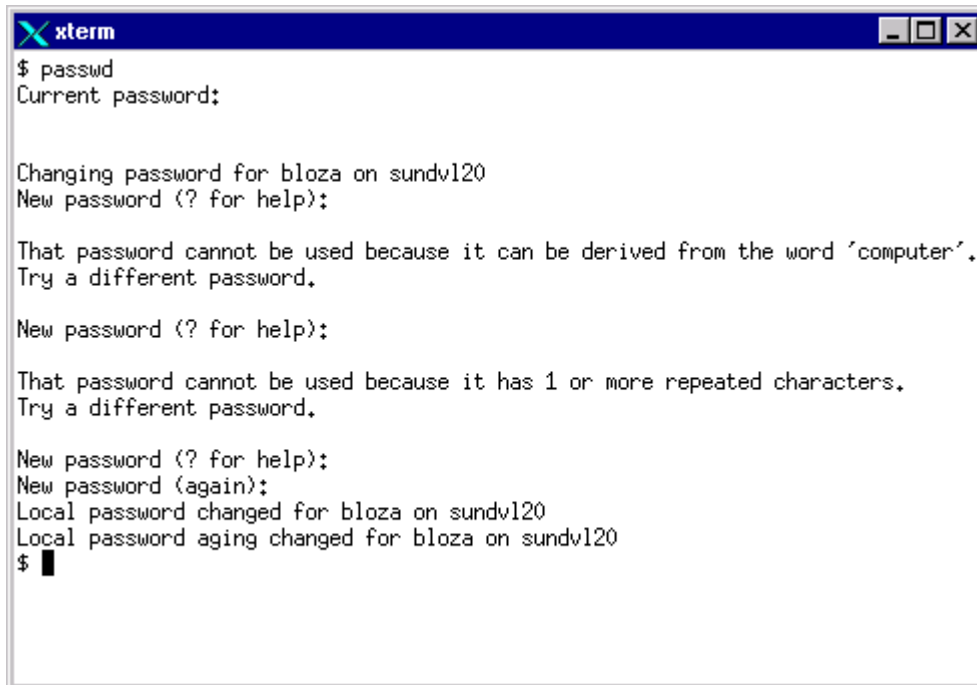
Table A: Configuration options for the `passwd.conf` file

<u>Directive type</u>	<u>Value</u>	<u>Description</u>
PasswdTolerance	number	Tolerance between old and new passwd files
ShadowTolerance	number	Tolerance between old and new shadow files
MatchTries	number	Chances to give user to correctly enter a password
MatchWait	number	Delay after the user enters an incorrect password
passwd.AlphaOnly	boolean	Allow alpha-only passwords
passwd.CharClasses	number	Minimum number of character classes required
passwd.Dictionaries	path	Password check dictionaries location
passwd.DisallowedChars	string	Change list of non-allowed characters
passwd.Help	path	Help file for passwd
passwd.History age	number	Password history age
passwd.History dbm	path	Set path to history db (DBM database)
passwd.History depth	number	How many password to keep per user
passwd.History file	path	Set path to history db (flat file)
passwd.History none		Disable password history
passwd.LengthWarn	boolean	Warn about passwords over MaxPassword length
passwd.MaxPasses	number	Maximum effective password length
passwd.MaxRepeat	number	How many repeated characters allowed
passwd.Message	path	Message of the day file for passwd
passwd.MinPassword	number	Minimum password length
passwd.PasswordChecks	string	Choose password check functions
passwd.PrintableOnly	boolean	Deny non-printable characters in passwords
passwd.Singlecase	boolean	Allow single-case passwords
passwd.WhiteSpace	boolean	Deny whitespace characters in passwords

If you select to replace the system `passwd` binary with `npasswd` during installation, it copies itself instead the `/usr/bin/passwd` and puts an original `passwd` utility under `/usr/local/lib/passwd/system/` directory.

We decided not to replace the system `/usr/bin/passwd` file with the `npasswd` binary (that's why we have answered no to the prompt asking if we wanted to replace system programs) for several reasons. First, `npasswd` doesn't allow you to delete a user password and force a user to choose a new one on the next login. You must specify a dummy password and notify a user what the password is. You also have to use `passwd -s -a` instead of `-sa` (show attributes for all entries) option. When you try to use `passwd -e` (change user shell) and `passwd -g` (change user GECOS information), `npasswd` misbehaves and the session needs to be restarted. Also, patching the `/usr/bin/passwd` may replace the `npasswd` binary.

So, we just linked `/usr/local/lib/passwd/npasswd` with `/usr/local/bin/passwd` and put this before `/usr/bin/` in the `PATH` variable in the user profile. So, when user types a `passwd` command he'll start `npasswd`. In [Figure A](#), we've depicted a typical password change session.



```
xterm
$ passwd
Current password:

Changing password for bloza on sundv120
New password (? for help):

That password cannot be used because it can be derived from the word 'computer'.
Try a different password.

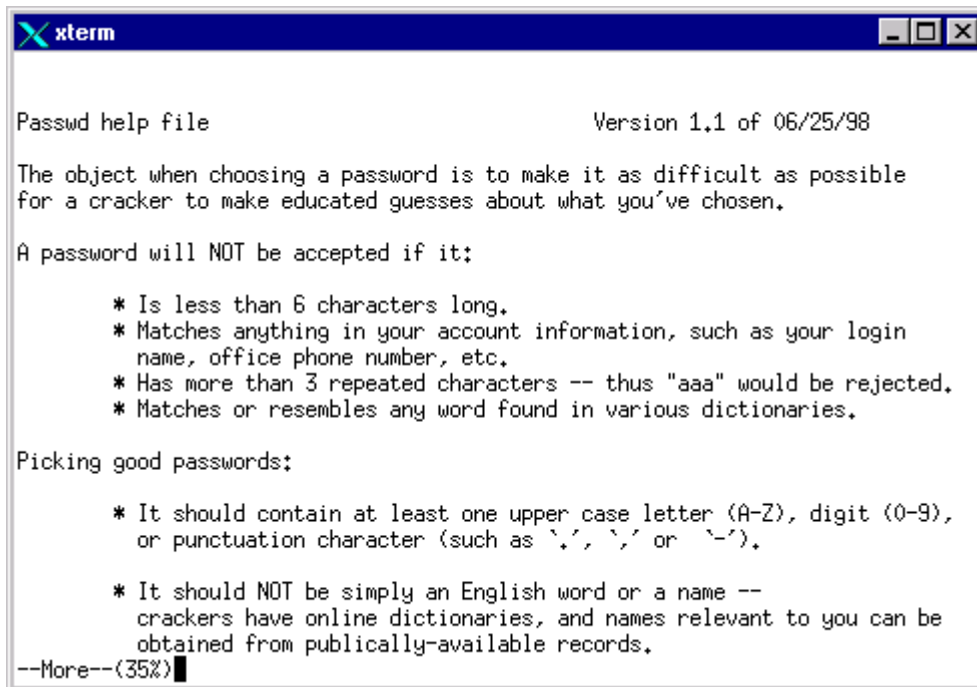
New password (? for help):

That password cannot be used because it has 1 or more repeated characters.
Try a different password.

New password (? for help):
New password (again):
Local password changed for bloza on sundv120
Local password aging changed for bloza on sundv120
$ █
```

Figure A: *This is a Typical password change session*

By typing a question mark (?), a user can see Help, as show in [Figure B](#):



```
Passwd help file                               Version 1.1 of 06/25/98

The object when choosing a password is to make it as difficult as possible
for a cracker to make educated guesses about what you've chosen.

A password will NOT be accepted if it:

    * Is less than 6 characters long.
    * Matches anything in your account information, such as your login
      name, office phone number, etc.
    * Has more than 3 repeated characters -- thus "aaa" would be rejected.
    * Matches or resembles any word found in various dictionaries.

Picking good passwords:

    * It should contain at least one upper case letter (A-Z), digit (0-9),
      or punctuation character (such as \', \', or \').

    * It should NOT be simply an English word or a name --
      crackers have online dictionaries, and names relevant to you can be
      obtained from publically-available records.

--More--(35%)
```

Figure B: A user can read Help by simply typing a question mark.

You can modify the Help contents by editing the `/usr/local/lib/passwd/passwd.help` file. Or you can create a message of the day for `npasswd` by altering a `passwd.motd` file. If you don't replace the `passwd` with the `npasswd`, the latter isn't invoked when the system password aging forces the user to change his password (or `passwd -f login_name` is issued). Because we didn't replace the `passwd` binary with the `npasswd`, we decided to implement our own method of password expiration.

Implementing your own method of password expiration

First we disabled the password aging by typing `passwd -x -1 user_name`. To do this, you have to be a root. Next we needed to force the users to change their passwords every 60 days or any time we want them to do so (like using the `passwd -f` command).

We achieved this goal in two stages. First we developed a shell script to create a marker file `.newpasswd.<user_name>` in the user home directory, as seen in [Listing A](#). This marker file is checked when the user logs on. If this file is found, the user will be forced to change their password.

We've altered the user's profile in order to call the script that looks for the marker file, as seen in [Listing B](#). The marker file is unique to each login name, so users can share the same home directory.

This script logs successful password changes to the `local3.notice` facility of `syslogd`, which writes to the `/var/adm/npasswd` file. To do this, we reconfigured the `/etc/syslog.conf` file. We forced users to change their passwords every 60 days by adding one line into the root's `crontab`. Forcing users to change their passwords on the next login can be done by modifying the `.newpasswd.<user_name>` file in the user home directory.

[Listing A: Script to create a marker file](#)

```
#!/bin/sh
#
nawk '
BEGIN {
    FS=":"
}

# Ignore system default users.
# Add more users here if you do not want them to be forced to change
# their password.
$1 == "root"    {next}
$1 == "daemon"  {next}
$1 == "bin"     {next}
$1 == "sys"     {next}
$1 == "adm"     {next}
$1 == "lp"      {next}
$1 == "smtp"    {next}
$1 == "uucp"    {next}
$1 == "nuucp"   {next}
$1 == "listen"  {next}
$1 == "nobody"  {next}
$1 == "noaccess" {next}
$1 == "nobody4" {next}

# Ignore users with / as a home directory (do not want to stomp on root).
$6 == "/" { next }

# For everyone else, create a marker file in their home directory
{
    # Create marker file in users home directory
    marker_file=$6/.newpasswd."$1

    if(system("/usr/bin/test -d " $6))
        print "Warning: ",$1,"has no home directory", $6
    else
        if(system("/usr/bin/touch " marker_file " 2>/dev/null"))
            print "Warning: Could not create marker file for", $6
        else
            print "Created marker file for", $1
    }
} ' /etc/passwd
```

[Listing B: Script to look for a marker file in the user home directory](#)

```
#!/bin/sh

USERNAME=`who am i | cut -d" " -f1`
MARKER_FILE=$HOME/.newpasswd.${USERNAME}
LOGGER="/usr/bin/logger -t PASSWDCHG -p local3.notice"
PASSWD="/usr/local/bin/passwd

trap 'exit' 1 2 3 5 15

if [ -f ${MARKER_FILE} ]; then
{
    echo
    echo
    echo "*****"
    echo "Please change your password now."
```

```

echo "You will not be allowed system access until you do!"
echo "*****"
echo
echo
echo ${PASSWD}

if [ $? -ne 0 ]; then
    exit 1
fi

${LOGGER} ${USERNAME} has successfully changed password
rm -f ${MARKER_FILE}
}
fi

#Put traps back to normal

trap - 1 2 3 5 15

```

Checkpassword

Another useful utility that `npasswd` comes with is the `checkpassword` command. You can do a preliminary check of a password without changing it. Just type `checkpassword` and on the prompt type a password you'd like to check (for more information see the man pages for `checkpassword`).

We also created our own password dictionaries that contain our company and business-related words. The password dictionary is a text file with one word per line. Applying the `makedict` command creates the dictionary hash files with the suffixes `.pwd`, `.pwi` and `.hwm`. To make a dictionary accessible to `npasswd`, copy the hash files (or make symbolic links) into the appropriate directory. The files should be world-readable. `Npasswd` will reject any dictionary with a world-writable hash file. For more information of how to create a dictionary file, you can read the online documentation.

Removing npasswd

Removing `npasswd` is an easy task. If you decide not to use `npasswd` anymore, you have to edit the user profile and put the `/usr/bin/` before `/usr/local/bin` in the `PATH` variable. Disable a crontab entry for the `npasswd` 'aging'. Enable password expiration by typing as root

```
passwd -n A -x B <user_name>
```

where `A` is the minimum days before the password can be changed and `B` is a maximum number of days.

Conclusion

After enabling `npasswd`, our lives became easier. You just need to periodically maintain a history database and check the `/var/adm/npasswd` file for users who didn't change their passwords in time.

