Passwords

All Linux users have a username and password associated with their account, except where a user account has been explicitly locked (designated *LK*), or where a system account has been specified not to have a password at all (NP). Many early exploits of Linux systems were associated with default passwords used on some system accounts, and the most common method of gaining unauthorized access to a Linux system remains password cracking and/or guessing. In this section, we examine the password database (*/etc/passwd*), and its more secure counterpart the shadow database (*/etc/shadow*), and examine strategies for making passwords safer.

The standard password database is stored in the file */etc/passwd*, and looks like this:

```
# cat /etc/passwd

root:x:0:1:Super-User:/:/sbin/sh

daemon:x:1:1::/:

bin:x:2:2::/usr/bin:

sys:x:3:3::/:

adm:x:4:4:Admin:/var/adm:

lp:x:71:8:Line Printer Admin:/usr/spool/lp:

uucp:x:5:5:uucp Admin:/usr/lib/uucp:

nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico

listen:x:37:4:Network Admin:/usr/net/nls:

nobody:x:60001:60001:Nobody:/:

noaccess:x:60002:60002:No Access User:/:

nobody4:x:65534:65534:SunOS 4.x Nobody:/:

postgres:x:1001:100:Postgres User:/usr/local/postgres:/bin/sh

htdig:x:1002:10:htdig:/opt/www:/usr/local/bin/bash

apache:x:1003:10:apache user:/usr/local/apache:/bin/sh
```

We have already seen some of the fields shown here when adding users to the system:

- The username field, which has a maximum of eight characters
- The encrypted password field, which in a system using shadow passwords is crossed with an *x*
- The user ID field, which contains the numeric and unique UID
- The primary group ID field, which contains the numeric GID
- The user comment, which contains a description of the user
- The path to the user’s home directory
- The user’s default shell

In older versions of Linux, the encrypted password field would have contained an encrypted password string like “X14oLaiYg7bO2”. However, this presented a security problem, as the login program required all users to have read access to the password file:

```
# ls -l /etc/passwd

-rw-r--r--   1 root     sys          605 Jul 24 11:04 /etc/passwd
```

Thus, any user with the lowest-form privilege would be able to access the encrypted password field for the root user, and could attempt to gain root access by guessing the password. A number of programs were specifically developed for this purpose, such as *crack*, which takes a standard Linux password file and uses a dictionary and some clever lexical rules to guess passwords.

Caution Once a root password has been obtained, a rogue user may perform any operation on a Linux system, including formatting hard disks, installing Trojan horses, launching attacks on other systems, and so forth.

The cryptographic algorithm used by Linux is not easy to crack—indeed, a brute force guess of a password composed of a completely random set of characters would take many CPU years to compute. The task would be made even more difficult (if not impossible) if the root password was changed weekly, again with a random set of characters. However, the reality is that most users enter passwords that are easily guessed from a dictionary, or from some knowledge about the user. Since we are constantly required to use PINs and passwords, people generally choose passwords that are easy to remember. However, easily remembered passwords are also the easiest to crack.

Linux has reduced the chances of a rogue user obtaining the password file in the first place by implementing a shadow password facility. This creates a file called */etc/shadow*, which is similar to the password file (*/etc/passwd*), but is only readable by root, and contains the encrypted password fields for each UID. Thus, if a rogue user cannot obtain the encrypted password entries, it is impossible to use them as the basis for a crack attack.

pwck

The pwck command is used to verify the accuracy of the password file. It reads */etc/passwd*, and it verifies that the expected number of fields exist for each entry in the file and validates the contents of the username, UID, and GID fields. It also checks whether the home directory exists and whether the default shell noted is a valid shell.

grpck

The grpck command is similar to the pwck command; you can use it to verify the accuracy of the group file. It reads */etc/group* and verifies that the expected number of fields exist for each entry in the file, and it validates the contents of the group name and GID fields. It also creates a list of usernames defined in the group and checks that these are contained in the */etc/passwd* file. If not, an error is reported, because an old user account may have been deleted from */etc/passwd* but may still be listed in a group.

pwconv

You can use the pwconv command to convert systems that do not have a shadow password file to use password shadowing. Most (if not all) modern systems would use password shadowing. However, if the */etc/shadow* file does not exist, the encrypted password is stripped from */etc/passwd*, and is replaced by *x*, indicating that the password for each user is shadowed. A shadow password file would then be created using the encrypted passwords extracted from the password file.

However, a more common use of *pwconv* is to update the shadow password file with entries that have been created manually in */etc/passwd*. Although this is not the recommended method of adding users to the system, some sites have scripts that create blocks of new user accounts by generating sequential usernames with generic group and password information. In such cases, it would be necessary to run *pwconv* after the script has been executed to ensure that entries created in */etc/passwd* are correctly transferred to */etc/shadow*.