S

Linux Shadow Password File: /etc/shadow Explained

The Linux /etc/shadow file stores password verifiers and account-aging data away from the publicly readable /etc/passwd file. Learn its nine fields, special markers, permissions, and safe administration practices.

PUBLISHER
Gridinsoft Support

UPDATED
Oct 8, 2026

READ TIME
4 min

CONTENTS

In this guide

01 /etc/passwd vs. /etc/shadow
02 The nine /etc/shadow fields
03 What password-field markers mean
04 How login verification works
05 Safe administration
06 If /etc/shadow is exposed or altered
07 Frequently asked questions

On Linux, `/etc/shadow` stores password information and optional password-aging data for local accounts. It is called a **shadow password file** because sensitive password verifiers were moved out of `/etc/passwd`, which must remain readable so programs can map user names, IDs, home directories, and shells.

The file does not normally contain readable passwords. It contains a password-hash representation or a special marker, plus account-aging fields. Access must be restricted; the Linux manual explicitly warns that regular users must not be able to read it if password security is to be maintained.

01 /etc/passwd vs. /etc/shadow

File	Typical contents	Access model
`/etc/passwd`	User name, UID, GID, description, home directory, login shell, and usually `x` in the password field	Readable by users and many programs
`/etc/shadow`	Password verifier or marker and password/account aging values	Readable only by privileged processes, commonly root and an authorized shadow group
`/etc/gshadow`	Protected group administration information	Restricted privileged access

Exact ownership and permissions vary by distribution. Do not replace distribution defaults with an arbitrary mode copied from another system.

02 The nine /etc/shadow fields

Each record is one colon-separated line:

1. **Login name** matching an existing local account.
2. **Password field** containing a verifier or special value.
3. **Last password change** as days since January 1, 1970.
4. **Minimum age** before the password may be changed again.
5. **Maximum age** after which a password change is required.
6. **Warning period** before password expiry.
7. **Inactivity period** after expiry before the account is disabled.

8. **Account expiration date** as days since January 1, 1970.
9. **Reserved field** for future use.

An empty aging field generally means that particular restriction is not set. Interpret values with the system's tools rather than editing the numeric dates manually.

03 What password-field markers mean

Formats and markers are implementation-dependent, but common cases include:

- A value beginning with an algorithm identifier such as `$id$...`, representing parameters, salt, and a password verifier.
- `!` or a value prefixed with `!`, commonly used to lock password authentication while retaining the previous verifier.
- `*` or another value that cannot be produced by the password-hashing function, preventing ordinary password login.
- An **empty password field**, which the manual warns may allow passwordless login for applications that honor it. Treat this as a serious configuration issue unless explicitly intended and controlled.

A locked password does not necessarily disable every way to use an account. SSH keys, tokens, scheduled services, or other authentication mechanisms may still work.

04 How login verification works

When a local password is set, a trusted system tool creates a verifier using the configured password-hashing method and salt. During password authentication, the supplied password is processed with the stored parameters and compared with the saved result. The original password does not need to be recovered. This is an example of one-way password hashing, not reversible storage.

05 Safe administration

- Use `passwd` to set or lock passwords and `chage` to inspect or modify aging.
- Use account-management tools such as `useradd`, `usermod`, and `userdel` instead of directly changing records.
- Use `vipw -s` only when expert manual recovery is necessary; it coordinates locking and basic validation.
- Use `pwck` to check consistency between account files.
- Protect backups and configuration-management exports containing shadow data as credentials.
- Prefer MFA and centralized identity where appropriate, while maintaining secure local recovery accounts.

06 If /etc/shadow is exposed or altered

1. Restrict access without destroying timestamps or evidence needed for investigation.
2. Determine which user or process read or changed the file and how privilege was obtained.
3. Assume offline password guessing is possible; reset affected passwords and reused credentials from trusted systems.
4. Review root access, sudo logs, SSH keys, PAM configuration, new accounts, scheduled tasks, and persistence.
5. Restore correct ownership and permissions using the distribution’s package, security baseline, or vendor guidance.
6. Rebuild the host if privileged compromise cannot be bounded.

07 Frequently asked questions

Can root read /etc/shadow?

Root and specifically authorized privileged processes normally can, because authentication and account tools require access. Ordinary users should not.

Should I manually paste a password hash into the file?

Usually no. Use supported account tools, which handle file locking, format, policy, and auditing and reduce the risk of corrupting authentication.

NEED MORE HELP?

We are here when the guide is not enough.

Visit the Gridinsoft Support Center for current product guidance or submit a support request with the details of your case.

Open this guide online at support.gridinsoft.com