Authentication and access controls

- Normal Unix authentication
- Managing account and password information
- Good password
- Creating encrypted password
- PAM and NSS
  - Configuration files accessed by PAM and NSS
  - The modern centralized system management
  - "Why GNU su does not support the wheel group"
  - Stricter password rule
- Security of authentication
  - Secure password on the Internet
  - Secure Shell
  - Extra security measures for the Internet
  - Securing the root password
- Other access controls
  - Access control lists (ACLs)
  - sudo
  - PolicyKit
  - Restricting access to some server services
  - Linux security features

When a person (or a program) requests access to the system, authentication confirms the identity to be a trusted one.

> [!WARNING] Configuration errors of PAM may lock you out of your own system. You must have a rescue CD handy or setup an alternative boot partition. To recover, boot the system with them and correct things from there.

Normal Unix authentication

Normal Unix authentication is provided by the `pam_unix`(8) module under the PAM (Pluggable Authentication Modules). Its 3 important configuration files, with "`:`" separated entries, are the following.

file	permission	user	group	description
`/etc/passwd`	`-rw-r--r--`	`root`	`root`	(sanitized) user account information
`/etc/shadow`	`-rw-r-----`	`root`	`shadow`	secure user account information
`/etc/group`	`-rw-r--r--`	`root`	`root`	group information

3 important configuration files for `pam_unix`(8)

"`/etc/passwd`" contains the following.

```
 ...
user1:x:1000:1000:User1 Name,,,:/home/user1:/bin/bash
user2:x:1001:1001:User2 Name,,,:/home/user2:/bin/bash
 ...
```

As explained in `passwd`(5), each "`:`" separated entry of this file means the following.

- Login name
- Password specification entry
- Numerical user ID
- Numerical group ID
- User name or comment field
- User home directory
- Optional user command interpreter

The second entry of "`/etc/passwd`" was used for the encrypted password entry. After the introduction of "`/etc/shadow`", this entry is used for the password specification entry.

content	meaning

(empty)	passwordless account
x	the encrypted password is in "`/etc/shadow`"

The second entry content of "`/etc/passwd`"

"`/etc/shadow`" contains the following.

```
 ...
user1:$1$Xop0FYH9$IfxyQwBe9b8tiyIkt2P4F/:13262:0:99999:7:::
user2:$1$vXGZLVbS$ElyErNf/agUDsm1DehJMS/:13261:0:99999:7:::
 ...
```

As explained in `shadow`(5), each "`:`" separated entry of this file means the following.

- Login name
- Encrypted password (The initial "`$1$`" indicates use of the MD5 encryption. The "*" indicates no login.)
- Date of the last password change, expressed as the number of days since Jan 1, 1970
- Number of days the user will have to wait before she will be allowed to change her password again
- Number of days after which the user will have to change her password
- Number of days before a password is going to expire during which the user should be warned
- Number of days after a password has expired during which the password should still be accepted
- Date of expiration of the account, expressed as the number of days since Jan 1, 1970
- ...

"`/etc/group`" contains the following.

```
group1:x:20:user1,user2
```

As explained in `group`(5), each "`:`" separated entry of this file means the following.

- Group name
- Encrypted password (not really used)
- Numerical group ID
- "," separated list of user names

> [!NOTE] "`/etc/gshadow`" provides the similar function as "`/etc/shadow`" for "`/etc/group`" but is not really used.

> [!NOTE] The actual group membership of a user may be dynamically added if "`auth optional pam_group.so`" line is added to "`/etc/pam.d/common-auth`" and set it in "`/etc/security/group.conf`". See `pam_group`(8).

> [!NOTE] The `base-passwd` package contains an authoritative list of the user and the group: "`/usr/share/doc/base-passwd/users-and-groups.html`".

Managing account and password information

Here are few notable commands to manage account information.

command	function
`getent passwd user_name`	browse account information of "`user_name`"
`getent shadow user_name`	browse shadowed account information of "`user_name`"
`getent group group_name`	browse group information of "`group_name`"
`passwd`	manage password for the account
`passwd -e`	set one-time password for the account activation
`chage`	manage password aging information

List of commands to manage account information

You may need to have the root privilege for some functions to work. See `crypt`(3) for the password and data encryption.

> [!NOTE] On the system set up with PAM and NSS as the Debian salsa machine, the content of local "`/etc/passwd`", "`/etc/group`" and "`/etc/shadow`" may not be actively used by the system. Above commands are valid even under such environment.

Good password

When creating an account during your system installation or with the `passwd`(1) command, you should choose a good password which consists of at least 6 to 8 characters including one or more characters from each of the following sets according to `passwd`(1).

- Lower case alphabetics
- Digits 0 through 9
- Punctuation marks

> [!WARNING] Do not choose guessable words for the password. Account name, social security number, phone number, address, birthday, name of your family members or pets, dictionary words, simple sequence of characters such as "12345" or "qwerty", ... are all bad choice for the password.

Creating encrypted password

There are independent tools to generate encrypted passwords with salt.

package	popcon	size	command	function
`whois`	@-@popcon1@-@	@-@psize1@-@	`mkpasswd`	over-featured front end to the `crypt`(3) library
`openssl`	@-@popcon1@-@	@-@psize1@-@	`openssl passwd`	compute password hashes (OpenSSL). `passwd`(1ssl)

List of tools to generate password

PAM and NSS

Modern Unix-like systems such as the Debian system provide PAM (Pluggable Authentication Modules) and NSS (Name Service Switch) mechanism to the local system administrator to configure his system. The role of these can be summarizes as the following.

- PAM offers a flexible authentication mechanism used by the application software thus involves password data exchange.
- NSS offers a flexible name service mechanism which is frequently used by the C standard library to obtain the user and group name for programs such as `ls`(1) and `id`(1).

These PAM and NSS systems need to be configured consistently.

The notable packages of PAM and NSS systems are the following.

package	popcon	size	description
`libpam-modules`	@-@popcon1@-@	@-@psize1@-@	Pluggable Authentication Modules (basic service)
`libpam-ldapd`	@-@popcon1@-@	@-@psize1@-@	Pluggable Authentication Module allowing LDAP interfaces
`libpam-systemd`	@-@popcon1@-@	@-@psize1@-@	Pluggable Authentication Module to register user sessions for `logind`
`libpam-doc`	@-@popcon1@-@	@-@psize1@-@	Pluggable Authentication Modules (documentation in html and text)
`libc6`	@-@popcon1@-@	@-@psize1@-@	GNU C Library: Shared libraries which also provides "Name Service Switch" service
`glibc-doc`	@-@popcon1@-@	@-@psize1@-@	GNU C Library: Manpages
`glibc-doc-reference`	@-@popcon1@-@	@-@psize1@-@	GNU C Library: Reference manual in info, pdf and html format (non-free)
`libnss-mdns`	@-@popcon1@-@	@-@psize1@-@	NSS module for Multicast DNS name resolution
`libnss-ldapd`	@-@popcon1@-@	@-@psize1@-@	NSS module for using LDAP as a naming service

List of notable PAM and NSS systems

- "The Linux-PAM System Administrators' Guide" in `libpam-doc` is essential for learning PAM configuration.
- "System Databases and Name Service Switch" section in `glibc-doc-reference` is essential for learning NSS configuration.

> [!NOTE] You can see more extensive and current list by "`aptitude search 'libpam-|libnss-'`" command. The acronym NSS may also mean "Network Security Service" which is different from "Name Service Switch".

> [!NOTE] PAM is the most basic way to initialize environment variables for each program with the system wide default value.

Under systemd, `libpam-systemd` package is installed to manage user logins by registering user sessions in the `systemd` control group hierarchy for logind. See `systemd-logind`(8), `logind.conf`(5), and `pam_systemd`(8).

Configuration files accessed by PAM and NSS

Here are a few notable configuration files accessed by PAM and NSS.

configuration file	function
`/etc/pam.d/program_name`	set up PAM configuration for the "`program_name`" program; see `pam`(7) and `pam.d`(5)
`/etc/nsswitch.conf`	set up NSS configuration with the entry for each service. See `nsswitch.conf`(5)
`/etc/nologin`	limit the user login by the `pam_nologin`(8) module
`/etc/securetty`	limit the tty for the root access by the `pam_securetty`(8) module
`/etc/security/access.conf`	set access limit by the `pam_access`(8) module
`/etc/security/group.conf`	set group based restraint by the `pam_group`(8) module
`/etc/security/pam_env.conf`	set environment variables by the `pam_env`(8) module
`/etc/environment`	set additional environment variables by the `pam_env`(8) module with the "`readenv=1`" argument
`/etc/default/locale`	set locale by `pam_env`(8) module with the "`readenv=1 envfile=/etc/default/locale`" argument (Debian)
`/etc/security/limits.conf`	set resource restraint (ulimit, core, ...) by the `pam_linits`(8) module
`/etc/security/time.conf`	set time restraint by the `pam_time`(8) module
`/etc/systemd/logind.conf`	set `systemd` login manager configuration (see `logind.conf`(5) and `systemd-logind.service`(8))

List of configuration files accessed by PAM and NSS

The limitation of the password selection is implemented by the PAM modules, `pam_unix`(8) and `pam_cracklib`(8). They can be configured by their arguments.

> [!

> [!TIP] PAM modules use suffix "`.so`" for their filenames.

The modern centralized system management

The modern centralized system management can be deployed using the centralized Lightweight Directory Access Protocol (LDAP) server to administer many Unix-like and non-Unix-like systems on the network. The open source implementation of the Lightweight Directory Access Protocol is OpenLDAP Software.

The LDAP server provides the account information through the use of PAM and NSS with `libpam-ldapd` and `libnss-ldapd` packages for the Debian system. Several actions are required to enable this (I have not used this setup and the following is purely secondary information. Please read this in this context.).

- You set up a centralized LDAP server by running a program such as the stand-alone LDAP daemon, `slapd`(8).
- You change the PAM configuration files in the "`/etc/pam.d/`" directory to use "`pam_ldap.so`" instead of the default "`pam_unix.so`".
- You change the NSS configuration in the "`/etc/nsswitch.conf`" file to use "`ldap`" instead of the default ("`compat`" or "`file`").
- You must make `libpam-ldapd` to use SSL (or TLS) connection for the security of password.
- You may make `libnss-ldapd` to use SSL (or TLS) connection to ensure integrity of data at the cost of the LDAP network overhead.
- You should run `nscd`(8) locally to cache any LDAP search results in order to reduce the LDAP network traffic.

See documentations in `nsswitch.conf`(5), `pam.conf`(5), `ldap.conf`(5), and "`/usr/share/doc/libpam-doc/html/`" offered by the `libpam-doc` package and "`info libc 'Name Service Switch'`" offered by the `glibc-doc` package.

Similarly, you can set up alternative centralized systems with other methods.

- Integration of user and group with the Windows system.
  - Access Windows domain services by the `winbind` and `libpam_winbind` packages.
  - See `winbindd`(8) and Integrating MS Windows Networks with Samba.
- Integration of user and group with the legacy Unix-like system.
  - Access NIS (originally called YP) or NIS+ by the `nis` package.
  - See The Linux NIS(YP)/NYS/NIS+ HOWTO.

"Why GNU su does not support the wheel group"

This is the famous phrase at the bottom of the old "`info su`" page by Richard M. Stallman. Not to worry: the current `su` command in Debian uses PAM, so that one can restrict the ability to use `su` to the `root` group by enabling the line with "`pam_wheel.so`" in "`/etc/pam.d/su`".

Stricter password rule

Installing the `libpam-cracklib` package enables you to force stricter password rule.

On a typical GNOME system which automatically installs `libpam-gnome-keyring`, "`/etc/pam.d/common-password`" looks like:

```
# here are the per-package modules (the "Primary" block)
password    requisite           pam_cracklib.so retry=3 minlen=8 difok=3
password    [success=1 default=ignore]  pam_unix.so obscure use_authtok try_first_pass
yescrypt
# here's the fallback if no module succeeds
password    requisite           pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
password    required            pam_permit.so
# and here are more per-package modules (the "Additional" block)
password    optional    pam_gnome_keyring.so
# end of pam-auth-update config
```

Security of authentication

> [!NOTE] The information here **may not be sufficient** for your security needs but it should be a **good start**.

Secure password on the Internet

Many popular transportation layer services communicate messages including password authentication in the plain text. It is very bad idea to transmit password in the plain text over the wild Internet where it can be intercepted. You can run these services over "Transport Layer Security" (TLS) or its predecessor, "Secure Sockets Layer" (SSL) to secure entire communication including password by the encryption.

insecure service name	port	secure service name	port
`www (http)`	80	`https`	443
`smtp (mail)`	25	`ssmtp (smtps)`	465
`ftp-data`	20	`ftps-data`	989
`ftp`	21	`ftps`	990
`telnet`	23	`telnets`	992
`imap2`	143	`imaps`	993
`pop3`	110	`pop3s`	995
`ldap`	389	`ldaps`	636

List of insecure and secure services and ports

The encryption costs CPU time. As a CPU friendly alternative, you can keep communication in plain text while securing just the password with the secure authentication protocol such as "Authenticated Post Office Protocol" (APOP) for POP and "Challenge-Response Authentication Mechanism MD5" (CRAM-MD5) for SMTP and IMAP. (For sending mail messages over the Internet to your mail server from your mail client, it is recently popular to use new message submission port 587 instead of traditional SMTP port 25 to avoid port 25 blocking by the network provider while authenticating yourself with CRAM-MD5.)

The traditional password-based authentication still suffers inherent security issues such as password reuse across multiple websites. See other authentication methods for the secure solution: Time-based one-time password (TOTP), Universal 2nd Factor (U2F), Web Authentication (WebAuthn, commonly referred as "passkey")

Secure Shell

The Secure Shell (SSH) program provides secure encrypted communications between two untrusted hosts over an insecure network with the secure authentication. It consists of the OpenSSH client, `ssh`(1), and the OpenSSH daemon, `sshd`(8). This SSH can be used to tunnel an insecure protocol communication such as POP and X securely over the Internet with the port forwarding feature.

The client tries to authenticate itself using host-based authentication, public key authentication, challenge-response authentication, or password authentication. The use of public key authentication enables the remote password-less login. See ???.

Extra security measures for the Internet

Even when you run secure services such as Secure Shell (SSH) and Point-to-point tunneling protocol (PPTP) servers, there are still chances for the break-ins using brute force password guessing attack etc. from the Internet. Use of the firewall policy (see ???) together with the following security tools may improve the security situation.

package	popcon	size	description
`knockd`	@-@popcon1@-@	@-@psize1@-@	small port-knock daemon `knockd`(1) and client `knock`(1)
`fail2ban`	@-@popcon1@-@	@-@psize1@-@	ban IPs that cause multiple authentication errors
`libpam-shield`	@-@popcon1@-@	@-@psize1@-@	lock out remote attackers trying password guessing

List of tools to provide extra security measures

Securing the root password

To prevent people to access your machine with root privilege, you need to make following actions.

- Prevent physical access to the system storage device (HDD / SSD / ...)
- Lock UEFI/BIOS and prevent booting from the removable media
- Set password for GRUB interactive session
- Lock GRUB menu from editing

With physical access to the system storage device, resetting the password is relatively easy with following steps.

1. Move the system storage device to a PC with USB bootable UEFI/BIOS.
2. Boot system with a rescue media (see ???).
3. Mount root partition with read/write access.
4. Edit "`/etc/passwd`" in the root partition and make the second entry for the `root` account empty.

If you have edit access to the GRUB menu entry (see ???) for `grub-rescue-pc` at boot time, it is even easier with following steps.

1. Boot system with the kernel parameter changed to something like "`root=/dev/sda6 rw init=/bin/sh`".
2. Edit "`/etc/passwd`" and make the second entry for the `root` account empty.
3. Reboot system.

The root shell of the system is now accessible without password.

> [!NOTE] Once one has root shell access, he can access everything on the system and reset any passwords on the system. Further more, he may compromise password for all user accounts using brute force password cracking tools such as `john` and `crack` packages (see ???). This cracked password may lead to compromise other systems.

The only reasonable software solution to avoid all these concerns is to use software encrypted root partition (or "`/etc`" partition) using dm-crypt and initramfs (see ???). You always need password to boot the system, though.

Other access controls

There are access controls to the system other than the password based authentication and file permissions.

> [!NOTE] See ??? for restricting the kernel secure attention key (SAK) feature.

Access control lists (ACLs)

ACLs are a superset of the regular permissions as explained in ???.

You encounter ACLs in action on modern desktop environment. When a formatted USB storage device is auto mounted as, e.g., "`/media/penguin/USBSTICK`", a normal user `penguin` can execute:

```
$ cd /media/penguin
$ ls -la
```

```
total 16
drwxr-x---+ 1 root    root    16 Jan 17 22:55 .
drwxr-xr-x  1 root    root    28 Sep 17 19:03 ..
drwxr-xr-x  1 penguin penguin 18 Jan  6 07:05 USBSTICK
```

"`+`" in the 11th column indicates ACLs are in action. Without ACLs, a normal user `penguin` shouldn't be able to list like this since `penguin` isn't in `root` group. You can see ACLs as:

```
 $ getfacl .
# file: .
# owner: root
# group: root
user::rwx
user:penguin:r-x
group::---
mask::r-x
other::---
```

Here:

- "`user::rwx`", "`group::---`", and "`other::---`" correspond to the regular owner, group, and other permissions.
- The ACL "`user:penguin:r-x`" allows a normal user `penguin` to have "`r-x`" permissions. This enabled "`ls -la`" to list directory content.
- The ACL "`mask::r-x`" sets the upper bound of permissions.

See "POSIX Access Control Lists on Linux", `acl`(5), `getfacl`(1), and `setfacl`(1) for more.

sudo

`sudo`(8) is a program designed to allow a sysadmin to give limited root privileges to users and log root activity. `sudo` requires only an ordinary user's password. Install `sudo` package and activate it by setting options in "`/etc/sudoers`". See configuration example at "`/usr/share/doc/sudo/examples/sudoers`" and ???.

My usage of `sudo` for the single user system (see ???) is aimed to protect myself from my own stupidity. Personally, I consider using `sudo` a better alternative than using the system from the root account all the time. For example, the following changes the owner of "`some_file`" to "`my_name`".

```
$ sudo chown my_name some_file
```

Of course if you know the root password (as self-installed Debian users do), any command can be run under root from any user's account using "`su -c`".

PolicyKit

PolicyKit is an operating system component for controlling system-wide privileges in Unix-like operating systems.

Newer GUI applications are not designed to run as privileged processes. They talk to privileged processes via PolicyKit to perform administrative operations.

PolicyKit limits such operations to user accounts belonging to the `sudo` group on the Debian system.

See `polkit`(8).

Restricting access to some server services

For system security, it is a good idea to disable as much server programs as possible. This becomes critical for network servers. Having unused servers, activated either directly as daemon or via super-server program, are considered security risks.

Many programs, such as `sshd`(8), use PAM based access control. There are many ways to restrict access to some server services.

- configuration files: "`/etc/default/program_name`"
- Systemd service unit configuration for daemon
- PAM (Pluggable Authentication Modules)
- "`/etc/inetd.conf`" for super-server
- "`/etc/hosts.deny`" and "`/etc/hosts.allow`" for TCP wrapper, `tcpd`(8)
- "`/etc/rpc.conf`" for Sun RPC
- "`/etc/at.allow`" and "`/etc/at.deny`" for `atd`(8)
- "`/etc/cron.allow`" and "`/etc/cron.deny`" for `crontab`(1)
- Network firewall of netfilter infrastructure

See ???, Configuration files accessed by PAM and NSS, and ???.

> [!TIP] Sun RPC services need to be active for NFS and other RPC based programs.

> [!TIP] If you have problems with remote access in a recent Debian system, comment out offending configuration such as "`ALL: PARANOID`" in "`/etc/hosts.deny`" if it exists. (But you must be careful on security risks involved with this kind of action.)

Linux security features

Linux kernel has evolved and supports security features not found in traditional UNIX implementations.

Linux supports extended attributes which extend the traditional UNIX attributes (see `xattr`(7)).

Linux divides the privileges traditionally associated with superuser into distinct units, known as `capabilities`(7), which can be independently enabled and disabled. Capabilities are a per-thread attribute since kernel version 2.2.

The Linux Security Module (LSM) framework provides a mechanism for various security checks to be hooked by new kernel extensions. For example:

- AppArmor
- Security-Enhanced Linux (SELinux)
- Smack (Simplified Mandatory Access Control Kernel)
- Tomoyo Linux

Since these extensions may tighten privilege model tighter than the ordinary Unix-like security model policies, even the root power may be restricted. You are advised to read the Linux Security Module (LSM) framework document at kernel.org.

Linux namespaces wrap a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource. Changes to the global resource are visible to other processes that are members of the namespace, but are invisible to other processes. Since kernel version 5.6, there are 8 kinds of namespaces (see `namespaces`(7), `unshare`(1), `nsenter`(1)).

As of Debian 11 Bullseye (2021), Debian uses unified cgroup hierarchy (a.k.a. cgroups-v2).

Usage examples of namespaces with cgroups to isolate their processes and to allow resource control are:

- Systemd. See ???.
- Sandbox environment. See ???.
- Linux containers such as Docker, LXC. See ???.

These functionalities can't be realized by Normal Unix authentication. These advanced topics are mostly out-of-scope for this introductory document.

□□□ #2
□ □□□ □□□ 2026-10-31 13:07:39 UTC
□ □□□ □□□ 2026-10-31 13:07:41 UTC