Linux Hardening

Research and apply hardening techniques to Linux systems.

What Is Linux?

Linux at its most basic is a *kernel*; a low-level software interface between the hardware of a computer and higher-level software. The Linux kernel is not an operating system by itself but serves as a key building block for hundreds of different Linux distros (short for 'distributions'): operating systems that are built on top of Linux.

There is no definitive distro of Linux, so keep that in mind throughout this guide. Each distro has its own quirks, and the open nature of many of them means that new versions can be created at a rapid rate, fixing old security issues and creating new ones. Research the specific distribution and version that you're interested in hardening!

Basic Linux Hardening Checklist

This section uses the Linux command line, but you don’t need to be familiar with it to complete this section.

A few tips:

- Take the time to research each change before implementing it! Understand the purpose of each change, and what effects it will have.
- Keep in mind that all the instructions below were written for Ubuntu `20.04.2.0` and may not translate perfectly to other versions or distros.

Ensure `root` Account has a Strong Password

The `root` account has the most privileges on a Linux system so it should be the most secure!

- In the terminal, type `sudo passwd root`. Then, you will authenticate with your password and enter the new password you would like to use for the root account. Don’t forget this new password!

```
~$ sudo passwd root
[sudo] enter password for admin: ***************
New password: ********************
Retype new password: ********************
passwd: Password updated successfully
```

Create User Accounts for Everyday Use

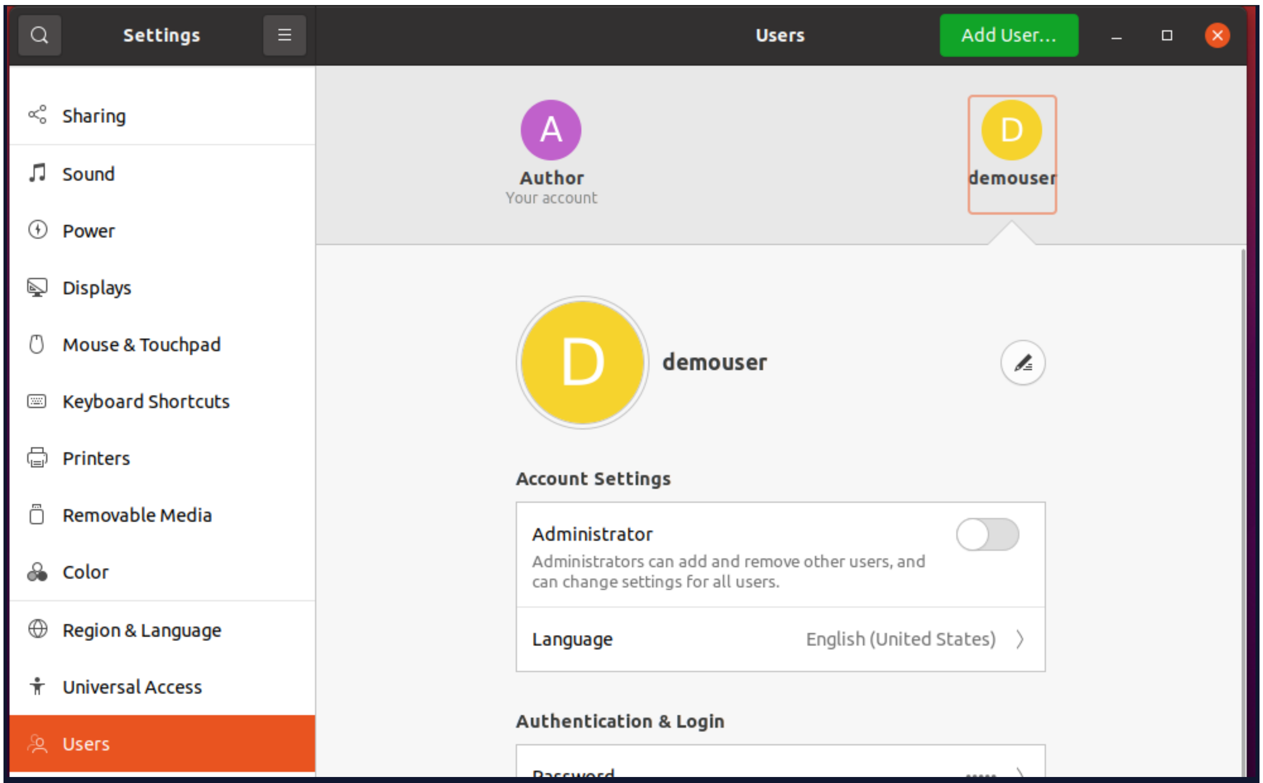
While it may be tempting to always log in as the `root` user, doing so can make it easier for attackers to compromise a system. Instead, log in with a standard user account, and use the `sudo` command for specific command-line operations that require elevated permissions.

- Accounts can be created from the terminal, but it’s simpler to create them from the Settings application.
- Open the Settings application, and navigate to the “Users” section in the sidebar.
- Unlock the user settings with your password.
- You can now create new accounts with the “Add User” button.

Limit `sudo` Privileges

`sudo` provides a way for user accounts to execute commands as root. This is a very powerful tool, and it can be very dangerous in the hands of attackers and inexperienced users. If an account doesn't need to use `sudo`, make sure it can't use `sudo`.

- Open the Settings application, and navigate to the "Users" section in the sidebar.
- Unlock the user settings with your password.
- Grant or revoke `sudo` privileges using the "Administrator" toggle for each account.



Account Settings

Administrator
Administrators can add and remove other users, and can change settings for all users.

Language English (United States)

Authentication & Login

Check Which Services are Running at Startup

Make sure there aren't any services running automatically that shouldn't be.

- In your terminal, use `service --status-all` to see what services have started and stopped:

```
~$ service --status-all
[ + ]  apparmor
[ - ]  sudo
```

Services marked with a `+` are currently running, and services marked with a `-` are stopped.

Uninstall Unused Software

If you're not using a piece of software, it's generally a good idea to remove it.

- You can view what packages are installed using the command `apt list --installed`.

```
~$ apt list --installed
```

- You can remove packages using the `apt purge` command, which removes the package, along with its configuration.

```
~$ apt purge <package>
```

- You can remove all unused dependencies (packages that were only used by other packages) by using the `apt autoremove` command.

```
~$ apt autoremove
```

Intermediate Linux Hardening

This section contains more complex hardening techniques that have lots of steps, require more familiarity with the Linux command line, or can cause unforeseen problems if done incorrectly. For the best experience in this section, you should be comfortable using the command line to create and edit configuration files.

If you're not familiar with the command line, we strongly recommend you learn how to use the command line in Codecademy's Learn the Command Line course before attempting this section.

This section involves making changes to the `sudoers` file, which is critical to the functionality and security of Linux. When making changes to the `sudoers` file:

- *NEVER use a standard text editor to edit the `sudoers` file.*
- *ALWAYS use `visudo` to edit the `sudoers` file, as it prevents the file from being saved with invalid formatting.*

Limit `sudo` Access for Non-Administrator Accounts

If a user needs `sudo` privileges for something specific, it's possible to grant limited access to `sudo` for use with specific commands.

- Make sure the user's `sudo` privileges are disabled through the method described in the "Limit `sudo` Privileges" section.
- Have the following information ready:
  - The username of the account you want to grant limited permissions to.

    - The path to the application or applications you want to allow the account to access. (This can be found easily using the command `which`.)
- Open a terminal, and run the command `sudo su` to log in as root (due to the way permissions are set up, only using `sudo` for the following commands may not work)

```
~$ sudo su
Password:
sh-3.2#
```

- Create a new file in the `/etc/sudoers.d` directory. (The name shouldn't matter, but you can keep things organized by naming it after the user whose permissions you're changing)

```
sh-3.2# cd /etc/sudoers.d
sh-3.2# touch exampleuser
```

- Open the file in your text editor of choice, and add the following line to the file: `<username> ALL=(root) <path>`. You will substitute `<username>` for the username to which you are granting permissions and `<path>` with the path to the application. For example:

  ```
  exampleuser ALL=(root) /usr/bin/whoami
  ```

  We will let the user "exampleuser" run the `whoami` as root using `sudo`.
- Save the file, and exit the terminal.

Be aware that some programs can be used for what is known as *privilege escalation*, where a user is able to grant themselves additional permissions that they shouldn't have. For example, allowing a user to run a terminal using `sudo` effectively allows them to run *any* program using `sudo`.

Require Passwords for `sudo` Use

Some distros are configured to have a set time limit on using `sudo` without re-entering the password. While convenient, this also opens up the possibility of an attacker waiting for you to use `sudo`, then taking advantage of that time period to run their own commands without authentication.

- Open the sudoers file with the command `sudo visudo`.
    - The default text editor for this is Vim, which can be difficult to use. You can run `visudo` with an easier editor by using `sudo EDITOR=nano visudo`.
- You should see a line that looks something like this:

  ```
  Defaults    env_reset
  ```

Edit that line so that it looks like this:

```
Defaults    env_reset, timestamp_timeout=0
```

- Save and exit the editor.

Ensure System Updates are Applied Regularly

- Libre-software has a guide for setting up automatic software updates for Ubuntu. The guide is written for the server version of the distro, but it works for the desktop version as well.

Ensure the SSH server is Disabled or Hardened

- SSH allows you to gain access to a remote computer running an SSH server, but, if you don't need to do so, it's best to just disable the SSH server. If you want to learn more about hardening SSH, Tecmint has a guide to hardening OpenSSH servers.

Enforce Strong Passwords

- If you share your system with other people, it's a good idea to enforce strong passwords for accounts. Linux Hint has a guide to enforcing password policies on Ubuntu using PAM.

Advanced Hardening

This section is for hardening that requires a good understanding of your system and digital environment. You will need to research what techniques and implementations are right for you, as well as how to apply them.

Enable System Startup Security

Depending on when the hardware was issued, BIOS or UEFI is the first piece of software that runs when you start your computer. If it is tampered with, then all subsequent programs, such as the operating system, can be compromised. You can start by password-protecting the configuration, disabling unused hardware ports, and restricting devices the computer can boot from, and more.

The NSA published a report in 2017 outlining best options and recommendations for hardening this part of your computer.

Encrypt System and User Files

*This generally needs to be set up when the OS is first installed, especially to use full-disk encryption. Encryption will cause a performance loss on most computers because the CPU has to spend time decrypting files, but helps secure the system against physical access.

Harden Your Kernel

- The kernel is the core of the OS, and vulnerabilities within it can leave the whole OS open to attack. The kernel is updated regularly, so there aren't likely to be any glaring flaws, but there may be ways you can further secure your kernel. The Arch Linux Wiki's security page has a section on kernel hardening.

Set Up the Firewall

- The Linux kernel has a built-in firewall, configured by the `iptables` program, but `iptables` has a reputation for being difficult to use. However, programs UFW (short for "Uncomplicated Firewall") exist as a frontend for `iptables`, making it much easier to configure your firewall.

Additional Resources

- The Arch Linux Wiki's security page has many recommendations and best practices. While aimed at Arch Linux, many of them will be applicable to other distros as well.
- Linux Hint's checklist for Linux hardening is a short list of best practices for desktop and server hardening.
- Computerworld's article on Linux server hardening provides a more detailed checklist to hardening server environments