DATE: 2025-01-19
INFORMATION CLASS: CUSTOMER CONFIDENTIAL

| | |
|---|---|
| **CUSTOMER** | VERYSECURE AB |
| **SUBJECT** | UBUNTU LINUX 24.04 DESKTOP |
| **DOCUMENT** | HARDENING GUIDE |
| **GROUP 4** | ALLEN CAMILLE MUCO<br>HUGO POLSTAM<br>ISAC SAFAROV<br>NICHOLAS JOHN STEVENS<br>SEBASTIAN EKEDAHL |

Security AB
orgnr. 1234-567890
www.example.com

# Table of contents

## Introduction

The goal of this guide is to provide a comprehensive hardening framework to enhance the security and resilience of Ubuntu Linux 24.04 Desktop systems.

The recommendations in this guide are based on industry best practices, including the CIS Benchmarks, NIST guidelines, and Ubuntu's official documentation. They cover key areas of system hardening, such as secure filesystem configurations, host-based firewall settings, user account management, password quality checks, and application control. Additionally, it emphasizes network security measures, logging, and monitoring to provide a multi-layered defense strategy.

These measures are intended to be implemented in a standard enterprise environment where some local network file sharing, printer usage, and communication via chat and email take place.

By following the steps outlined, VerySecure AB can establish a secure baseline for Ubuntu systems, reducing vulnerabilities and enabling effective response to emerging threats.

# 1. Filesystem Partitioning

These recommendations are intended to enhance the security and integrity of the system by applying appropriate mount options to critical filesystems during the initial installation.

## 1.1 Create Partitions

During the partitioning step of installation, create separate partitions with these file schemes. Set partition sizes within the recommended ranges for a workstation listed below:

| | | | | | |
|---|---|---|---|---|---|
| /boot/efi | FAT32 | 500MB | /var | ext4 | 10–20G |
| /boot | ext4 | 1G | /var/tmp | ext4 | 5–10G |
| / (root) | ext4 | 25–50G | /var/log | ext4 | 10–50G |
| /tmp | tmpfs | 4–8G | /var/log/audit | ext4 | 5–10G |
| /home | ext4 | 20–25G | | | |

## 1.2 Set Mount Options

Once installation is complete the mount options can be updated via bash. Open `/etc/fstab` in a text editor and ensure these additional mount options are included (see Example 1 below):

| | | | | |
|---|---|---|---|---|
| /tmp | nodev, nosuid, noexec | /var/tmp | nodev, nosuid, noexec | |
| /dev/shm | nodev, nosuid, noexec | /var/log | nodev, nosuid | |
| /home | nodev, nosuid | /var/log/audit | nodev, nosuid, noexec | |
| /var | nodev, nosuid | | | |

Reload system daemon after making the changes:

```
$ systemctl daemon-reload
```

Remount each filesystem partition individually to apply changes and verify prior to reboot:

```
$ mount -o remount /tmp
```
*(repeat for each partition with changed mount options)*

## 1.3 Verify Layout

Review the partition table to ensure all mount points and options are correctly set. List block devices, filesystems, and mount points and verify mount options once more before rebooting the system:

```
$ lsblk -f
$ findmnt | grep -E "/tmp|/dev/shm|/var|/home"
$ cat /etc/ftstab
```

```
# /etc/fstab: static file system information.
# Use 'blkid' to print the universally unique identifier for a device;
# this may be used with UUID= as a more robust way to name devices. See fstab(5).
#
# <file system>     <mount point>   <type>   <options>                          <dump> <pass>
# Primary file system
UUID=<device uuid>  /boot/efi       vfat     defaults                             0      1
UUID=<device uuid>  /boot           ext4     defaults                             0      1
UUID=<device uuid>  /               ext4     defaults                             0      1

# Temporary file systems
UUID=<device uuid>  /tmp            ext4     defaults,nodev,nosuid,noexec         0      1
tmpfs               /dev/shm        tmpfs    defaults,nodev,nosuid,noexec         0      0

# Home and var partitions
UUID=<device uuid>  /home           ext4     defaults,nodev,nosuid                0      1
UUID=<device uuid>  /var            ext4     defaults,nodev,nosuid                0      1
UUID=<device uuid>  /var/tmp        ext4     defaults,nodev,nosuid,noexec         0      1
UUID=<device uuid>  /var/log        ext4     defaults,nodev,nosuid                0      1
UUID=<device uuid>  /var/log/audit  ext4     defaults,nodev,nosuid,noexec         0      1
```

*Example 1: Use the* `blkid` *bash command to print the UUIDs in order to more robustly name devices.*

## 2. Host-Based Firewall Configuration

To avoid conflicts, ensure that only one firewall utility is configured and active on the system. The Uncomplicated Firewall (UFW) is a frontend for iptables and provides a simplified framework for managing netfilter. UFW depends on the iptables package, so both must be installed on the system. This guide assumes an unconfigured UFW for its implementation (`ufw reset`).

### 2.1. Ensure a single firewall is installed

Verify that UFW is installed and remove iptables-persistent if it is present on the system. Running both UFW and the services included in the iptables-persistent package may lead to conflict. *Note: Rules should be ordered so that allow rules come before deny rules.*

### 2.2 Enable UFW service

When running ufw enable, UFW will flush its chains. It may drop existing connections (eg ssh). Run the following command before running ufw enable. The rules will still be flushed, but the ssh port will be open after enabling the firewall.

```
$ ufw allow proto tcp from any to any port 22
```

Enable ufw service:

```
$ systemctl unmask ufw.service
$ systemctl --now enable ufw.service
$ ufw enable
```

*Note: Configuration of a live system's firewall directly over a remote connection can result in being locked out.*

### 2.3 Configure loopback traffic

Configure the loopback interface to accept traffic. Configure all other interfaces to deny traffic to the loopback network (127.0.0.0/8 for IPv4 and ::1/128 for IPv6).

```
$ ufw allow in on lo                    $ ufw deny in from 127.0.0.0/8
$ ufw allow out on lo                   $ ufw deny in from ::1
```

### 2.4 Configure outbound connections

If rules are not in place for new outbound connections all packets will be dropped by the default policy preventing network usage. The rules below are appropriate for a standard endpoint.

```
To                         Action      From
--                         ------      ----
80/tcp                     ALLOW OUT   Anywhere       -- HTTP (web browsing)
443                        ALLOW OUT   Anywhere       -- HTTPS (secure web browsing)
8080                       ALLOW OUT   Anywhere       -- HTTP (alternative port)
123/udp                    ALLOW OUT   Anywhere       -- NTP (time synchronization)
53                         ALLOW OUT   Anywhere       -- DNS (domain name system)
853                        ALLOW OUT   Anywhere       -- DNS over TLS (secure DNS)
22/tcp                     ALLOW OUT   Anywhere       -- SSH (remote login and file transfers)

10.0.0.0/24 36727          ALLOW OUT   Anywhere       -- File sharing services
10.0.0.0/24 1900/udp       ALLOW OUT   Anywhere       -- File sharing/media streaming via DLNA/UPnP
multicast-grp-ip 1900/udp  ALLOW OUT   Anywhere       -- Device discovery on local networks

80/tcp (v6)                ALLOW OUT   Anywhere (v6)  -- HTTP (IPv6 web browsing)
443 (v6)                   ALLOW OUT   Anywhere (v6)  -- HTTPS (IPv6 secure web browsing)
8080 (v6)                  ALLOW OUT   Anywhere (v6)  -- HTTP (IPv6 alternative port)
123/udp (v6)               ALLOW OUT   Anywhere (v6)  -- NTP (IPv6 time synchronization)
53 (v6)                    ALLOW OUT   Anywhere (v6)  -- DNS (IPv6 domain name system)
853 (v6)                   ALLOW OUT   Anywhere (v6)  -- DNS over TLS (IPv6 secure DNS)
22/tcp (v6)                ALLOW OUT   Anywhere (v6)  -- SSH (IPv6 remote login and file transfers)
```

*Ex 2.4 outbound rule syntax:* `ufw allow out from any to any proto tcp port 8080`

## 2.5 Configure Inbound Connections

Ensure a rule exists for every open port on non-loopback addresses. Check socket stats for current ports status:

```
$ ss -tulpn
```

The inbound rules below are appropriate for a standard endpoint:

```
To                  Action      From
--                  ------      ----
22/tcp              ALLOW IN    Anywhere                   -- SSH: could be limited to specific IPs
22/tcp (v6)         ALLOW IN    Anywhere (v6)              -- SSH IPv6
3702/udp            ALLOW IN    10.0.0.0/24                -- Web Services Dynamic Discovery
42369/udp           ALLOW IN    10.0.0.0/24                -- Web Services Dynamic Discovery
1900/udp            ALLOW IN    10.0.0.0/24                -- File sharing/media streaming via DLNA/UPnP
1900/udp            ALLOW IN    multicast-grp-ip           -- Device discovery on local networks
36727/udp           ALLOW IN    10.0.0.0/24                -- File sharing services on local network
36727/tcp           ALLOW IN    192.168.1.0/24             -- File sharing services on local network
Anywhere            DENY IN     127.0.0.0/8                -- Loopback
```

*Example 2.5 inbound syntax:* `ufw allow from 10.0.0.0/24 to any proto udp port 1900`

## 2.6 Ensure UFW default deny firewall policy

A default deny policy on connections ensures that any unconfigured network usage will be rejected. Any port or protocol without an explicit allow before the default deny is applied will be blocked.

Set default deny policy for incoming and outgoing connections:

```
$ ufw default deny incoming
$ ufw default deny outgoing
$ ufw default deny routed
```

Enable logging for firewall events:

```
ufw logging on
```

## 2.7 Verify Configuration

Reload UFW and check the active rules:

```
$ ufw reload
$ ufw status verbose
```

Sample expected output:

```
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), disabled (routed)

To                         Action      From
--                         ------      ----
---------- <RULES CONTINUE HERE> ----------
```

*Example output showing correctly configured status, logging, and defaults.*

# 3. Application Control

Application control is a security strategy that aims to ensure only trusted and approved applications run on a system. By implementing application control, organizations can prevent malware and unwanted applications from affecting the system's security and performance. This is achieved by applying rules and policies that govern which applications can be installed and executed on Ubuntu Linux systems[4].

## 3.1 Whitelisting

Whitelisting on Ubuntu Linux is a security method distinct from traditional security technologies like antivirus software, which utilize blacklists to block known malicious activities and permit all else. In contrast, whitelisting technologies are designed to permit known good activities and block all others1. This means that only specifically approved and trusted applications are allowed to run, while all other programs are blocked by default. By implementing whitelisting, organizations can reduce the risk of malware or unwanted programs running on the system[4].

**Key aspects of whitelisting on Ubuntu Linux:**

- Approval Process: Administrators create a list of allowed applications based on their reliability and security.
- Regular Updates: The whitelist must be regularly updated to include new trusted applications and updates.
- Security Enhancement: By restricting execution to only trusted applications, the risk of security incidents is minimized.

Whitelisting can be implemented using security tools such as AppArmor on Ubuntu Linux. AppArmor works by assigning security profiles to each application and restricting their access to system resources[5].

```
$ apt-get install apparmor apparmor-utils
```

AppArmor profiles define rules for each application and found here: `/etc/apparmor.d/`

Activate a profile: `$ apparmor_parser -r /etc/apparmor.d/profile_name`

## 3.2 Bloatware

Bloatware refers to unwanted pre-installed applications that come with the operating system or other software packages. These applications can consume valuable system resources and sometimes pose security risks if not regularly updated. On Ubuntu Linux, application control can be used to identify and remove bloatware, improving the system's performance and security.

**Examples of bloatware on Ubuntu:**

- Snap Packages: Canonical's Snap technology is controversial and can be resource-heavy. Many users prefer alternatives like Flatpak[6].
- Unnecessary Software: Programs that are not used regularly, such as various games, screensavers, and system utilities.
- System Tools and Drivers: various system tools and drivers that are not be needed by users[7].

**How to remove bloatware:**

- Remove Snap Packages: Use `$ sudo snap remove –purge` to remove Snap packages.
- Remove unecessary software: Use `$ sudo apt-get remove` to uninstall unrequired packages.
- Disable unnecessary services that start on boot using `systemctl` or other tools.

**Validating bloatware removal:**

- Check Installed Packages: Use the command `$ dpkg --list` to list all installed packages and verify that the bloatware has been removed.
- Use `systemctl` to list all running services and verify that unnecessary services are disabled.

## 4. Account Management

Account management involves creating, maintaining, and securing user accounts to ensure only authorized access to the system. Implementing least privilege principles and strong password policies are fundamental for enhancing system security.

### 4.1 Principal of Least Privilege

The principle of least privilege means that both users and processes should obtain the minimal level of access necessary to perform their tasks.

1. Create user accounts with limited privileges. A user should never have more access than necessary to perform their tasks.
2. Minimize Root Access. No account should have root access at all times. Root access should be granted only when necessary and revoked immediately after the task is completed.
3. Regular Audits: Review user accounts access levels regularly to make sure what the appropriate level of access are in use.

### 4.2 Password Policies

Password policies ensure that only strong, unique passwords are used and maintained securely.

1. Strong Passwords: Use pam_pwquality module to require at least 15 characters, including a mix of uppercase, lowercase, numbers, and special characters.
2. Regular Changes: Enforce periodic password changes every 90 days. Set automatic notifications to users 14 days before the required password change. Users must contact the IT-department if the password change was not made in time.
3. No Reuse: Prevent reuse of previous passwords. Set the history limit to the last 5 passwords.
4. Multi-Factor Authentication: Use MFA where it is appropriate.
5. Require use of 1Pass password manager to handle all passwords.
6. Account Lockout: Temporarily lock accounts after 3 failed login attempts.

### 4.3 Validation

Ensure that the required packages `libpam-pwquality` and `cracklib-runtime` are available in your Ubuntu repositories. Use `$ apt install <package_names>` to install them.

Validate copnfigurations and settings in the files below:

```
/etc/login.defs
/etc/security/pwquality.conf
/etc/pam.d/common-password
```

# 5. Network Security

## 5.1 Unused services

There are many services enabled and running in the background that can be unnecessary depending on the type of system you are running. Therefore, it's best to check what services are running on the machine and evaluate their importance.

Get a list of all enabled services:

```
$ systemctl list-unit-files --type=service --state=enabled
```

Get descriptions of what the services do, include `--all` flag.

Check services currently running on the machine:

```
$ systemctl list-units --type=service --state=running
```

After determining which services are not required on the system, disable a service before you remove it to prevent it from running in the background:

```
$ systemctl disable <service>
```

Stop process to prevent it from running in current session:

```
$ systemctl stop <service>
```

Refresh the systemd daemon to make sure the changes take effect:

```
$ systemctl daemon-reload
```

Recommended services to turn off depending on the endpoint requirements: bluetooth, sshd, telnet-server, rsh-server, cups, rpcbind, smbd.

## 5.2 Routing and packet filtering

Routing and packet filtering settings are changed by altering the paramaters in "/etc/sysctl.conf".

```
# /etc/sysctl.conf
#
# Disable IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1

# Diable ip forwarding
net.ipv4.ip_forward = 0

# Disable packet redirect sending
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0

# Ignore broadcast icmp requests
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

*Routing and packet filtering settings are changed by altering the paramaters in* `/etc/sysctl.conf`.

## 5.3 Close unused ports

List the open and listening ports with:

```
$ ss -tulpn
```

You can close open ports using UFW firewall:

```
$ ufw deny [port number]
```

Ports that are recommended to close:

`Port 21 (FTP)`. File Transfer Protocol is unsafe partially because it does not encrypt its traffic.

`Port 23 (Telnet)`. Telnet is an old and outdated alternative to ssh.

## 5.4 VPN recommendations

Use a strong encryption protocol like AES-256. Enable a kill switch to block all network traffic if the tunneled connection suddenly drops to ensure no unencrypted data is transmitted. Also autoconnect should be enabled to ensure connection to the vpn.

## 5.5 Validation and testing

Confirm the service enabled on the system:

```
$ systemctl list-unit-files --type=service --state=enabled
$ systemctl status <service>
```

# 6. Logging and Monitoring

Securing Ubuntu Linux systems through proper logging and monitoring is essential for quickly detecting, analyzing, and responding to potential security threats. Effective logging and monitoring minimize the risk of undetected intrusions and security incidents.

## 6.1 Fundamental Principles of Logging

**Identify critical logs**

- Authentication: `/var/log/auth.log` or `/var/log/secure` (depending on the distribution).
- System Events: `/var/log/syslog` for general system logs.
- Kernel Events: `/var/log/kern.log` for kernel-related incidents.

**Standardize log management**

Use `rsyslog` to collect, filter, and route logs. Implement `logrotate` to automatically rotate, compress, and delete older log files to avoid large logs. Configure these settings:
`/etc/logrotate.conf` or `/etc/logrotate.d/`

**Restrict access to log files**

Limit log file access to root only `chmod 600 /var/log/*` to reduce the risk of unauthorized users reading or tampering with logs.

## 6.2 Monitoring Principles

**Auditd for file monitoring**

Install and enable `auditd` to track changes to critical system files:

```
$ auditctl -w /etc/passwd -p wa -k passwd_changes
```

This rule monitors read and write actions on `/etc/passwd` and logs all changes.

**Remote logging**

Forward logs to a centralized logging server to streamline analysis and enable long-term storage. To configure this, edit the `/etc/rsyslog.conf` file and add the following directive:

```
*.* @@<central-log-server>:514
```

**Incident identification**

Use monitoring tools like `OSSEC`, for file integrity monitoring and log analysis and `Nagios`, for real-time monitoring of system performance and services to detect unusual activities. These tools send notifications for anomalies, such as sudden permission changes or unexpected service interruptions.

## 6.3 Security Guidelines

Enable sudo logging to track administrative commands executed with sudo privileges by adding the following line to `/etc/sudoers`

```
Defaults logfile="/var/log/sudo.log"
```

**Encrypt log transfers**

Secure logs during remote transmission using TLS encryption. The directives below can be configured within `/etc/rsyslog.conf`

```
$DefaultNetstreamDriver gtls
$ActionSendStreamDriverMode 1
$ActionSendStreamDriverAuthMode anon
```

**Implement alerts and notifications**

Configure monitoring tools to send email or SMS alerts for excessive login attempts, changes to critical files or services, and high CPU or memory usage, which may indicate an attack.

## Summary

Securing Ubuntu Linux 24.04 Desktop systems requires a systematic approach addressing every aspect of endpoint protection. This guide has highlighted key steps to enhance security, from partitioning the filesystem and configuring firewalls to implementing account management and application control policies. Network security and logging have been emphasized as vital components of a robust defense.

By applying least privilege principles, enforcing strong passwords, and using tools like AppArmor, organizations can reduce the attack surface significantly. Proactive measures such as disabling unused services and enabling audit logging help identify and mitigate potential threats.

Regular validation, monitoring, and updates are crucial for maintaining a secure system. Following these practices ensures Ubuntu Linux endpoints are protected while enabling efficient and secure operation in a dynamic digital work environment.

*Allen Camille Muco*
*Hugo Polstam*
*Isac Safarov*
*Nicholas John Stevens*
*Sebastian Ekedahl*

References:
Further reading on filesystems: refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.html
Searchable Ubuntu community: help.ubuntu.com/community
Downloadable CIS Benchmark documents (Ubuntu Linus 24.04 v.1.0.0): www.cisecurity.org/cis-benchmarks
[4] NISTs Guide to ApplicationWhitelisting: nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-167.pdf
[5] AppArmor: documentation.ubuntu.com/server/how-to/security/apparmor/?_ga=2.247339137.409454790.1736606199-1771076647.1736606199
[6] How to debloat a Ubuntu system: linuxtldr.com/debloat-ubuntu/
[7] Bloatware removal script: gist.github.com/NickSeagull/ed43a80db6a54d69ded3e18f8babaf19
How to disable and remove unnecessary services: www.tecmint.com/disable-unwanted-services-linux/
NIST National Vulnerability Database: nvd.nist.gov/vuln