CIS Benchmark Testing Checklist

Ubuntu Linux 24.04 LTS

Based on CIS Ubuntu Linux 24.04 Benchmark v1.0.0

Document Information

Document Version:	1.0.0
Release Date:	December 2, 2025
Classification:	Internal Use Only
Status:	Active

Csalab

Security Assessment Team

WARNING

This document contains security-sensitive information.
Do not distribute without proper authorization.
Always maintain backup before implementation.

Contents

Executive Summary

Document Purpose

This document serves as a comprehensive testing checklist for implementing and validating CIS (Center for Internet Security) Benchmark controls on Ubuntu Linux 24.04 LTS systems. It provides a structured approach to ensure system security hardening meets industry-standard security requirements.

Scope

- **Target Systems:** Ubuntu Linux 24.04 LTS Servers
- **Benchmark Version:** CIS Ubuntu Linux 24.04 Benchmark v1.0.0
- **Compliance Levels:** Level 1 and Level 2 Server profiles
- **Assessment Type:** Comprehensive security configuration review

Key Objectives

1. Systematically implement CIS Benchmark security controls
2. Validate proper configuration of security settings
3. Document compliance status and exceptions
4. Ensure minimal impact on system functionality and performance
5. Provide clear rollback procedures for all changes
6. Prevent system lockout through comprehensive testing

Document Usage

This checklist should be completed by qualified security professionals or system administrators with:

- Strong understanding of Linux system administration
- Knowledge of security hardening principles
- Authorization to make system-level changes
- Access to system backup and recovery procedures
- Console/Out-of-Band management access
- Understanding of organizational security policies

Testing Phases

Phase	Description	Duration
Phase 1	Pre-Testing Preparation	2-4 hours
Phase 2	Baseline Functional Testing	1-2 hours
Phase 3	CIS Benchmark Implementation	6-10 hours
Phase 4	Post-Implementation Validation	2-4 hours
Phase 5	Documentation & Sign-Off	1-2 hours
	Total Estimated Time:	**12-22 hours**

IMPORTANT

Before proceeding:

- Ensure full system backup is completed and verified
- Obtain proper change management approvals
- Schedule appropriate maintenance window (minimum 8 hours)
- Verify rollback procedures are documented and tested
- Notify all relevant stakeholders
- Ensure console/OOB access is available and tested
- Have emergency contacts ready

1 Glossary & Terminology

Term	Definition
AIDE	Advanced Intrusion Detection Environment - A file and directory integrity checker
AppArmor	Mandatory Access Control (MAC) system for Linux kernel security
ASLR	Address Space Layout Randomization - Memory protection mechanism
CIS	Center for Internet Security - Non-profit organization providing security benchmarks
Core Dump	Memory snapshot of a program at crash time (security risk if not restricted)
GRUB	Grand Unified Bootloader - Default bootloader for Linux systems
MAC	Mandatory Access Control - Security architecture for access control
NTP	Network Time Protocol - Protocol for clock synchronization
OOB	Out-of-Band - Management access independent of main network (e.g., console, iLO, iDRAC)
PAM	Pluggable Authentication Modules - Authentication framework for Linux
Prelink	Program to reduce startup time by pre-linking (conflicts with ASLR)
SELinux	Security-Enhanced Linux - Alternative MAC system (not used in Ubuntu)
Systemd	System and service manager for Linux
Sysctl	Interface for kernel parameter management at runtime
UFW	Uncomplicated Firewall - User-friendly interface for iptables
UUID	Universally Unique Identifier - Used for partition identification
faillock	PAM module for account lockout after failed login attempts
auditd	Linux audit daemon for security event logging
journald	systemd logging service
rsyslog	Reliable system logging daemon

2 Project Information

Assessment Information

Field	Value
Server Name	_____
IP Address	_____
Server Function	_____
CIS Profile	☐ Level 1 Server ☐ Level 2 Server
Benchmark Version	CIS Ubuntu Linux 24.04 v1.0.0
Document Version	1.0.0
Test Date	_____
Tester Name	_____
Environment	☐ Lab ☐ Dev ☐ Staging ☐ Production

2.1 Document Purpose

This comprehensive checklist is designed to guide security professionals through the complete testing and validation process of CIS Benchmark implementation on Ubuntu 24.04 LTS systems. It covers all phases from pre-testing preparation to post-implementation validation with complete coverage of all CIS controls.

2.2 Scope

This checklist covers:

- Pre-testing system information collection
- Baseline functional testing
- Complete CIS Benchmark implementation testing (Sections 1-6)
- All 100+ CIS controls with detailed procedures
- Post-implementation validation
- Documentation and sign-off procedures
- Rollback and recovery procedures

2.3 Contact Information & Escalation

Role	Name	Contact	Availability
Project Lead	_____	_____	_____
Security Engineer	_____	_____	_____
System Administrator	_____	_____	_____
Application Owner	_____	_____	_____
Change Manager	_____	_____	_____
Escalation Manager	_____	_____	_____

Table 2: Project Contact Information

Emergency Contacts

- **24/7 Operations Center:** _____
- **Security Incident Hotline:** _____
- **Infrastructure On-Call:** _____
- **Vendor Support (if applicable):** _____

2.4 Prerequisites

Before using this checklist:

☐ Obtain CIS Ubuntu Linux 24.04 Benchmark document

☐ Ensure administrative access to target system

☐ Prepare backup and rollback procedures

☐ Schedule appropriate maintenance window (minimum 8 hours)

☐ Notify all stakeholders

☐ Verify console/OOB access is working

☐ Review all warning boxes in this document

☐ Understand rollback procedures

☐ Have emergency contacts ready

3 Phase 1: Pre-Testing Preparation

CRITICAL: Mandatory Pre-Testing Requirements

- ☐ Full system backup completed and verified
- ☐ Console/out-of-band access available AND TESTED
- ☐ Rollback plan documented and approved
- ☐ Change Request approved by management
- ☐ All stakeholders notified
- ☐ Maintenance window scheduled (minimum 8 hours)
- ☐ Emergency contacts list verified
- ☐ Backup restoration tested (if possible)

3.1 System Information Collection

3.1.1 System Details

```
# Collect OS Version
lsb_release -a > /tmp/os_version.txt

# Collect Kernel Version
uname -a > /tmp/kernel_version.txt

# Collect System Architecture
dpkg --print-architecture > /tmp/architecture.txt

# Collect Hostname
hostname -f > /tmp/hostname.txt

# Display all
echo "=== System Information ==="
cat /tmp/os_version.txt
echo ""
cat /tmp/kernel_version.txt
echo ""
cat /tmp/architecture.txt
echo ""
cat /tmp/hostname.txt
```

Checklist:

☐ OS Version: Ubuntu _____ recorded

☐ Kernel Version: _____ recorded

☐ Architecture: _____ recorded

☐ Hostname: _____ recorded

☐ All information saved to documentation folder

3.1.2 Hardware Information

```
# CPU Information
lscpu > /tmp/cpu_info.txt

# Memory Information
free -h > /tmp/memory_info.txt

# Disk Information
df -h > /tmp/disk_info.txt
lsblk > /tmp/block_devices.txt

# Display summary
echo "=== Hardware Information ==="
echo "CPU:"
lscpu | grep "Model name"
lscpu | grep "^CPU(s):"
echo ""
echo "Memory:"
free -h | grep "^Mem:"
echo ""
echo "Disk:"
df -h | grep -E "^/dev"
```

Checklist:

☐ CPU Cores: _____ CPU Model: _____

☐ Total Memory: _____ GB

☐ Disk Space: / = _____ GB, /var = _____ GB

☐ Information saved to documentation folder

3.1.3 Network Information

```
# IP Configuration
ip addr show > /tmp/ip_config.txt

# Routing Table
ip route show > /tmp/routing_table.txt

# DNS Configuration
cat /etc/resolv.conf > /tmp/dns_config.txt

# Listening Ports
ss -tulnp > /tmp/listening_ports.txt

# Display summary
echo "=== Network Information ==="
echo "IP Address:"
ip addr show | grep "inet " | grep -v "127.0.0.1"
echo ""
echo "Default Gateway:"
ip route | grep default
echo ""
echo "DNS Servers:"
grep "^nameserver" /etc/resolv.conf
echo ""
echo "Listening Ports:"
ss -tulnp | grep LISTEN
```

Checklist:

☐ Primary IP: _____ Interface: _____

☐ Default Gateway: _____

☐ DNS Servers: _____

☐ Open Ports documented: ☐ Yes ☐ No

☐ Count of listening ports: _____

3.2 Application & Service Inventory

3.2.1 Running Services

```
# List Running Services
systemctl list-units --type=service --state=running > /tmp/running_services.txt

# List Enabled Services
systemctl list-unit-files --state=enabled > /tmp/enabled_services.txt

# List Failed Services
systemctl list-units --type=service --state=failed > /tmp/failed_services.txt

# Display summary
echo "=== Service Status ==="
echo "Running Services:"
systemctl list-units --type=service --state=running | grep "\.service" | wc -l
echo ""
echo "Enabled Services:"
systemctl list-unit-files --state=enabled | grep "\.service" | wc -l
echo ""
echo "Failed Services:"
systemctl list-units --type=service --state=failed
```

Checklist:

☐ Total running services: _____

☐ Total enabled services: _____

☐ Failed services (should be 0): _____

☐ If failed services exist, investigate before proceeding

3.2.2 Installed Packages

```
# List All Installed Packages
dpkg --get-selections > /tmp/packages_list.txt

# List Manually Installed Packages
apt-mark showmanual > /tmp/manual_packages.txt

# Count Total Packages
echo "Total installed packages: $(dpkg --get-selections | wc -l)"

# Check for security updates
apt update
apt list --upgradable > /tmp/upgradable_packages.txt
```

Checklist:

☐ Total installed packages: _____

☐ Package list saved: ☐ Yes ☐ No

☐ Security updates available: _____

☐ Critical packages identified:

- ☐ Web Server: _____ Version: _____
- ☐ Database: _____ Version: _____
- ☐ Application: _____ Version: _____
- ☐ Monitoring Agent: _____ Version: _____

3.3 Backup & Rollback Preparation

ALL items must be checked before proceeding!

3.3.1 Full System Backup

```
#!/bin/bash
# CIS Implementation Backup Script
# Save as: /root/cis_backup.sh

BACKUP_DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/backup/cis_pre_${BACKUP_DATE}"

echo "=== CIS Benchmark Pre-Implementation Backup ==="
echo "Backup started: $(date)"
echo "Backup directory: ${BACKUP_DIR}"

# Create backup directory
mkdir -p ${BACKUP_DIR}/{etc,boot,var,home,logs}

# Backup system configuration
echo "Backing up /etc..."
tar -czf ${BACKUP_DIR}/etc_backup.tar.gz /etc/ 2>/dev/null

echo "Backing up /boot..."
tar -czf ${BACKUP_DIR}/boot_backup.tar.gz /boot/ 2>/dev/null

echo "Backing up logs..."
tar -czf ${BACKUP_DIR}/var_backup.tar.gz /var/log/ /var/spool/ 2>/dev/null

# Backup package state
echo "Backing up package state..."
dpkg --get-selections > ${BACKUP_DIR}/dpkg_selections.txt
apt-mark showmanual > ${BACKUP_DIR}/apt_manual.txt

# Backup network configuration
echo "Backing up network configuration..."
ip addr show > ${BACKUP_DIR}/network_config.txt
ip route show > ${BACKUP_DIR}/routing.txt
ip rule show > ${BACKUP_DIR}/routing_rules.txt

# Backup firewall rules
```

```
echo "Backing up firewall rules..."
if command -v ufw &> /dev/null; then
    ufw status numbered > ${BACKUP_DIR}/ufw_rules.txt 2>/dev/null || echo "UFW not
    active" > ${BACKUP_DIR}/ufw_rules.txt
fi
iptables-save > ${BACKUP_DIR}/iptables_rules.txt
ip6tables-save > ${BACKUP_DIR}/ip6tables_rules.txt

# Backup important service configs
echo "Backing up service configurations..."
systemctl list-unit-files --state=enabled > ${BACKUP_DIR}/enabled_services.txt
systemctl list-units --type=service --state=running > ${BACKUP_DIR}/running_services.txt

# Backup user information
echo "Backing up user information..."
getent passwd > ${BACKUP_DIR}/passwd_backup.txt
getent group > ${BACKUP_DIR}/group_backup.txt
getent shadow > ${BACKUP_DIR}/shadow_backup.txt

# Create backup info file
cat > ${BACKUP_DIR}/BACKUP_INFO.txt <<EOF
=== CIS Benchmark Pre-Implementation Backup ===
Backup Created: $(date)
Hostname: $(hostname)
Kernel: $(uname -r)
OS: $(lsb_release -d | cut -f2)
Purpose: Pre-CIS Benchmark Implementation
Backup Location: ${BACKUP_DIR}
Tester: $(whoami)

=== Backup Contents ===
- System configuration (/etc)
- Boot configuration (/boot)
- Log files (/var/log, /var/spool)
- Package selections and manual packages
- Network configuration
- Firewall rules (UFW and iptables)
- Service status
- User and group information

=== Restoration Instructions ===
To restore this backup:
1. Boot from rescue media if needed
2. Mount the root filesystem
3. Extract archives: tar -xzf etc_backup.tar.gz -C /
4. Restore package selections: dpkg --set-selections < dpkg_selections.txt
5. Reboot and verify

=== File Sizes ===
EOF

# Add file sizes
du -sh ${BACKUP_DIR}/* >> ${BACKUP_DIR}/BACKUP_INFO.txt

echo ""
echo "Backup completed: ${BACKUP_DIR}"
echo "Backup size: $(du -sh ${BACKUP_DIR} | cut -f1)"
echo ""
echo "Please verify backup integrity!"

# Verify backup
```

```
echo "=== Verifying Backup ==="
if [ -f "${BACKUP_DIR}/etc_backup.tar.gz" ]; then
    tar -tzf ${BACKUP_DIR}/etc_backup.tar.gz > /dev/null 2>&1
    if [ $? -eq 0 ]; then
        echo "[PASS] /etc backup verified"
    else
        echo "[FAIL] /etc backup corrupted!"
        exit 1
    fi
fi

echo ""
echo "Backup verification complete"
echo "Backup info saved to: ${BACKUP_DIR}/BACKUP_INFO.txt"
```

Execute Backup Script:

```
# Make executable
chmod +x /root/cis_backup.sh

# Run backup
/root/cis_backup.sh

# Verify backup completed
ls -lah /backup/cis_pre_*
cat /backup/cis_pre_*/BACKUP_INFO.txt
```

Backup Verification Checklist:

☐ VM Snapshot created (if VM)

- Snapshot Name: _____
- Snapshot Time: _____
- Snapshot Verified: ☐ Yes ☐ No
- Snapshot Size: _____ GB

☐ System Configuration Backup

- ☐ /etc/ backed up (Size: _____)
- ☐ /boot/ backed up (Size: _____)
- ☐ /var/ backed up (Size: _____)
- ☐ Backup integrity verified: ☐ Yes ☐ No
- ☐ Backup location: _____

☐ Database Backup (if applicable)

- ☐ MySQL/MariaDB backed up
- ☐ PostgreSQL backed up
- ☐ MongoDB backed up
- ☐ N/A (no database)
- Backup location: _____

☐ Package State Backup

- ☐ dpkg selections saved
- ☐ apt-mark manual packages saved

☐ Network Configuration Backup

- ☐ IP configuration saved
- ☐ Routing table saved
- ☐ Firewall rules saved

☐ Backup stored in secure location: _____

☐ Backup location has sufficient space: ☐ Yes ☐ No

☐ Backup retention period defined: _____ days

☐ Backup accessible from console/OOB: ☐ Yes ☐ No

3.3.2 Rollback Plan Documentation

Rollback Checklist:

☐ Rollback procedure documented: ☐ Yes ☐ No

☐ Rollback steps tested (if possible): ☐ Yes ☐ No

☐ Rollback time estimate: _____ minutes

☐ Rollback owner assigned: _____

☐ Emergency contacts list updated: ☐ Yes ☐ No

☐ Out-of-band access available: ☐ Yes ☐ No

☐ Console access tested: ☐ Yes ☐ No

Rollback Procedures:

1. **VM Snapshot Rollback (Fastest Method):**

```
# For VMware
vim-cmd vmsvc/snapshot.revert <vm_id> <snapshot_id>

# For KVM/libvirt
virsh snapshot-revert <domain> <snapshot_name>

# For VirtualBox
VBoxManage snapshot <vm_name> restore <snapshot_name>

# After revert, verify services
systemctl status
ss -tulnp
```

2. **Configuration Rollback (If VM snapshot not available):**

```
# Restore configuration files
cd /backup/cis_pre_*/
tar -xzf etc_backup.tar.gz -C /

# Restore firewall rules
ufw disable
iptables-restore < iptables_rules.txt

# Restart affected services
systemctl daemon-reload
systemctl restart ssh
systemctl restart networking
systemctl restart systemd-resolved
```

```
# Verify
systemctl status ssh
ip addr show
ping -c 3 8.8.8.8
```

3. **Selective Service Rollback (If only specific service fails):**

```
# Example: Restore SSH config only
cp /backup/cis_pre_*/etc/ssh/sshd_config /etc/ssh/sshd_config
systemctl restart sshd

# Example: Restore firewall only
ufw disable
# Or restore specific rules

# Example: Restore PAM config
cp /backup/cis_pre_*/etc/pam.d/* /etc/pam.d/
```

4. **Package Rollback (If package issues):**

```
# Restore package selections
dpkg --set-selections < /backup/cis_pre_*/dpkg_selections.txt

# Install/remove packages to match backup state
apt-get dselect-upgrade
```

Execute rollback immediately if:

- Critical service is down for > 15 minutes
- System is inaccessible via normal means
- SSH access is completely lost (console required)
- Multiple services are failing simultaneously
- Data corruption or loss has occurred
- Security incident is active
- Quick fix attempts have failed (3 attempts)
- System is unstable or crashing

Document rollback decision:
Reason: _____
Authorized by: _____ Time: _____
Rollback method: ☐ VM Snapshot ☐ Config Restore ☐ Selective

3.4 Stakeholder Communication

Communication Checklist:

☐ Security Team Notified

- Contact: _____ Date: _____
- Method: ☐ Email ☐ Meeting ☐ Slack

☐ Operations Team Notified

- Contact: _____ Date: _____
- Method: ☐ Email ☐ Meeting ☐ Slack

☐ Application Owner Notified

- Contact: _____ Date: _____
- Method: ☐ Email ☐ Meeting ☐ Slack

☐ Management Approval Obtained

- Approver: _____ Date: _____
- Method: ☐ Email ☐ Meeting ☐ Change Request

☐ Change Management

- ☐ Change Request ID: _____
- ☐ Approval Status: ☐ Approved ☐ Pending
- ☐ Maintenance Window: From _____ To _____
- ☐ Impact Assessment: ☐ Low ☐ Medium ☐ High
- ☐ Rollback Plan: ☐ Documented ☐ Approved
- ☐ Communication Sent: ☐ Email ☐ Slack ☐ Meeting

☐ Support Readiness

- ☐ On-call team briefed
- ☐ Extended support coverage arranged
- ☐ War room/bridge setup (if needed)
- ☐ Escalation path defined
- ☐ Emergency contacts verified

Communication Template

Email Subject: [MAINTENANCE] CIS Benchmark Implementation - [Server Name]
Email Body:

```
Dear Team,

We will be implementing CIS Benchmark security hardening on:
- Server: [Server Name]
- Date: [Date]
- Time Window: [Start] - [End]
- Expected Impact: [Describe]
- Services Affected: [List]

During this maintenance:
- System will remain online
- Brief service interruptions possible
- SSH access may be temporarily affected

Emergency Contact: [Name] - [Phone]
Rollback Plan: [Available/Time]

Please acknowledge receipt and report any concerns.

Best regards,
[Your Name]
```

4 Phase 2: Baseline Functional Testing

Purpose: Establish System Baseline

This phase establishes the baseline functionality of the system BEFORE CIS implementation. All tests must pass to ensure a stable starting point.
Critical: Document any existing issues before proceeding. Do not implement CIS controls on a system with pre-existing problems.

4.1 Authentication & Access Testing

4.1.1 Local Console Access

```
# Test 1: Root Login (if allowed - will be disabled later)
# Login via console with root credentials
# Document if root login works

# Test 2: Regular User Login
# Login via console with regular user credentials

# Test 3: Check login logs
tail -20 /var/log/auth.log

# Test 4: Verify console accessibility
# This is CRITICAL - you must have console access
# Test console/OOB access now before proceeding
```

Checklist:

☐ Root Login Test (if currently allowed)

- Test Result: ☐ PASS ☐ FAIL ☐ Not Allowed
- Notes: _____

☐ Regular User Login Test

- Username: _____
- Test Result: ☐ PASS ☐ FAIL
- Can access home directory: ☐ Yes ☐ No
- Notes: _____

☐ Console/OOB Access Test

- Console type: ☐ Physical ☐ iLO/iDRAC ☐ KVM ☐ Serial
- Access working: ☐ Yes ☐ No
- **If No: DO NOT PROCEED WITH CIS IMPLEMENTATION**

☐ Service Account Login (if applicable)

- Test Result: ☐ PASS ☐ FAIL ☐ N/A

Console/OOB access is MANDATORY before proceeding!
Why: CIS implementation may lock you out of SSH. Console is your safety net.
Test console access now:

- Can you access console?
- Can you login via console?
- Can you see system output?
- Can you interact with system?

If console access is not working, STOP and fix it first!

4.1.2 SSH Access Testing

```
# Test 1: SSH from admin workstation
ssh user@server_ip

# Test 2: SSH with different authentication methods
ssh -o PreferredAuthentications=password user@server_ip
ssh -o PreferredAuthentications=publickey user@server_ip

# Test 3: SSH with different users
ssh admin_user@server_ip
ssh regular_user@server_ip

# Test 4: Check SSH version and configuration
ssh -V
sshd -T | head -20

# Test 5: Document current SSH settings
sshd -T > /tmp/sshd_config_baseline.txt
```

Checklist:

☐ SSH from Admin Workstation

- Test Result: ☐ PASS ☐ FAIL
- Authentication Method: ☐ Password ☐ Key ☐ Both
- Connection Time: _____ seconds
- SSH Version: _____

☐ SSH with Different Users

☐ Admin User: ☐ PASS ☐ FAIL
☐ Regular User: ☐ PASS ☐ FAIL
☐ Service Account: ☐ PASS ☐ FAIL ☐ N/A

☐ SSH Port Forwarding (if used)

- Command: `ssh -L 8080:localhost:80 user@server`
- Test Result: ☐ PASS ☐ FAIL ☐ N/A
- Required for operations: ☐ Yes ☐ No

☐ SSH Configuration Documented

- SSH Version: _____
- Port: _____
- Protocol: _____
- PermitRootLogin: _____
- PasswordAuthentication: _____
- Baseline config saved: ☐ Yes ☐ No

4.1.3 Sudo Access Testing

```
# Test 1: List sudo permissions
sudo -l

# Test 2: Test sudo execution
sudo id

# Test 3: Switch to root
sudo su -

# Test 4: Test sudo without password (if configured)
sudo -n whoami 2>&1

# Test 5: Check sudoers configuration
sudo visudo -c
cat /etc/sudoers
ls -la /etc/sudoers.d/

# Test 6: Document sudo configuration
sudo -l > /tmp/sudo_baseline.txt
```

Checklist:

☐ sudo -l (list permissions)

- Output Saved: ☐ Yes ☐ No
- Permissions appropriate: ☐ Yes ☐ No
- Can run all commands: ☐ Yes ☐ Limited

☐ sudo id (test execution)

- Test Result: ☐ PASS ☐ FAIL
- Returned UID: _____ (should be 0)
- Password required: ☐ Yes ☐ No

☐ sudo su - (switch to root)

- Test Result: ☐ PASS ☐ FAIL
- Can become root: ☐ Yes ☐ No

☐ sudo without password (if configured)

- Test Result: ☐ PASS ☐ FAIL ☐ N/A
- NOPASSWD configured: ☐ Yes ☐ No

☐ sudoers configuration valid

- Syntax Check: ☐ PASS ☐ FAIL

- No syntax errors: ☐ Confirmed

Ensure you have working sudo access before proceeding. CIS implementation requires sudo for most operations, and you'll need it for rollback if needed.
Test all sudo scenarios:

- Regular sudo command
- Becoming root (sudo su -)
- Running privileged services
- Editing configuration files

4.1.4 PAM Configuration Test

```
# Test 1: Check current PAM configuration
ls -la /etc/pam.d/
cat /etc/pam.d/common-auth
cat /etc/pam.d/common-password
cat /etc/pam.d/common-session
cat /etc/pam.d/common-account

# Test 2: Check password quality settings
if [ -f /etc/security/pwquality.conf ]; then
    cat /etc/security/pwquality.conf
else
    echo "pwquality not configured"
fi

# Test 3: Account lockout configuration
if [ -f /etc/security/faillock.conf ]; then
    cat /etc/security/faillock.conf
else
    echo "faillock not configured"
fi

# Test 4: Backup PAM configuration
mkdir -p /tmp/pam_backup
cp -r /etc/pam.d /tmp/pam_backup/
cp /etc/security/*.conf /tmp/pam_backup/ 2>/dev/null

# Test 5: Test password complexity (optional, use test account)
# passwd testuser
# Try setting weak password
```

Checklist:

☐ PAM configuration documented

- PAM files backed up: ☐ Yes ☐ No
- Configuration seems normal: ☐ Yes ☐ Issues Found

☐ Password quality module

- pwquality installed: ☐ Yes ☐ No
- Current requirements: _____

☐ Account lockout

- faillock configured: ☐ Yes ☐ No
- Lockout policy: _____

☐ PAM modules loaded

- pam_unix: ☐ Present
- pam_systemd: ☐ Present
- pam_limits: ☐ Present

4.2 Network Connectivity Testing

4.2.1 Outbound Connectivity

```
# Test 1: ICMP (Ping)
ping -c 4 8.8.8.8
ping -c 4 google.com

# Test 2: DNS Resolution
dig google.com
nslookup ubuntu.com
host www.example.com

# Test 3: HTTP/HTTPS
curl -I http://www.google.com
curl -I https://www.google.com
wget --spider http://archive.ubuntu.com

# Test 4: NTP (if used)
if systemctl is-active systemd-timesyncd >/dev/null 2>&1; then
    timedatectl status
    timedatectl show-timesync --all
fi

# Test 5: Trace route
traceroute -n google.com | head -10
mtr -r -c 10 8.8.8.8

# Test 6: Repository access
apt update
echo "Exit code: $?"
```

Checklist:

☐ ICMP (Ping)

- ☐ ping 8.8.8.8 - Result: ☐ PASS ☐ FAIL
- ☐ Packet Loss: _____ % (should be 0%)
- ☐ Average RTT: _____ ms

☐ DNS Resolution

- ☐ dig google.com - Result: ☐ PASS ☐ FAIL
- ☐ nslookup ubuntu.com - Result: ☐ PASS ☐ FAIL
- ☐ DNS Server responding: _____
- ☐ Resolution time: _____ ms

☐ HTTP/HTTPS

- ☐ curl http://google.com - Result: ☐ PASS ☐ FAIL
- ☐ HTTP Status Code: _____
- ☐ curl https://google.com - Result: ☐ PASS ☐ FAIL
- ☐ SSL verification: ☐ Working ☐ Issues

☐ NTP (if used)

- ☐ NTP query - Result: ☐ PASS ☐ FAIL ☐ N/A
- ☐ Time synchronized: ☐ Yes ☐ No
- ☐ Time source: _____

☐ Repository Access

- ☐ apt update - Result: ☐ PASS ☐ FAIL
- ☐ All repos accessible: ☐ Yes ☐ Some Failed
- ☐ If failed, document: _____

4.2.2 Inbound Connectivity

```
# Test 1: List all listening ports
ss -tulnp | grep LISTEN

# Test 2: SSH (Port 22 or custom)
nc -zv localhost 22
telnet localhost 22

# Test 3: HTTP (Port 80) - if applicable
curl http://localhost
nc -zv localhost 80

# Test 4: HTTPS (Port 443) - if applicable
curl https://localhost
openssl s_client -connect localhost:443 </dev/null

# Test 5: Custom Application Ports
# Document each listening port
ss -tulnp | grep LISTEN > /tmp/listening_ports_baseline.txt

# Test 6: Test from external host
# From another machine:
# nc -zv <server_ip> 22
# curl http://<server_ip>
```

Checklist:

☐ SSH (Port 22 or custom)

- Port: _____
- Test: nc -zv localhost 22
- Result: ☐ PASS ☐ FAIL
- External access: ☐ Working ☐ Not Tested

☐ HTTP (Port 80)

- Test: curl http://localhost

- Result: ☐ PASS ☐ FAIL ☐ N/A
- Status code: _____
- External access: ☐ Working ☐ Not Tested ☐ N/A

☐ HTTPS (Port 443)

- Test: curl https://localhost
- Result: ☐ PASS ☐ FAIL ☐ N/A
- SSL Certificate Valid: ☐ Yes ☐ No ☐ N/A
- Certificate expiry: _____
- External access: ☐ Working ☐ Not Tested ☐ N/A

☐ Custom Application Ports

- Port 1: _____ Service: _____
- Result: ☐ PASS ☐ FAIL ☐ N/A
- Port 2: _____ Service: _____
- Result: ☐ PASS ☐ FAIL ☐ N/A
- Port 3: _____ Service: _____
- Result: ☐ PASS ☐ FAIL ☐ N/A

☐ All listening ports documented

- Total ports: _____
- List saved: ☐ Yes ☐ No
- All ports identified: ☐ Yes ☐ Unknown Ports Found

4.2.3 Internal Network Testing

```
# Test 1: Connection to Database Server
if [ -n "$DB_SERVER" ]; then
    nc -zv $DB_SERVER 3306 || nc -zv $DB_SERVER 5432
fi

# Test 2: Connection to Application Server
if [ -n "$APP_SERVER" ]; then
    curl http://$APP_SERVER:$APP_PORT/health
fi

# Test 3: Connection to Storage/NFS
if systemctl is-active nfs-client >/dev/null 2>&1; then
    showmount -e $NFS_SERVER
fi

# Test 4: Connection to LDAP/AD (if applicable)
if [ -n "$LDAP_SERVER" ]; then
    ldapsearch -x -H ldap://$LDAP_SERVER -b "" -s base
fi

# Test 5: Test inter-service communication
# Document all required connections
netstat -ntp 2>/dev/null | grep ESTABLISHED
```

Checklist:

☐ Connection to Database Server

- Server: _____ Port: _____
- Result: ☐ PASS ☐ FAIL ☐ N/A
- Can query database: ☐ Yes ☐ No ☐ N/A

☐ Connection to Application Server

- Server: _____ Port: _____
- Health check: ☐ Healthy ☐ Unhealthy ☐ N/A
- Response time: _____ ms

☐ Connection to Storage/NFS

- Server: _____
- Mount test: ☐ PASS ☐ FAIL ☐ N/A
- Shares accessible: _____

☐ Connection to LDAP/AD

- Server: _____
- Connection: ☐ PASS ☐ FAIL ☐ N/A
- Authentication: ☐ Working ☐ Not Working ☐ N/A

☐ Network Performance

☐ Bandwidth Test (if applicable)

* Throughput: _____ Mbps
* Tool used: _____

☐ Latency Test

* Average Latency to gateway: _____ ms
* Average Latency to internet: _____ ms

☐ All required network paths tested

☐ No network issues identified

4.3 Service & Application Testing

4.3.1 System Services

```
# Test 1: SSH Service
systemctl status ssh
systemctl is-enabled ssh
systemctl is-active ssh

# Test 2: Cron Service
systemctl status cron
crontab -l
ls -la /etc/cron.*

# Test 3: Rsyslog Service
systemctl status rsyslog
logger "Baseline test message $(date)"
sleep 2
```

```
tail -5 /var/log/syslog | grep "Baseline test"

# Test 4: Time Synchronization
systemctl status systemd-timesyncd
timedatectl status

# Test 5: Journal Service
systemctl status systemd-journald
journalctl -xe --no-pager | tail -20

# Test 6: Document all running services
systemctl list-units --type=service --state=running > /tmp/services_baseline.txt
```

Checklist:

☐ SSH Service

- Status: ☐ Active ☐ Inactive ☐ Failed
- Enabled: ☐ Yes ☐ No
- Process ID: _____

☐ Cron Service

- Status: ☐ Active ☐ Inactive ☐ Failed
- Cron jobs count: User: _____ System: _____
- Cron jobs documented: ☐ Yes ☐ No

☐ Rsyslog Service

- Status: ☐ Active ☐ Inactive ☐ Failed
- Test message logged: ☐ Yes ☐ No
- Log rotation working: ☐ Yes ☐ No ☐ Not Tested

☐ Time Synchronization

- Status: ☐ Active ☐ Inactive ☐ Failed
- Time synchronized: ☐ Yes ☐ No
- NTP Server: _____
- Time accurate: ☐ Yes ☐ No

☐ Journal Service

- Status: ☐ Active ☐ Inactive ☐ Failed
- Disk usage: _____ MB
- Persistence: ☐ Enabled ☐ Disabled

☐ All critical services running

☐ No failed services found

4.3.2 Web Server (if applicable)

```
# Test 1: Apache2/Nginx Status
if systemctl list-units --type=service | grep -q apache2; then
    systemctl status apache2
    apache2 -v
elif systemctl list-units --type=service | grep -q nginx; then
    systemctl status nginx
    nginx -v
fi

# Test 2: Web Server Response
curl -I http://localhost
curl -I https://localhost 2>/dev/null

# Test 3: SSL Certificate
if systemctl is-active nginx >/dev/null 2>&1 || systemctl is-active apache2 >/dev/null
    2>&1; then
    echo | openssl s_client -connect localhost:443 2>/dev/null | openssl x509 -noout -
    dates
fi

# Test 4: Virtual Hosts
if command -v apache2ctl >/dev/null 2>&1; then
    apache2ctl -S
elif command -v nginx >/dev/null 2>&1; then
    nginx -T 2>&1 | grep "server_name"
fi

# Test 5: Access and Error Logs
if [ -d /var/log/apache2 ]; then
    tail -20 /var/log/apache2/access.log
    tail -20 /var/log/apache2/error.log
elif [ -d /var/log/nginx ]; then
    tail -20 /var/log/nginx/access.log
    tail -20 /var/log/nginx/error.log
fi
```

Checklist:

☐ Web Server Status

- Server Type: ☐ Apache2 ☐ Nginx ☐ Other: _____
- Status: ☐ Active ☐ Inactive ☐ Failed ☐ N/A
- Version: _____

☐ Web Server Response

- HTTP Status Code: _____
- Response Time: _____ ms
- Content serving: ☐ Working ☐ Issues

☐ SSL Certificate

- Certificate Valid: ☐ Yes ☐ No ☐ N/A
- Expiry Date: _____
- Issuer: _____

☐ Virtual Hosts Configured

- Number of vhosts: _____
- Configuration documented: ☐ Yes ☐ No

☐ No errors in logs

☐ All sites accessible

4.3.3 Database (if applicable)

```
# Test 1: Database Service Status
for db in mysql mariadb postgresql mongodb; do
    if systemctl list-units --type=service | grep -q $db; then
        echo "=== $db ==="
        systemctl status $db
    fi
done

# Test 2: Database Connection Test
# MySQL/MariaDB
if command -v mysql >/dev/null 2>&1; then
    mysql -u root -p -e "SELECT 1;" 2>/dev/null && echo "MySQL: OK"
fi

# PostgreSQL
if command -v psql >/dev/null 2>&1; then
    sudo -u postgres psql -c "SELECT 1;" 2>/dev/null && echo "PostgreSQL: OK"
fi

# MongoDB
if command -v mongo >/dev/null 2>&1; then
    mongo --eval "db.runCommand({ping: 1})" 2>/dev/null && echo "MongoDB: OK"
fi

# Test 3: Database Version
mysql --version 2>/dev/null
psql --version 2>/dev/null
mongod --version 2>/dev/null | head -1

# Test 4: Database Performance
if command -v mysql >/dev/null 2>&1; then
    mysql -u root -p -e "SHOW PROCESSLIST;" 2>/dev/null
fi

# Test 5: Database Logs
tail -20 /var/log/mysql/error.log 2>/dev/null
tail -20 /var/log/postgresql/postgresql-*.log 2>/dev/null | head -20
```

Checklist:

☐ Database Service Status

- Type: ☐ MySQL ☐ PostgreSQL ☐ MongoDB ☐ Other
- Status: ☐ Active ☐ Inactive ☐ Failed ☐ N/A
- Version: _____

☐ Database Connection Test

- Connection: ☐ Success ☐ Failed ☐ N/A
- Authentication: ☐ Password ☐ Socket ☐ Other

☐ Database Performance

- Query Response Time: _____ ms
- Active Connections: _____

☐ Database Logs

- Recent Errors: ☐ None ☐ Some (document below)
- Error Details: _____

4.4 Logging & Monitoring Baseline

4.4.1 System Logging

```
# Test 1: Syslog Configuration
cat /etc/rsyslog.conf
ls -la /etc/rsyslog.d/

# Test 2: Log Test
logger "PRE-CIS Baseline Test $(date)"
sleep 2
grep "PRE-CIS Baseline Test" /var/log/syslog

# Test 3: Log Rotation
cat /etc/logrotate.conf
ls -la /etc/logrotate.d/

# Test 4: Log Files Accessibility
ls -la /var/log/
tail -10 /var/log/syslog
tail -10 /var/log/auth.log

# Test 5: Log space usage
du -sh /var/log/
df -h /var/log/
```

Checklist:

☐ Syslog Configuration

- Configuration Valid: ☐ Yes ☐ No
- Custom rules: _____

☐ Log Test

- Test message: logger "PRE-CIS Baseline Test"
- Message found in syslog: ☐ Yes ☐ No
- Result: ☐ PASS ☐ FAIL

☐ Log Rotation

- Configuration present: ☐ Yes ☐ No
- Rotation frequency: ☐ Daily ☐ Weekly ☐ Monthly
- Retention: _____ rotations

☐ Important logs present

☐ /var/log/syslog

- ☐ /var/log/auth.log
- ☐ /var/log/kern.log
- ☐ Application logs

☐ Log disk space sufficient

- Current usage: _____ MB
- Available space: _____ GB

5 Phase 3: CIS Benchmark Implementation Testing

Before proceeding with CIS implementation:

☐ All Phase 1 and Phase 2 tests must PASS

☐ Backup must be completed and verified

☐ Rollback plan must be ready

☐ Console/OOB access must be available

☐ Never close current SSH session until new connection is verified

☐ Test each section incrementally

☐ Document all changes as you make them

☐ Take snapshots after each major section

5.1 Section 1: Initial Setup

5.1.1 1.1 Filesystem Configuration

1.1.1 Disable unused filesystems

1.1.1.1 Disable cramfs

```
# Disable cramfs
echo "install cramfs /bin/true" > /etc/modprobe.d/cramfs.conf
rmmod cramfs 2>/dev/null

# Verify
lsmod | grep cramfs
modprobe -n -v cramfs
```

Checklist:

☐ cramfs disabled in modprobe

☐ Module unloaded

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.1.2 Disable freevxfs

```
# Disable freevxfs
echo "install freevxfs /bin/true" > /etc/modprobe.d/freevxfs.conf
rmmod freevxfs 2>/dev/null

# Verify
lsmod | grep freevxfs
```

Checklist:

☐ freevxfs disabled

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.1.3 Disable jffs2

```
# Disable jffs2
echo "install jffs2 /bin/true" > /etc/modprobe.d/jffs2.conf
rmmod jffs2 2>/dev/null

# Verify
lsmod | grep jffs2
```

Checklist:

☐ jffs2 disabled

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.1.4 Disable hfs

```
# Disable hfs
echo "install hfs /bin/true" > /etc/modprobe.d/hfs.conf
rmmod hfs 2>/dev/null

# Verify
lsmod | grep hfs
```

Checklist:

☐ hfs disabled

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.1.5 Disable hfsplus

```
# Disable hfsplus
echo "install hfsplus /bin/true" > /etc/modprobe.d/hfsplus.conf
rmmod hfsplus 2>/dev/null

# Verify
lsmod | grep hfsplus
```

Checklist:

☐ hfsplus disabled

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.1.6 Disable squashfs

```
# Check if snap packages are using squashfs
snap list 2>/dev/null | head -5

# If no snaps are installed, disable squashfs
if ! command -v snap >/dev/null 2>&1 || [ -z "$(snap list 2>/dev/null | tail -n +2)" ];
    then
    echo "install squashfs /bin/true" > /etc/modprobe.d/squashfs.conf
    rmmod squashfs 2>/dev/null
    echo "squashfs disabled"
else
    echo "WARNING: Snap packages in use, NOT disabling squashfs"
    echo "EXCEPTION: squashfs required for snap packages" > /tmp/cis_exceptions.txt
fi
```

```
# Verify squashfs module status
lsmod | grep squashfs
modprobe -n -v squashfs
```

Verification:

☐ squashfs decision documented

☐ If disabled: module not loaded in lsmod

☐ If disabled: install directive set to /bin/true

☐ If kept: snap packages dependency documented

1.1.1.7 Disable usb-storage

```
# Disable usb-storage kernel module
echo "install usb-storage /bin/true" > /etc/modprobe.d/usb-storage.conf

# Remove the module if currently loaded
rmmod usb-storage 2>/dev/null

# Verify usb-storage module is disabled
lsmod | grep usb-storage
modprobe -n -v usb-storage
```

Verification:

☐ usb-storage module disabled in modprobe

☐ usb-storage module not currently loaded

☐ modprobe test shows install /bin/true

☐ USB storage devices will not work (document if needed)

CAUTION: Disabling squashfs will break snap packages!
Ubuntu 24.04 uses snap for some system applications. Only disable squashfs if:

- You don't use snap packages
- You have removed all snap packages
- You have verified no applications depend on snap

To check snap usage:

```
snap list
```

```
# Check if snap is in use
snap list

# If no snaps are installed, disable squashfs
if [ $(snap list 2>/dev/null | wc -l) -eq 0 ]; then
    echo "install squashfs /bin/true" > /etc/modprobe.d/squashfs.conf
```

```
    rmmod squashfs 2>/dev/null
    echo "squashfs disabled"
else
    echo "WARNING: Snap packages in use, NOT disabling squashfs"
    echo "EXCEPTION: squashfs required for snap packages" > /tmp/cis_exceptions.txt
fi

# Verify
lsmod | grep squashfs
```

Checklist:

☐ Snap packages checked

- Snap packages installed: _____
- Snap required: ☐ Yes ☐ No

☐ squashfs decision

☐ Disabled (no snap in use)

☐ Kept enabled (snap required) - EXCEPTION DOCUMENTED

☐ Result: ☐ PASS ☐ FAIL ☐ N/A (Exception)

1.1.1.9 Disable udf

```
# Disable udf
echo "install udf /bin/true" > /etc/modprobe.d/udf.conf
rmmod udf 2>/dev/null

# Verify
lsmod | grep udf
```

Checklist:

☐ udf disabled

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

Summary: Verify all unused filesystems are disabled

```
# Verify all disabled
lsmod | grep -E "(cramfs|freevxfs|jffs2|hfs|hfsplus|udf|squashfs|usb-storage)"

# Should return nothing (or only squashfs if snap is used)
echo "If output is empty (or only squashfs), all filesystems are disabled: PASS"
```

Overall Filesystem Disable Checklist:

☐ cramfs disabled: ☐ PASS ☐ FAIL

☐ freevxfs disabled: ☐ PASS ☐ FAIL

☐ jffs2 disabled: ☐ PASS ☐ FAIL

☐ hfs disabled: ☐ PASS ☐ FAIL

☐ hfsplus disabled: ☐ PASS ☐ FAIL

☐ squashfs disabled: ☐ PASS ☐ EXCEPTION ☐ FAIL

☐ usb-storage disabled: ☐ PASS ☐ FAIL ☐ EXCEPTION

☐ udf disabled: ☐ PASS ☐ FAIL

☐ Overall Result: ☐ PASS ☐ FAIL

1.1.2 Ensure /tmp is a separate partition

```
# Check if /tmp is already a separate partition
findmnt --target /tmp

# If not separate, create using tmpfs
if ! findmnt --target /tmp | grep -q "/tmp"; then
    echo "tmpfs /tmp tmpfs defaults,rw,nosuid,nodev,noexec,relatime 0 0" >> /etc/fstab

    # Mount /tmp
    mount -o remount /tmp
fi

# Verify mount options
mount | grep /tmp
```

Checklist:

☐ /tmp is separate partition

- findmnt –target /tmp
- Output: _____
- Result: ☐ PASS ☐ FAIL

☐ Mount options correct

- mount | grep /tmp
- Has nosuid: ☐ Yes ☐ No
- Has nodev: ☐ Yes ☐ No
- Has noexec: ☐ Yes ☐ No

☐ /tmp accessible and writable

- `touch /tmp/test && rm /tmp/test`
- Result: ☐ PASS ☐ FAIL

1.1.2.1 Ensure nodev option set on /tmp partition

```
# Verify nodev option
mount | grep /tmp | grep nodev

# If not set, add to /etc/fstab and remount
sed -i 's|\(tmpfs\s\+/tmp\s\+tmpfs\s\+\)\(.*\)|\1\2,nodev|' /etc/fstab
mount -o remount /tmp

# Verify
mount | grep /tmp
```

Checklist:

☐ nodev option set

☐ Result: ☐ PASS ☐ FAIL

1.1.2.2 Ensure noexec option set on /tmp partition

```
# Verify noexec option
mount | grep /tmp | grep noexec

# Test noexec - should fail
cat > /tmp/test.sh << 'EOF'
#!/bin/bash
echo "This should not execute"
EOF
chmod +x /tmp/test.sh
/tmp/test.sh 2>&1  # Should fail with permission denied

# Clean up
rm /tmp/test.sh
```

Checklist:

☐ noexec option set

☐ Script execution blocked

- Test result: ☐ Blocked (PASS) ☐ Executed (FAIL)

☐ Result: ☐ PASS ☐ FAIL

1.1.2.3 Ensure nosuid option set on /tmp partition

```
# Verify nosuid option
mount | grep /tmp | grep nosuid

# Verify in fstab
grep "/tmp" /etc/fstab | grep nosuid
```

Checklist:

☐ nosuid option set

☐ Option persisted in /etc/fstab

☐ Result: ☐ PASS ☐ FAIL

1.1.3 Ensure /var is a separate partition

```
# Verify /var is separate partition
findmnt --target /var

# Check mount options
mount | grep /var

# Check disk space
df -h /var
```

Checklist:

☐ /var is separate partition

- findmnt –target /var
- Result: ☐ PASS ☐ FAIL ☐ N/A

☐ Disk space sufficient

- df -h /var
- Available: _____ GB
- Sufficient: ☐ Yes ☐ No

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.4 Ensure /var/tmp is a separate partition

```
# Option 1: Check if /var/tmp is separate partition
findmnt --target /var/tmp

# Option 2: Bind mount to /tmp (recommended if not separate)
if ! findmnt --target /var/tmp | grep -q "/var/tmp"; then
    echo "/tmp /var/tmp none bind 0 0" >> /etc/fstab
    mount --bind /tmp /var/tmp
fi

# Verify
mount | grep /var/tmp
```

Checklist:

☐ /var/tmp configuration

- Method: ☐ Separate partition ☐ Bind mount ☐ N/A
- Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.4.1 Ensure nodev option set on /var/tmp partition

```
# Verify nodev option on /var/tmp
mount | grep /var/tmp | grep nodev
```

Checklist:

☐ nodev option set

☐ Result: ☐ PASS ☐ FAIL

1.1.4.2 Ensure noexec option set on /var/tmp partition

```
# Verify noexec option on /var/tmp
mount | grep /var/tmp | grep noexec
```

Checklist:

☐ noexec option set

☐ Result: ☐ PASS ☐ FAIL

1.1.4.3 Ensure nosuid option set on /var/tmp partition

```
# Verify nosuid option on /var/tmp
mount | grep /var/tmp | grep nosuid
```

Checklist:

☐ nosuid option set

☐ Result: ☐ PASS ☐ FAIL

1.1.5 Ensure /var/log is a separate partition

```
# Verify /var/log is separate partition
findmnt --target /var/log

# Check disk space
df -h /var/log

# Test logging still works
logger "Test after /var/log configuration"
sleep 2
tail /var/log/syslog | grep "Test after /var/log"
```

Checklist:

☐ /var/log is separate partition

- findmnt –target /var/log
- Result: ☐ PASS ☐ FAIL ☐ N/A

☐ Disk space for logs

- df -h /var/log
- Available: _____ GB
- Growth rate: _____ MB/day
- Sufficient: ☐ Yes ☐ No

☐ Logging still functional

- Test message logged: ☐ Yes ☐ No
- Result: ☐ PASS ☐ FAIL

1.1.6 Ensure /home is a separate partition

```
# Verify /home is separate partition
findmnt --target /home

# Check mount options
mount | grep /home

# Test user access
su - $(awk -F: '$3>=1000 {print $1; exit}' /etc/passwd) -c "
    cd ~
    touch testfile
    ls -la testfile
    rm testfile
"
```

Checklist:

☐ /home is separate partition

- findmnt –target /home
- Result: ☐ PASS ☐ FAIL ☐ N/A

☐ User home directories accessible

- Test user can access home: ☐ Yes ☐ No

- Test user can create files: ☐ Yes ☐ No
- Result: ☐ PASS ☐ FAIL

1.1.6.1 Ensure nodev option set on /home partition

```
# Verify nodev option on /home
mount | grep /home | grep nodev

# If not set, add to /etc/fstab
# Example: UUID=xxx /home ext4 defaults,nodev 0 2

# Remount
mount -o remount /home

# Verify
mount | grep /home
```

Checklist:

☐ nodev option set on /home

☐ Users can still access home directories

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

1.1.7 Ensure /dev/shm is a separate partition

NEW: /dev/shm Configuration

This control was missing in the original script. /dev/shm is used for shared memory and should have restrictive mount options to prevent exploitation.

```
# Check current /dev/shm configuration
findmnt --target /dev/shm
mount | grep /dev/shm

# Ensure /dev/shm is configured with proper options in /etc/fstab
# Add or modify line:
grep -q "^tmpfs\s\+/dev/shm" /etc/fstab || \
echo "tmpfs /dev/shm tmpfs defaults,noexec,nodev,nosuid 0 0" >> /etc/fstab

# If line exists, ensure it has correct options
sed -i 's|^tmpfs\s\+/dev/shm\s\+tmpfs.*|tmpfs /dev/shm tmpfs defaults,noexec,nodev,
    nosuid 0 0|' /etc/fstab

# Remount with new options
mount -o remount /dev/shm

# Verify
mount | grep /dev/shm
echo "Expected: tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,...)"
```

Checklist:

☐ /dev/shm is mounted

- findmnt –target /dev/shm
- Result: ☐ Yes ☐ No

☐ Mount options configured

- ☐ noexec: mount | grep /dev/shm | grep noexec
- ☐ nodev: mount | grep /dev/shm | grep nodev
- ☐ nosuid: mount | grep /dev/shm | grep nosuid

☐ Test /dev/shm functionality

- Create test file: touch /dev/shm/test
- Test result: ☐ PASS ☐ FAIL
- Remove test: rm /dev/shm/test

☐ Test noexec on /dev/shm

- Create script: `echo '#!/bin/bash' > /dev/shm/test.sh`
- Make executable: `chmod +x /dev/shm/test.sh`
- Try execute: `/dev/shm/test.sh`
- Should fail: ☐ Yes (PASS) ☐ No (FAIL)
- Clean up: `rm /dev/shm/test.sh`

☐ Configuration persisted in /etc/fstab

- grep "/dev/shm" /etc/fstab
- Entry present: ☐ Yes ☐ No

☐ Application compatibility

- Applications using /dev/shm: _____
- Impact: ☐ None ☐ Minor ☐ Major
- Issues found: _____

☐ Overall Result: ☐ PASS ☐ FAIL

Application Compatibility Note

Some applications (like PostgreSQL, Oracle, some Java applications) use /dev/shm for shared memory. The noexec option generally doesn't affect these, but monitor application logs after implementation.
Common applications using /dev/shm:

- PostgreSQL
- Oracle Database
- Java applications (some)
- Docker containers
- Systemd services

If issues occur, consider documenting as exception with compensating controls.

Filesystem Configuration: Reboot Test

After making filesystem changes, a reboot test is MANDATORY to ensure the system boots correctly with new fstab entries.
Before rebooting:

- Verify /etc/fstab syntax: `mount -a`
- Check for errors
- Ensure console/OOB access is available
- Inform stakeholders

```
# Test fstab before reboot
mount -a
echo "Exit code: $?"
# If exit code is 0, fstab is valid

# Check for errors
dmesg | tail -20

# Backup current fstab
cp /etc/fstab /etc/fstab.backup.$(date +%Y%m%d)

# Review fstab
cat /etc/fstab

# If all looks good, reboot
# init 6
# Or: reboot
```

Post-Reboot Checklist:

☐ System boots successfully

- Boot time: _____ seconds
- Result: ☐ PASS ☐ FAIL

☐ All partitions mounted correctly

- findmnt
- All expected mounts present: ☐ Yes ☐ No
- Mount options correct: ☐ Yes ☐ No

☐ Verify specific mounts

☐ /tmp: mount | grep /tmp
☐ /var: mount | grep /var
☐ /var/tmp: mount | grep /var/tmp
☐ /var/log: mount | grep /var/log
☐ /home: mount | grep /home
☐ /dev/shm: mount | grep /dev/shm

☐ fstab entries correct

- cat /etc/fstab
- Syntax correct: ☐ Yes ☐ No
- No duplicate entries: ☐ Confirmed

☐ No boot errors

- journalctl -b -p err
- Errors Found: ☐ None ☐ Some (list below)
- Error Details: _____

☐ Services started correctly

- systemctl –failed
- Failed services: ☐ None ☐ Some (list below)
- Details: _____

☐ Applications functioning

- Web server: ☐ Running ☐ Issues
- Database: ☐ Running ☐ Issues
- Custom apps: ☐ Running ☐ Issues

☐ Users can login

- Console login: ☐ Working ☐ Issues
- SSH login: ☐ Working ☐ Issues
- sudo working: ☐ Yes ☐ No

☐ Overall Reboot Test: ☐ PASS ☐ FAIL

If system doesn't boot or has issues after reboot:

1. Boot to recovery mode
2. Mount root filesystem
3. Restore backup fstab: `cp /etc/fstab.backup.YYYYMMDD /etc/fstab`
4. Or comment problematic entries
5. Reboot again
6. Investigate and fix issues
7. Re-test carefully

Common issues:

- Wrong UUID in fstab
- Typo in mount options
- Partition doesn't exist
- Wrong filesystem type

Always have console/OOB access for recovery!

5.1.2 1.2 Package Management

1.2.1 Ensure GPG keys are configured

```
# List all configured GPG keys
apt-key list

# Check keyring files
ls -la /etc/apt/trusted.gpg.d/
ls -la /usr/share/keyrings/

# Verify repository signatures
apt-get update 2>&1 | grep -E "(NO_PUBKEY|BADSIG)"

# List repository keys with fingerprints
apt-key adv --list-public-keys --with-fingerprint --with-colons
```

Verification:

☐ GPG keys properly configured for official repositories

☐ No missing repository keys

☐ All keys have valid fingerprints

☐ Third-party repositories have proper GPG verification

1.2.2 Ensure package manager repositories are configured

```
# Check repositories
cat /etc/apt/sources.list
ls -la /etc/apt/sources.list.d/

# Verify Ubuntu official repositories
grep -E "^(deb|deb-src)" /etc/apt/sources.list /etc/apt/sources.list.d/*.list

# Check for unofficial/untrusted repositories
find /etc/apt -name "*.list" -exec grep -l "http.*" {} \;

# Test repository connectivity and signature verification
apt update
apt_exit=$?

# Check for errors
if [ $apt_exit -eq 0 ]; then
    echo "Repository configuration: OK"
else
    echo "Repository configuration: FAILED"
fi

# List available security updates
apt list --upgradable | grep -i security
```

Checklist:

☐ /etc/apt/sources.list reviewed

- Valid repositories: ☐ Yes ☐ No
- Security updates enabled: ☐ Yes ☐ No

- Main repository: _____

☐ /etc/apt/sources.list.d/ reviewed

- Number of additional repos: _____
- All repos legitimate: ☐ Yes ☐ No

☐ apt update test

- Result: ☐ PASS ☐ FAIL
- Errors: _____

☐ Security updates available

- Count: _____
- Critical: _____

5.1.3 1.3 Configure Bootloader

1.3.1 Ensure bootloader password is set

```
# Check current GRUB configuration
grep -E "^password|^set superusers" /boot/grub/grub.cfg 2>/dev/null || echo "No GRUB
    password found"

# Check if password is configured in grub.d
find /etc/grub.d/ -name "*" -exec grep -l "password\|superusers" {} \;

# Generate GRUB password hash
grub-mkpasswd-pbkdf2

# Create password configuration
cat > /etc/grub.d/40_custom << 'EOF'
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.

# Set superuser and password
set superusers="root"
password_pbkdf2 root <GRUB_PASSWORD_HASH_HERE>
EOF

# Make executable
chmod +x /etc/grub.d/40_custom

# Update GRUB configuration
update-grub
```

Verification:

☐ GRUB superuser configured

☐ Password hash generated and configured

☐ GRUB configuration updated successfully

☐ Bootloader requires password for admin functions

1.3.2 Ensure bootloader authentication is enabled

```
# Check GRUB password enforcement
grep -A 5 -B 5 "password\|superusers" /boot/grub/grub.cfg

# Verify unrestricted access is disabled
grep -E "^menuentry" /boot/grub/grub.cfg | head -5

# Check for potential security issues
find /boot/grub/ -name "*.cfg" -exec grep -L "password\|superusers" {} \;
```

Verification:

- ☐ Password protection enabled for admin menu items
- ☐ No unrestricted access to boot parameters
- ☐ Single-user mode protected
- ☐ Rescue mode requires authentication

1.3.3 Ensure permissions on bootloader config are configured

```
# Check permissions on GRUB files
stat -c "%a %U:%G %n" /boot/grub/grub.cfg
stat -c "%a %U:%G %n" /etc/default/grub
stat -c "%a %U:%G %n" /etc/grub.d/*

# Set proper permissions if needed
chown root:root /boot/grub/grub.cfg
chmod og-rwx /boot/grub/grub.cfg

chown root:root /etc/default/grub
chmod og-rwx /etc/default/grub

chown -R root:root /etc/grub.d
chmod -R og-rwx /etc/grub.d
```

Verification:

- ☐ grub.cfg permissions: 600 or more restrictive
- ☐ grub files owned by root:root
- ☐ No write access for non-privileged users
- ☐ Configuration files properly protected

Bootloader Security Summary:

- ☐ Bootloader password set and enforced
- ☐ Single-user/recovery modes protected
- ☐ Boot parameter modification restricted
- ☐ Configuration files properly secured

1.2.3 Ensure package updates are installed

```
# Update package database
apt update

# Check for available updates
apt list --upgradable

# Check security updates
apt list --upgradable | grep -i security

# Document current versions (before update)
dpkg -l > /tmp/package-list-before-cis.txt

# OPTIONAL: Install updates (consider maintenance window)
# apt upgrade -y
# apt full-upgrade -y

# If updates installed, reboot may be required
[ -f /var/run/reboot-required ] && cat /var/run/reboot-required
```

Checklist:

☐ Package database updated

- apt update result: ☐ Success ☐ Failed

☐ Available updates documented

- Total updates: _____ packages
- Security updates: _____ packages
- Kernel updates: _____

☐ Update decision

- ☐ Updates installed during CIS implementation
- ☐ Updates deferred to separate change
- ☐ No updates available

☐ If updates installed:

- Reboot required: ☐ Yes ☐ No
- Reboot scheduled: _____

☐ Package list saved

☐ Result: ☐ PASS ☐ FAIL

1.2.4 Ensure automatic updates are configured

```
# Install unattended-upgrades
apt install -y unattended-upgrades apt-listchanges

# Configure (interactive)
dpkg-reconfigure -plow unattended-upgrades

# Or configure non-interactively
cat > /etc/apt/apt.conf.d/50unattended-upgrades << 'EOF'
// Automatically upgrade packages from these repositories
```

```
Unattended-Upgrade::Allowed-Origins {
    "${distro_id}:${distro_codename}";
    "${distro_id}:${distro_codename}-security";
    "${distro_id}ESMApps:${distro_codename}-apps-security";
    "${distro_id}ESM:${distro_codename}-infra-security";
};

// Package blacklist (if needed)
Unattended-Upgrade::Package-Blacklist {
    // "kernel-image.*";
};

// Auto-reboot configuration
Unattended-Upgrade::Automatic-Reboot "false";
Unattended-Upgrade::Automatic-Reboot-WithUsers "false";
Unattended-Upgrade::Automatic-Reboot-Time "03:00";

// Email notifications
Unattended-Upgrade::Mail "root";
Unattended-Upgrade::MailReport "on-change";

// Remove unused dependencies
Unattended-Upgrade::Remove-Unused-Dependencies "true";
Unattended-Upgrade::Remove-Unused-Kernel-Packages "true";
EOF

# Enable automatic updates
cat > /etc/apt/apt.conf.d/20auto-upgrades << 'EOF'
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "7";
APT::Periodic::Unattended-Upgrade "1";
EOF

# Check service
systemctl status unattended-upgrades

# Enable service
systemctl enable unattended-upgrades

# Test (dry run)
unattended-upgrade --dry-run --debug
```

Checklist:

☐ unattended-upgrades installed

- Package installed: ☐ Yes ☐ No
- Version: _____

☐ Service enabled and running

- systemctl status unattended-upgrades
- Status: ☐ Active ☐ Inactive
- Enabled: ☐ Yes ☐ No

☐ Configuration reviewed

- Security updates enabled: ☐ Yes ☐ No
- Auto-reboot configured: ☐ Yes ☐ No ☐ N/A

- Email notifications: ☐ Configured ☐ Not Configured
- Email recipient: _____

☐ Update frequency

- Check daily: ☐ Yes ☐ No
- Download daily: ☐ Yes ☐ No
- Install daily: ☐ Yes ☐ No

☐ Package blacklist

- Blacklisted packages: _____
- Reason: _____

☐ Test upgrade (dry run)

- Dry-run successful: ☐ Yes ☐ No
- Packages would be upgraded: _____

☐ Result: ☐ PASS ☐ FAIL

Auto-Reboot Consideration

Important considerations for automatic reboots:
Disable auto-reboot if:

- High-availability system
- 24/7 operations
- Requires manual testing after updates
- Cluster/load-balanced setup

Enable auto-reboot if:

- Non-critical system
- Scheduled maintenance window available
- Security is top priority
- Can tolerate brief downtime

Current setting: ☐ Auto-reboot enabled ☐ Auto-reboot disabled
If enabled, reboot time: _____ (default: 03:00)

5.1.4 1.4 Filesystem Integrity Checking (AIDE)

1.4.1 Ensure AIDE is installed

```
# Install AIDE
apt install -y aide aide-common

# Check installation
dpkg -s aide

# Check version
aide --version
```

Checklist:

☐ AIDE installation

- Command: apt install -y aide aide-common
- Result: ☐ Success ☐ Failed

☐ AIDE version

- aide –version
- Version: _____

☐ AIDE packages installed

- aide: ☐ Installed
- aide-common: ☐ Installed

1.4.2 Ensure filesystem integrity is regularly checked

```
# Configure AIDE (edit if needed)
cat /etc/aide/aide.conf

# Initialize AIDE database (this will take time!)
echo "Initializing AIDE database (this may take 10-30 minutes)..."
aideinit

# Check progress
tail -f /var/log/aide/aide_init.log

# Once complete, move database to production
if [ -f /var/lib/aide/aide.db.new ]; then
    mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
    echo "AIDE database initialized"
else
    echo "ERROR: AIDE database initialization failed"
    exit 1
fi

# Run AIDE check (first run, should report no changes)
aide --check

# Configure cron for regular checks
cat > /etc/cron.d/aide << 'EOF'
# Run AIDE check daily at 5 AM
0 5 * * * root /usr/bin/aide --check | mail -s "AIDE Report for $(hostname)" root
EOF

# Test performance impact
echo "Testing AIDE performance impact..."
time aide --check

# Check resource usage
top -b -n 1 | grep aide
```

Checklist:

☐ Initialize AIDE database

- Command: aideinit
- Result: ☐ Success ☐ Failed

- Time taken: _____ minutes
- Database size: _____ MB

☐ Move database to production

- Command: mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
- Result: ☐ Success ☐ Failed
- Database exists: ☐ Yes ☐ No

☐ Run AIDE check

- Command: aide –check
- Result: ☐ Success ☐ Failed
- Changes detected: _____ (should be 0 on first run)
- Check duration: _____ minutes

☐ Configure cron for regular checks

- Cron file: /etc/cron.d/aide
- Cron configured: ☐ Yes ☐ No
- Schedule: Daily at _____
- Email reports: ☐ Configured ☐ Not Configured
- Email recipient: _____

☐ Performance Impact Assessment

- CPU during AIDE check: _____ %
- Memory usage: _____ MB
- I/O impact: ☐ Low ☐ Medium ☐ High
- Impact on applications: ☐ None ☐ Minor ☐ Major

☐ AIDE configuration reviewed

- Config file: /etc/aide/aide.conf
- Paths monitored: ☐ Appropriate ☐ Need adjustment
- Exclusions documented: ☐ Yes ☐ No

☐ Result: ☐ PASS ☐ FAIL

AIDE Usage Notes

After implementing AIDE:
When you make legitimate changes:

```
# Update AIDE database after legitimate changes
aide --update
mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
```

To check for changes:

```
# Manual check
aide --check

# Or check email reports from cron job
```

Common legitimate changes to update:

- After system updates
- After configuration changes
- After new software installation
- After log rotation

Investigate immediately if AIDE reports unexpected changes!

5.1.5 1.4 Secure Boot Settings

1.5.1 Ensure bootloader password is set

Setting a GRUB password protects the bootloader from unauthorized modifications, but you MUST remember this password! If forgotten, you'll need physical access and may need to boot from rescue media.
Store the password securely!

```
# Generate password hash
echo "Enter GRUB password:"
grub-mkpasswd-pbkdf2

# Copy the generated hash (starts with grub.pbkdf2.sha512...)

# Create or edit /etc/grub.d/40_custom
cat >> /etc/grub.d/40_custom << 'EOF'

# GRUB Password Protection
set superusers="root"
password_pbkdf2 root grub.pbkdf2.sha512.10000.XXXXX...
EOF

# Replace XXXXX... with your actual password hash!

# Make executable
chmod +x /etc/grub.d/40_custom

# Update GRUB
update-grub
```

```
# Verify in grub.cfg (password hash should be present)
grep "^password" /boot/grub/grub.cfg

# IMPORTANT: Test this on next reboot!
```

Checklist:

☐ GRUB password generated

- grub-mkpasswd-pbkdf2 executed: ☐ Yes ☐ No
- Password hash obtained: ☐ Yes ☐ No
- Password documented securely: ☐ Yes ☐ No

☐ GRUB configuration updated

- /etc/grub.d/40_custom edited: ☐ Yes ☐ No
- superusers set to "root": ☐ Yes ☐ No
- Password hash added: ☐ Yes ☐ No

☐ GRUB regenerated

- update-grub executed: ☐ Yes ☐ No
- No errors: ☐ Yes ☐ Errors found

☐ GRUB password verified

- grep "^password" /boot/grub/grub.cfg
- Password line present: ☐ Yes ☐ No

☐ Test plan for next reboot

- Test edit boot parameters (should require password): ☐ Planned
- Test recovery mode access (should require password): ☐ Planned
- Console access available for testing: ☐ Yes ☐ No

☐ Password security

- Password stored in password vault: ☐ Yes ☐ No
- Password shared with authorized personnel: ☐ Yes ☐ No
- Password complexity adequate: ☐ Yes ☐ No

☐ Result: ☐ PASS ☐ FAIL

1.5.2 Ensure permissions on bootloader config are configured

```
# Set permissions on grub.cfg
chown root:root /boot/grub/grub.cfg
chmod og-rwx /boot/grub/grub.cfg

# Verify
stat /boot/grub/grub.cfg
ls -l /boot/grub/grub.cfg

# Expected: -r-------- 1 root root (permissions 0400 or 0600)
```

Checklist:

☐ File permissions

- stat /boot/grub/grub.cfg
- Permissions: _____ (should be 0400 or 0600)
- Owner: _____ (should be root)
- Group: _____ (should be root)
- Result: ☐ PASS ☐ FAIL

☐ File not world-readable

- ls -l /boot/grub/grub.cfg
- Others have no permissions: ☐ Yes (PASS) ☐ No (FAIL)

1.5.3 Ensure authentication required for single user mode

```
# Check if root password is set (not locked)
grep "^root:" /etc/shadow

# The second field should NOT be * or !
# If it is, root account is locked

# If root is locked, set a password
# passwd root

# Verify root password is set
grep "^root:" /etc/shadow | cut -d: -f2 | head -c 1

# Should return $ (indicating password hash), not * or !

# Note: With GRUB password set (1.4.1), single user mode
# is already protected. This provides additional protection.
```

Checklist:

☐ Root password status

- grep "^root:" /etc/shadow
- Password field: _____
- Status: ☐ Password Set (PASS) ☐ Locked (FAIL)

☐ If root was locked

- Root password set: ☐ Yes ☐ No
- New password documented: ☐ Yes (secure location) ☐ No

☐ Single user mode protection

- GRUB password: ☐ Set (primary protection)
- Root password: ☐ Set (additional protection)
- Both protections active: ☐ Yes (best) ☐ Partial

☐ Test single user mode (if possible)

- Reboot to single user mode
- Password Required: ☐ Yes ☐ No

- Result: ☐ PASS ☐ FAIL ☐ Not Tested

Root Password Security

Root password security considerations:
Best practices:

- Use a strong, unique password for root
- Store in password vault
- Share with authorized personnel only
- Regular password rotation
- Use sudo for daily operations (don't login as root)

Documentation:

- Root password set date: _____
- Next rotation date: _____
- Authorized personnel: _____
- Password vault location: _____

5.1.6 1.6 Additional Process Hardening

1.6.1 Ensure ASLR is enabled

```
# Check current ASLR setting
sysctl kernel.randomize_va_space

# Value should be 2 (full randomization)
# 0 = disabled
# 1 = conservative randomization
# 2 = full randomization (recommended)

# Set ASLR to 2 (full randomization)
sysctl -w kernel.randomize_va_space=2

# Make persistent
echo "kernel.randomize_va_space = 2" >> /etc/sysctl.d/99-cis.conf

# Apply
sysctl -p /etc/sysctl.d/99-cis.conf

# Verify
sysctl kernel.randomize_va_space
```

Checklist:

☐ Check ASLR setting

- sysctl kernel.randomize_va_space
- Value: _____ (should be 2)
- Result: ☐ PASS ☐ FAIL

☐ Set ASLR to full randomization

- sysctl -w kernel.randomize_va_space=2
- Result: ☐ Success ☐ Failed

☐ Persistent configuration

- File: /etc/sysctl.d/99-cis.conf
- Entry present: ☐ Yes ☐ No
- Result: ☐ PASS ☐ FAIL

☐ Application compatibility

- Applications running: ☐ Normal ☐ Issues
- Notes: _____

1.6.2 Ensure prelink is not installed

```
# Check if prelink is installed
dpkg -s prelink 2>/dev/null

# If installed, remove (conflicts with ASLR)
apt purge prelink -y

# Verify removal
dpkg -s prelink 2>/dev/null
echo "Exit code: $?"  # Should be non-zero (package not found)
```

Checklist:

☐ Check prelink

- dpkg -s prelink
- Result: ☐ Not Installed (PASS) ☐ Installed (need to remove)

☐ If installed, remove

- apt purge prelink
- Result: ☐ Removed ☐ Failed ☐ N/A

☐ Verify removal

- Package removed: ☐ Yes (PASS) ☐ No (FAIL)

1.6.3 Ensure Automatic Error Reporting is not enabled

```
# Check apport status
systemctl status apport

# Disable apport service
systemctl stop apport
systemctl disable apport
systemctl mask apport

# Disable in configuration
sed -i 's/enabled=1/enabled=0/' /etc/default/apport

# Verify
systemctl status apport
grep enabled /etc/default/apport
```

Checklist:

☐ Check apport status

- systemctl status apport
- Status: ☐ Inactive (PASS) ☐ Active (need to disable)

☐ Disable apport service

- systemctl disable apport
- systemctl mask apport
- Result: ☐ Success ☐ Failed

☐ Configuration disabled

- grep enabled /etc/default/apport
- Shows: enabled=0: ☐ Yes (PASS) ☐ No (FAIL)

☐ Verify service disabled

- systemctl is-enabled apport
- Result: ☐ masked ☐ disabled (PASS)

1.6.4 Ensure core dumps are restricted

```
# Add limits configuration
grep -q "hard core" /etc/security/limits.conf || \
echo "* hard core 0" >> /etc/security/limits.conf

# Add sysctl configuration
grep -q "fs.suid_dumpable" /etc/sysctl.d/99-cis.conf || \
echo "fs.suid_dumpable = 0" >> /etc/sysctl.d/99-cis.conf

# Apply sysctl
sysctl -w fs.suid_dumpable=0

# Configure systemd coredump
mkdir -p /etc/systemd/coredump.conf.d/
cat > /etc/systemd/coredump.conf.d/99-cis.conf << 'EOF'
[Coredump]
Storage=none
ProcessSizeMax=0
EOF

# Restart systemd-coredump
systemctl daemon-reload

# Verify
ulimit -c
sysctl fs.suid_dumpable
systemctl cat systemd-coredump | grep -A 5 "\[Coredump\]"
```

Checklist:

☐ Limits configuration

- grep "hard core" /etc/security/limits.conf
- Entry present: ☐ Yes (PASS) ☐ No (FAIL)

☐ Sysctl configuration

- sysctl fs.suid_dumpable
- Value: _____ (should be 0)
- Result: ☐ PASS ☐ FAIL

☐ Systemd coredump configuration

- File: /etc/systemd/coredump.conf.d/99-cis.conf
- Storage=none: ☐ Yes ☐ No
- ProcessSizeMax=0: ☐ Yes ☐ No
- Result: ☐ PASS ☐ FAIL

☐ Test core dump restriction

- ulimit -c
- Value: _____ (should be 0)
- Result: ☐ PASS ☐ FAIL

5.1.7 1.7 Mandatory Access Control (AppArmor)

1.7.1 Ensure AppArmor is installed

```
# Install AppArmor
apt install -y apparmor apparmor-utils

# Check installation
dpkg -s apparmor
dpkg -s apparmor-utils

# Check version
apparmor_status --version
```

Checklist:

☐ AppArmor installed

- dpkg -s apparmor
- Result: ☐ Installed (PASS) ☐ Not Installed (FAIL)

☐ AppArmor utilities installed

- dpkg -s apparmor-utils
- Result: ☐ Installed (PASS) ☐ Not Installed (FAIL)

☐ AppArmor version

- apparmor_status –version
- Version: _____

1.7.2 Ensure AppArmor is enabled in bootloader

WARNING: This change requires a system reboot to take effect.
Before rebooting:

- Verify GRUB configuration is correct
- Ensure console/OOB access is available
- Test GRUB configuration syntax
- Inform stakeholders

```
# Backup GRUB configuration
cp /etc/default/grub /etc/default/grub.backup

# Edit GRUB configuration to enable AppArmor
# Check if already present
if grep -q "apparmor=1 security=apparmor" /etc/default/grub; then
    echo "AppArmor already enabled in GRUB"
else
    # Add AppArmor parameters
    sed -i 's/GRUB_CMDLINE_LINUX="\(.*\)"/GRUB_CMDLINE_LINUX="\1 apparmor=1 security=
    apparmor"/' /etc/default/grub
fi

# Update GRUB
update-grub

# Verify in grub.cfg
grep "apparmor=1 security=apparmor" /boot/grub/grub.cfg

# Check current kernel parameters (before reboot)
cat /proc/cmdline | grep apparmor

# REBOOT REQUIRED
# After reboot, verify:
# cat /proc/cmdline | grep apparmor
# Should show: apparmor=1 security=apparmor
```

Checklist:

☐ Backup GRUB configuration

- Backup created: ☐ Yes ☐ No
- Location: /etc/default/grub.backup

☐ GRUB configuration updated

- sed command executed: ☐ Yes ☐ No
- Parameters added: apparmor=1 security=apparmor

☐ GRUB regenerated

- update-grub executed: ☐ Yes ☐ No
- No errors: ☐ Yes ☐ Errors found

☐ Verify in grub.cfg

- grep "apparmor=1 security=apparmor" /boot/grub/grub.cfg
- Result: ☐ Found (PASS) ☐ Not Found (FAIL)

☐ Reboot scheduled

- Reboot time: _____
- Stakeholders notified: ☐ Yes ☐ No
- Console access ready: ☐ Yes ☐ No

☐ POST-REBOOT: Verify kernel parameters

- cat /proc/cmdline
- apparmor=1 present: ☐ Yes ☐ No
- security=apparmor present: ☐ Yes ☐ No
- Result: ☐ PASS ☐ FAIL

☐ POST-REBOOT: System boots successfully

- Boot successful: ☐ Yes ☐ No
- All services started: ☐ Yes ☐ Some failed
- SSH accessible: ☐ Yes ☐ No

1.7.3 Ensure AppArmor profiles are loaded

```
# Check AppArmor status (must be done AFTER reboot from 1.6.2)
aa-status

# Should show:
# - apparmor module is loaded
# - profiles are loaded
# - profiles are in enforce or complain mode

# Count profiles
aa-status | grep "profiles are loaded"
aa-status | grep "profiles are in enforce mode"
aa-status | grep "profiles are in complain mode"

# Check for unconfined processes
aa-unconfined

# List available profiles
ls /etc/apparmor.d/
```

Checklist:

☐ AppArmor module loaded

- aa-status shows module loaded: ☐ Yes ☐ No

☐ Profiles loaded

- Total profiles: _____
- Enforce mode: _____
- Complain mode: _____
- Unconfined: _____

☐ No unconfined processes (or documented)

- aa-unconfined
- Result: ☐ None (PASS) ☐ Some (document)
- Unconfined processes: _____

☐ Result: ☐ PASS ☐ FAIL

1.7.4 Ensure all AppArmor profiles are enforcing

IMPORTANT: Setting all profiles to enforce mode may break applications!
Recommended approach:

1. Start with complain mode to identify issues
2. Monitor logs for denials
3. Test application functionality
4. Move to enforce mode gradually
5. Document exceptions

Have rollback ready!

```
# Phase 1: Set all profiles to COMPLAIN mode first (safer)
echo "Phase 1: Setting profiles to complain mode for testing..."
aa-complain /etc/apparmor.d/*

# Wait and monitor
echo "Monitor for 15 minutes, checking logs..."
sleep 900

# Check for denials
grep -i "apparmor.*DENIED" /var/log/syslog | tail -50

# Phase 2: If no critical issues, set to ENFORCE mode
echo "Phase 2: Setting profiles to enforce mode..."
aa-enforce /etc/apparmor.d/*

# Verify
aa-status | grep "profiles are in enforce mode"
aa-status | grep "profiles are in complain mode"

# Check for unconfined processes
aa-unconfined

# Monitor for denials
echo "Monitoring for denials (check continuously)..."
tail -f /var/log/syslog | grep -i "apparmor.*DENIED"
```

Phase 1: Complain Mode Testing Checklist:

☐ Set profiles to complain mode

- Command: aa-complain /etc/apparmor.d/*
- Result: ☐ Success ☐ Failed

☐ Monitor for 15 minutes

- Monitoring period: _____ minutes
- Denials found: _____

☐ Application testing in complain mode

- ☐ Web Server: ☐ Working ☐ Issues
- ☐ Database: ☐ Working ☐ Issues
- ☐ Application: ☐ Working ☐ Issues
- ☐ SSH: ☐ Working ☐ Issues
- ☐ Custom services: ☐ Working ☐ Issues

☐ Review AppArmor denials

- grep -i "apparmor.*DENIED" /var/log/syslog
- Critical denials: _____
- Can be resolved: ☐ Yes ☐ No

Phase 2: Enforce Mode Implementation Checklist:

☐ Set profiles to enforce mode

- Command: aa-enforce /etc/apparmor.d/*
- Result: ☐ Success ☐ Failed

☐ Verify enforcement

- aa-status | grep "profiles are in enforce mode"
- Count: _____
- All profiles enforced: ☐ Yes ☐ Some in complain

☐ Immediate application testing

- ☐ Web Server: ☐ Working ☐ Broken
- ☐ Database: ☐ Working ☐ Broken
- ☐ Application: ☐ Working ☐ Broken
- ☐ SSH: ☐ Working ☐ Broken
- ☐ All critical services: ☐ Working ☐ Issues

☐ Monitor for active denials

- tail -f /var/log/syslog | grep -i "apparmor.*DENIED"
- Active denials blocking operations: ☐ Yes ☐ No

☐ If services break:

- ☐ Identify broken service
- ☐ Set to complain mode: aa-complain /etc/apparmor.d/service
- ☐ Document exception
- ☐ Create custom profile or wait for updates

Exception Handling:

```
# If specific service has issues, set only that to complain mode
# Example: nginx
aa-complain /etc/apparmor.d/usr.sbin.nginx

# Verify
aa-status | grep nginx

# Document the exception
cat >> /tmp/cis_exceptions.txt << EOF
AppArmor Exception:
- Service: nginx
- Reason: Application requires access not in default profile
- Profile: /etc/apparmor.d/usr.sbin.nginx
- Mode: complain
- Date: $(date)
- Approved by: [Name]
- Compensating controls: [List]
- Review date: [Date]
EOF
```

Exception Documentation Checklist:

☐ Profiles in complain mode (exceptions)

- Profile 1: _____
- Reason: _____
- Profile 2: _____
- Reason: _____
- Profile 3: _____
- Reason: _____

☐ Exception approval

- Approved by: _____
- Date: _____
- Review scheduled: _____

☐ Compensating controls

- Control 1: _____
- Control 2: _____
- Control 3: _____

Troubleshooting AppArmor Issues

If application fails after enforcing AppArmor:
Step 1: Identify the issue

```
# Check logs for denials
grep -i "apparmor.*DENIED" /var/log/syslog | tail -50

# Check which profile is causing issues
aa-status | grep -A 10 "processes are in enforce mode"
```

Step 2: Immediate remediation

```
# Option A: Set specific profile to complain mode
aa-complain /etc/apparmor.d/problematic-profile

# Option B: Temporarily disable AppArmor for testing
# (NOT RECOMMENDED for production)
# systemctl stop apparmor
# aa-teardown
```

Step 3: Long-term solution

- Create custom profile with aa-genprof
- Modify existing profile to allow required access
- Document as approved exception
- Implement compensating controls

Step 4: Create custom profile

```
# Generate profile for application
aa-genprof /path/to/application

# Follow prompts, test application
# Allow necessary operations
# Save profile
```

Final AppArmor Checklist:

☐ AppArmor enabled and active

☐ Profiles loaded

☐ Majority of profiles in enforce mode

- Enforce mode count: _____
- Complain mode count: _____
- Enforcement ratio: _____% (target: >80%)

☐ All critical applications working

☐ Exceptions documented and approved

☐ Monitoring in place for denials

☐ Overall Result: ☐ PASS ☐ PASS WITH EXCEPTIONS ☐ FAIL

5.1.8 1.8 Command Line Warning Banners

1.8.1 Ensure message of the day is configured properly

```
# Configure /etc/motd with security warning
cat > /etc/motd << 'EOF'
################################################################################
#                        AUTHORIZED ACCESS ONLY                                #
#                                                                              #
#  This system is for authorized use only. Individuals using this system      #
#  without authority or in excess of their authority are subject to having    #
#  all their activities on this system monitored and recorded.                #
#                                                                              #
#  Anyone using this system expressly consents to such monitoring and is      #
#  advised that if such monitoring reveals possible evidence of criminal      #
#  activity, system personnel may provide the evidence to law enforcement.    #
################################################################################
EOF

# Set permissions
chmod 644 /etc/motd
chown root:root /etc/motd

# Verify
cat /etc/motd
stat /etc/motd
```

Checklist:

☐ /etc/motd configured with warning

- Banner created: ☐ Yes ☐ No
- Appropriate warning text: ☐ Yes ☐ No

☐ No OS information disclosed

- `cat /etc/motd | grep -i "ubuntu\|linux\|version"`
- Result: ☐ No OS info (PASS) ☐ OS info present (FAIL)

☐ Permissions set correctly

- Permissions: _____ (should be 644)
- Owner: _____:group (should be root:root)
- Result: ☐ PASS ☐ FAIL

☐ Banner displays on login

- Test login and verify banner appears
- Result: ☐ Displays (PASS) ☐ Not displaying (FAIL)

1.8.2 Ensure local login warning banner is configured properly

```
# Configure /etc/issue for console login
cat > /etc/issue << 'EOF'
################################################################################
#                        AUTHORIZED ACCESS ONLY                                #
################################################################################
EOF
```

```
# Set permissions
chmod 644 /etc/issue
chown root:root /etc/issue

# Verify
cat /etc/issue
ls -l /etc/issue
```

Checklist:

☐ /etc/issue configured

☐ No system information disclosed

☐ Permissions: 644, root:root

☐ Result: ☐ PASS ☐ FAIL

1.8.3 Ensure remote login warning banner is configured properly

```
# Configure /etc/issue.net for network login
cat > /etc/issue.net << 'EOF'
################################################################################
#                        AUTHORIZED ACCESS ONLY                                #
################################################################################
EOF

# Set permissions
chmod 644 /etc/issue.net
chown root:root /etc/issue.net

# Configure SSH to use banner (will be detailed in section 5.2)
grep -q "^Banner" /etc/ssh/sshd_config || \
echo "Banner /etc/issue.net" >> /etc/ssh/sshd_config

# Restart SSH
systemctl restart sshd

# Verify
cat /etc/issue.net
grep "^Banner" /etc/ssh/sshd_config
```

Checklist:

☐ /etc/issue.net configured

☐ SSH Banner directive set

☐ SSH restarted successfully

☐ Banner displays on SSH login

- Test from remote: ssh user@server
- Banner appears: ☐ Yes (PASS) ☐ No (FAIL)

☐ Result: ☐ PASS ☐ FAIL

1.8.4 Ensure permissions on /etc/motd are configured

```
# Set permissions on /etc/motd
chmod u-x,go-wx /etc/motd
chown root:root /etc/motd

# Verify
stat /etc/motd
ls -l /etc/motd
```

Checklist:

☐ Owner: root:root

☐ Permissions: 644 or more restrictive

☐ Result: ☐ PASS ☐ FAIL

1.8.5 Ensure permissions on /etc/issue are configured

```
# Set permissions on /etc/issue
chmod u-x,go-wx /etc/issue
chown root:root /etc/issue

# Verify
stat /etc/issue
```

Checklist:

☐ Owner: root:root

☐ Permissions: 644 or more restrictive

☐ Result: ☐ PASS ☐ FAIL

1.8.6 Ensure permissions on /etc/issue.net are configured

```
# Set permissions on /etc/issue.net
chmod u-x,go-wx /etc/issue.net
chown root:root /etc/issue.net

# Verify
stat /etc/issue.net
```

Checklist:

☐ Owner: root:root

☐ Permissions: 644 or more restrictive

☐ Result: ☐ PASS ☐ FAIL

5.1.9 1.9 GNOME Display Manager

1.9.1 Ensure GNOME Display Manager is removed or configured

```
# Check if GDM is installed
dpkg -l | grep gdm3

# Option 1: Remove GDM (recommended for servers)
apt purge gdm3 -y

# Option 2: Configure GDM banner (if GUI is needed)
if dpkg -l | grep -q gdm3; then
    mkdir -p /etc/dconf/db/gdm.d/
    cat > /etc/dconf/db/gdm.d/01-banner-message << 'EOF'
[org/gnome/login-screen]
banner-message-enable=true
banner-message-text='Authorized access only. All activity is monitored.'
EOF

    # Update dconf
    dconf update

    # Verify
    grep -r "banner-message" /etc/dconf/db/gdm.d/
fi
```

Checklist:

☐ GDM installation status

- Check: dpkg -l | grep gdm3
- Result: ☐ Removed ☐ Configured ☐ N/A

☐ If removed:

- apt purge gdm3 completed: ☐ Yes
- No GUI needed: ☐ Confirmed

☐ If configured:

- Banner enabled: ☐ Yes ☐ No
- Banner text appropriate: ☐ Yes ☐ No
- dconf updated: ☐ Yes ☐ No

☐ Result: ☐ PASS ☐ FAIL ☐ N/A

Section 1 - Initial Setup Summary

Completion Checklist for Section 1:

- ☐ **1.1 Filesystem Configuration**
  - ☐ All unnecessary kernel modules disabled (cramfs, freevxfs, jffs2, hfs, hfsplus, udf)
  - ☐ usb-storage module disabled (if not needed)
  - ☐ squashfs handled appropriately (with snap exception consideration)
  - ☐ All partition mount options configured (/tmp, /var, /home, /var/tmp, /var/log, /dev/shm)
  - ☐ nodev, nosuid, noexec options applied where required
- ☐ **1.2 Package Management**
  - ☐ GPG keys properly configured and verified
  - ☐ Package repositories configured correctly
  - ☐ Security updates identified and documented
  - ☐ Update policy established (manual/automatic)
- ☐ **1.3 Configure Bootloader**
  - ☐ GRUB password configured and enforced
  - ☐ Bootloader authentication enabled
  - ☐ Bootloader configuration files properly secured
- ☐ **1.4 Filesystem Integrity Checking**
  - ☐ AIDE installed and configured
  - ☐ Initial database created
  - ☐ Regular integrity checking scheduled
- ☐ **1.5 Secure Boot Settings**
  - ☐ Bootloader password requirements enforced
  - ☐ Single-user mode protected
  - ☐ Boot configuration secured
- ☐ **1.6 Additional Process Hardening**
  - ☐ ASLR enabled and configured
  - ☐ Prelink removed or disabled
  - ☐ Error reporting disabled
  - ☐ Core dumps restricted
- ☐ **1.7 Mandatory Access Control (AppArmor)**
  - ☐ AppArmor installed and enabled
  - ☐ All profiles loaded and in enforce/complain mode
  - ☐ Enforcement implemented (with exception handling)

☐ **1.8 Command Line Warning Banners**

- ☐ Login banners configured for MOTD, local and remote access
- ☐ Banner file permissions properly set (644, root:root)
- ☐ Legal warnings displayed appropriately

☐ **1.9 GNOME Display Manager**

- ☐ GDM removed (servers) or configured with banners (desktops)
- ☐ GUI security settings applied

Section 1 Overall Status: _____
Exceptions Documented: _____
Next Steps: Proceed to Section 2 - Services

5.2 Section 2: Services

Service Hardening

This section ensures unnecessary services are disabled or removed to minimize attack surface.
Important: Verify each service is not needed before disabling. Check with application owners!

5.2.1 2.1 Special Purpose Services

2.1.1 Time Synchronization

```
# Ensure time synchronization is in use
systemctl status systemd-timesyncd

# If not active, enable it
systemctl enable systemd-timesyncd
systemctl start systemd-timesyncd

# Configure time sync
timedatectl set-ntp true

# Verify
timedatectl status
timedatectl show-timesync --all

# Check time sources
timedatectl timesync-status
```

Checklist:

☐ Time synchronization enabled

- systemctl status systemd-timesyncd
- Status: ☐ Active (PASS) ☐ Inactive (FAIL)

☐ NTP synchronized

- timedatectl status
- "System clock synchronized: yes": ☐ Yes ☐ No
- "NTP service: active": ☐ Yes ☐ No

☐ Time source configured

- timedatectl show-timesync –all
- Server: _____
- Poll interval: _____

☐ Result: ☐ PASS ☐ FAIL

2.2 Ensure unnecessary services are not installed

2.2.1 Ensure X Window System is not installed

```
# Check for X Window System
dpkg -l xserver-xorg* 2>/dev/null

# If installed and not needed, remove
```

```
apt purge xserver-xorg* -y

# Verify removal
dpkg -l xserver-xorg* 2>/dev/null
```

Checklist:

☐ X Window System check

- dpkg -l xserver-xorg*
- Result: ☐ Not Installed (PASS) ☐ Installed

☐ If installed:

- GUI required: ☐ Yes (exception) ☐ No (remove)
- If removed: ☐ Success ☐ Failed

2.2.2 Ensure Avahi Server is not installed

```
# Check Avahi
systemctl status avahi-daemon 2>/dev/null
dpkg -l avahi-daemon 2>/dev/null

# Remove if not needed
apt purge avahi-daemon -y

# Verify
systemctl status avahi-daemon 2>/dev/null
```

Checklist:

☐ Avahi Server

- Status: ☐ Not Installed (PASS) ☐ Installed (remove)
- If removed: ☐ Success

2.2.3 Ensure CUPS is not installed

```
# Check CUPS
systemctl status cups 2>/dev/null
dpkg -l cups 2>/dev/null

# Remove if not needed (servers typically don't need printing)
apt purge cups -y

# Verify
dpkg -l cups 2>/dev/null
```

Checklist:

☐ CUPS

- Status: ☐ Not Installed (PASS) ☐ Installed
- Printing needed: ☐ No (remove) ☐ Yes (exception)

2.2.4 Ensure DHCP Server is not installed

```
# Check DHCP Server
systemctl status isc-dhcp-server 2>/dev/null
dpkg -l isc-dhcp-server 2>/dev/null

# Remove
apt purge isc-dhcp-server -y
```

Checklist:

☐ DHCP Server: ☐ Not Installed (PASS) ☐ Removed

2.2.5 Ensure LDAP server is not installed

```
# Check LDAP
systemctl status slapd 2>/dev/null
dpkg -l slapd 2>/dev/null

# Remove if not LDAP server
apt purge slapd -y
```

Checklist:

☐ LDAP Server: ☐ Not Installed (PASS) ☐ Exception (LDAP server)

2.2.6 Ensure NFS is not installed

```
# Check NFS
systemctl status nfs-server 2>/dev/null
dpkg -l nfs-kernel-server 2>/dev/null

# Remove if not NFS server
apt purge nfs-kernel-server -y
```

Checklist:

☐ NFS: ☐ Not Installed (PASS) ☐ Exception (NFS required)

2.2.7 Ensure DNS Server is not installed

```
# Check DNS Server
systemctl status bind9 2>/dev/null
dpkg -l bind9 2>/dev/null

# Remove
apt purge bind9 -y
```

Checklist:

☐ DNS Server: ☐ Not Installed (PASS) ☐ Exception

2.2.8 Ensure FTP Server is not installed

```
# Check FTP
systemctl status vsftpd 2>/dev/null
dpkg -l vsftpd 2>/dev/null

# Remove
apt purge vsftpd -y
```

Checklist:

☐ FTP Server: ☐ Not Installed (PASS) ☐ Removed

2.2.9 Ensure HTTP server is not installed (unless required)

```
# Check HTTP servers
systemctl status apache2 2>/dev/null
systemctl status nginx 2>/dev/null

# Only remove if NOT a web server
# If this IS a web server, document exception
if [ "$IS_WEB_SERVER" != "yes" ]; then
    apt purge apache2 nginx -y
fi
```

Checklist:

☐ HTTP Server

- Is web server: ☐ Yes (exception) ☐ No (remove)
- If exception, documented: ☐ Yes

2.2.10 Ensure IMAP/POP3 server is not installed

```
# Check mail servers
systemctl status dovecot 2>/dev/null
dpkg -l dovecot-imapd dovecot-pop3d 2>/dev/null

# Remove
apt purge dovecot-imapd dovecot-pop3d -y
```

Checklist:

☐ IMAP/POP3: ☐ Not Installed (PASS) ☐ Removed

2.2.11 Ensure Samba is not installed

```
# Check Samba
systemctl status smbd 2>/dev/null
dpkg -l samba 2>/dev/null

# Remove if not file server
apt purge samba -y
```

Checklist:

☐ Samba: ☐ Not Installed (PASS) ☐ Exception (file server)

2.2.12 Ensure HTTP Proxy Server is not installed

```
# Check Squid
systemctl status squid 2>/dev/null
dpkg -l squid 2>/dev/null

# Remove
apt purge squid -y
```

Checklist:

☐ HTTP Proxy: ☐ Not Installed (PASS) ☐ Removed

2.2.13 Ensure SNMP Server is not installed

```
# Check SNMP
systemctl status snmpd 2>/dev/null
dpkg -l snmpd 2>/dev/null

# Remove (use secure monitoring instead)
apt purge snmpd -y
```

Checklist:

☐ SNMP Server: ☐ Not Installed (PASS) ☐ Removed

2.2.14 Ensure rsync service is not installed

```
# Check rsync service (not rsync command)
systemctl status rsync 2>/dev/null
dpkg -l rsync 2>/dev/null

# Disable rsync service (keep rsync command)
systemctl disable rsync 2>/dev/null
systemctl stop rsync 2>/dev/null
systemctl mask rsync 2>/dev/null
```

Checklist:

☐ rsync service: ☐ Disabled (PASS) ☐ Exception

Service Removal Summary Checklist:

☐ All unnecessary services identified

☐ All removals approved by application owners

☐ No application dependencies on removed services

☐ No errors in application logs after removal

☐ All required services still running

- systemctl list-units –type=service –state=failed
- Failed services: _____ (should be 0 or expected)

☐ Exceptions documented

- Services kept: _____
- Business justification: _____

5.3 Section 3: Network Configuration

Network configuration changes can lock you out!
MANDATORY before proceeding:

- Console/OOB access MUST be available and tested
- Never close current SSH session until new one verified
- Test each change incrementally
- Have rollback commands ready
- Inform all stakeholders

5.3.1 3.1 Disable Unused Network Protocols and Devices

3.1.1 Disable DCCP

```
# Disable DCCP protocol
echo "install dccp /bin/true" > /etc/modprobe.d/dccp.conf
rmmod dccp 2>/dev/null

# Verify
lsmod | grep dccp
modprobe -n -v dccp
```

Checklist:

☐ DCCP disabled: ☐ PASS ☐ FAIL ☐ N/A

3.1.2 Disable SCTP

```
# Disable SCTP protocol
echo "install sctp /bin/true" > /etc/modprobe.d/sctp.conf
rmmod sctp 2>/dev/null

# Verify
lsmod | grep sctp
```

Checklist:

☐ SCTP disabled: ☐ PASS ☐ FAIL ☐ N/A

3.1.3 Disable RDS

```
# Disable RDS protocol
echo "install rds /bin/true" > /etc/modprobe.d/rds.conf
rmmod rds 2>/dev/null

# Verify
lsmod | grep rds
```

Checklist:

☐ RDS disabled: ☐ PASS ☐ FAIL ☐ N/A

3.1.4 Disable TIPC

```
# Disable TIPC protocol
echo "install tipc /bin/true" > /etc/modprobe.d/tipc.conf
rmmod tipc 2>/dev/null

# Verify
lsmod | grep tipc
```

Checklist:

☐ TIPC disabled: ☐ PASS ☐ FAIL ☐ N/A

3.1.5 Ensure wireless interfaces are disabled

NEW: Wireless Interface Control

This control was missing in the original script. Wireless interfaces on servers should be disabled to reduce attack surface.

```
# Check for wireless interfaces
echo "Checking for wireless interfaces..."
ip link show
nmcli device status 2>/dev/null

# Find wireless interfaces
WIRELESS_INTERFACES=$(find /sys/class/net/*/ -type l -name wireless 2>/dev/null | awk -F
    '/' '{print $5}')

if [ -n "$WIRELESS_INTERFACES" ]; then
    echo "Wireless interfaces found: $WIRELESS_INTERFACES"

    # Method 1: Using rfkill (if available)
    if command -v rfkill >/dev/null 2>&1; then
        apt install -y rfkill
        rfkill list
        rfkill block all
        echo "Wireless blocked using rfkill"
    fi

    # Method 2: Disable specific interfaces
    for iface in $WIRELESS_INTERFACES; do
        echo "Disabling interface: $iface"
        ip link set $iface down

        # Disable in NetworkManager (if used)
        if command -v nmcli >/dev/null 2>&1; then
            nmcli device set $iface managed no
        fi
    done

    # Method 3: Blacklist wireless drivers (permanent)
    echo "# Blacklist wireless drivers" > /etc/modprobe.d/blacklist-wireless.conf
    echo "blacklist iwlwifi" >> /etc/modprobe.d/blacklist-wireless.conf
    echo "blacklist ath9k" >> /etc/modprobe.d/blacklist-wireless.conf
    echo "blacklist rtl8192ce" >> /etc/modprobe.d/blacklist-wireless.conf
    echo "blacklist rtl8192cu" >> /etc/modprobe.d/blacklist-wireless.conf
    echo "blacklist b43" >> /etc/modprobe.d/blacklist-wireless.conf
    echo "blacklist bcma" >> /etc/modprobe.d/blacklist-wireless.conf
```

```
    # Update initramfs
    update-initramfs -u

    echo "Wireless interfaces disabled"
else
    echo "No wireless interfaces found"
fi

# Verify
ip link show | grep -i "wlan\|wifi\|wireless"
rfkill list 2>/dev/null
```

Checklist:

☐ Check for wireless interfaces

- Command: ip link show
- Wireless interfaces found: _____
- Count: _____

☐ Disable wireless (if present)

- Method used: ☐ rfkill ☐ ip link ☐ driver blacklist
- Result: ☐ Success ☐ Failed ☐ N/A (no wireless)

☐ Verify wireless disabled

- ip link show | grep -i wireless
- Result: ☐ No interfaces ☐ Interfaces down ☐ Still active

☐ Driver blacklist configured

- File: /etc/modprobe.d/blacklist-wireless.conf
- Created: ☐ Yes ☐ No ☐ N/A
- initramfs updated: ☐ Yes ☐ No ☐ N/A

☐ Business requirement check

- Wireless required for operations: ☐ Yes ☐ No
- If Yes, document exception: _____
- Compensating controls: _____

☐ Reboot test (if drivers blacklisted)

- Reboot performed: ☐ Yes ☐ No
- Wireless still disabled: ☐ Yes ☐ No

☐ Overall Result: ☐ PASS ☐ FAIL ☐ N/A ☐ EXCEPTION

Exception Handling

If wireless is required (rare for servers):
Document the exception:

- Business justification: _____
- Approved by: _____ Date: _____
- Compensating controls:
  - WPA3 encryption required
  - 802.1X authentication
  - Network segmentation
  - IDS/IPS monitoring
  - Regular security assessments
- Review date: _____

Summary: Verify all unused protocols disabled

```
# Verify all protocols disabled
echo "=== Checking disabled protocols ==="
lsmod | grep -E "(dccp|sctp|rds|tipc)"

# Should return nothing
if [ $? -eq 0 ]; then
    echo "WARNING: Some protocols still loaded!"
else
    echo "All protocols disabled: PASS"
fi

# Check wireless
ip link show | grep -i "wlan\|wifi" || echo "No wireless interfaces: PASS"
```

Overall Protocol Disable Checklist:

☐ DCCP disabled: ☐ PASS ☐ FAIL

☐ SCTP disabled: ☐ PASS ☐ FAIL

☐ RDS disabled: ☐ PASS ☐ FAIL

☐ TIPC disabled: ☐ PASS ☐ FAIL

☐ Wireless disabled: ☐ PASS ☐ FAIL ☐ N/A ☐ EXCEPTION

☐ Overall Result: ☐ PASS ☐ FAIL

5.3.2 3.2 Network Parameters (Host Only)

3.2.1 Ensure IP forwarding is disabled

```
# Disable IPv4 forwarding
sysctl -w net.ipv4.ip_forward=0
echo "net.ipv4.ip_forward = 0" >> /etc/sysctl.d/60-netipv4_sysctl.conf

# Disable IPv6 forwarding
```

```
sysctl -w net.ipv6.conf.all.forwarding=0
echo "net.ipv6.conf.all.forwarding = 0" >> /etc/sysctl.d/60-netipv6_sysctl.conf

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf
sysctl -p /etc/sysctl.d/60-netipv6_sysctl.conf

# Verify
sysctl net.ipv4.ip_forward
sysctl net.ipv6.conf.all.forwarding
```

Checklist:

☐ IPv4 forwarding disabled (0): ☐ PASS ☐ FAIL

☐ IPv6 forwarding disabled (0): ☐ PASS ☐ FAIL

☐ Configuration persisted: ☐ PASS ☐ FAIL

3.2.2 Ensure packet redirect sending is disabled

```
# Disable packet redirect sending
sysctl -w net.ipv4.conf.all.send_redirects=0
sysctl -w net.ipv4.conf.default.send_redirects=0

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf

# Verify
sysctl net.ipv4.conf.all.send_redirects
sysctl net.ipv4.conf.default.send_redirects
```

Checklist:

☐ Send redirects disabled: ☐ PASS ☐ FAIL

☐ Configuration persisted: ☐ PASS ☐ FAIL

3.2.3 Ensure source routed packets are not accepted

```
# Disable source routing for IPv4
sysctl -w net.ipv4.conf.all.accept_source_route=0
sysctl -w net.ipv4.conf.default.accept_source_route=0

# Disable source routing for IPv6
sysctl -w net.ipv6.conf.all.accept_source_route=0
sysctl -w net.ipv6.conf.default.accept_source_route=0

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
EOF

cat >> /etc/sysctl.d/60-netipv6_sysctl.conf << 'EOF'
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0
```

```
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf
sysctl -p /etc/sysctl.d/60-netipv6_sysctl.conf

# Verify
sysctl net.ipv4.conf.all.accept_source_route
sysctl net.ipv6.conf.all.accept_source_route
```

Checklist:

☐ IPv4 source routing disabled: ☐ PASS ☐ FAIL

☐ IPv6 source routing disabled: ☐ PASS ☐ FAIL

3.2.4 Ensure ICMP redirects are not accepted

```
# Disable ICMP redirects for IPv4
sysctl -w net.ipv4.conf.all.accept_redirects=0
sysctl -w net.ipv4.conf.default.accept_redirects=0

# Disable ICMP redirects for IPv6
sysctl -w net.ipv6.conf.all.accept_redirects=0
sysctl -w net.ipv6.conf.default.accept_redirects=0

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
EOF

cat >> /etc/sysctl.d/60-netipv6_sysctl.conf << 'EOF'
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf
sysctl -p /etc/sysctl.d/60-netipv6_sysctl.conf

# Verify
sysctl net.ipv4.conf.all.accept_redirects
sysctl net.ipv6.conf.all.accept_redirects
```

Checklist:

☐ ICMP redirects disabled: ☐ PASS ☐ FAIL

3.2.5 Ensure secure ICMP redirects are not accepted

```
# Disable secure ICMP redirects
sysctl -w net.ipv4.conf.all.secure_redirects=0
sysctl -w net.ipv4.conf.default.secure_redirects=0

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
EOF

# Apply
```

```
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf

# Verify
sysctl net.ipv4.conf.all.secure_redirects
```

Checklist:

☐ Secure ICMP redirects disabled: ☐ PASS ☐ FAIL

3.2.6 Ensure suspicious packets are logged

```
# Enable logging of martian packets
sysctl -w net.ipv4.conf.all.log_martians=1
sysctl -w net.ipv4.conf.default.log_martians=1

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf

# Verify
sysctl net.ipv4.conf.all.log_martians

# Check logs for martian packets
dmesg | grep "martian source" | tail -10
```

Checklist:

☐ Martian packet logging enabled: ☐ PASS ☐ FAIL

☐ Logs checked: ☐ Yes ☐ No

3.2.7 Ensure broadcast ICMP requests are ignored

```
# Ignore broadcast ICMP requests
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.icmp_echo_ignore_broadcasts = 1
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf

# Verify
sysctl net.ipv4.icmp_echo_ignore_broadcasts
```

Checklist:

☐ Broadcast ICMP ignored: ☐ PASS ☐ FAIL

3.2.8 Ensure bogus ICMP responses are ignored

```
# Ignore bogus ICMP error responses
sysctl -w net.ipv4.icmp_ignore_bogus_error_responses=1

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
```

```
net.ipv4.icmp_ignore_bogus_error_responses = 1
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf

# Verify
sysctl net.ipv4.icmp_ignore_bogus_error_responses
```

Checklist:

☐ Bogus ICMP responses ignored: ☐ PASS ☐ FAIL

3.2.9 Ensure Reverse Path Filtering is enabled

```
# Enable reverse path filtering
sysctl -w net.ipv4.conf.all.rp_filter=1
sysctl -w net.ipv4.conf.default.rp_filter=1

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf

# Verify
sysctl net.ipv4.conf.all.rp_filter
```

Checklist:

☐ Reverse path filtering enabled: ☐ PASS ☐ FAIL

3.2.10 Ensure TCP SYN Cookies is enabled

```
# Enable TCP SYN cookies (protection against SYN flood attacks)
sysctl -w net.ipv4.tcp_syncookies=1

cat >> /etc/sysctl.d/60-netipv4_sysctl.conf << 'EOF'
net.ipv4.tcp_syncookies = 1
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv4_sysctl.conf

# Verify
sysctl net.ipv4.tcp_syncookies
```

Checklist:

☐ TCP SYN Cookies enabled: ☐ PASS ☐ FAIL

3.2.11 Ensure IPv6 router advertisements are not accepted

```
# Disable IPv6 router advertisements
sysctl -w net.ipv6.conf.all.accept_ra=0
sysctl -w net.ipv6.conf.default.accept_ra=0

cat >> /etc/sysctl.d/60-netipv6_sysctl.conf << 'EOF'
```

```
net.ipv6.conf.all.accept_ra = 0
net.ipv6.conf.default.accept_ra = 0
EOF

# Apply
sysctl -p /etc/sysctl.d/60-netipv6_sysctl.conf

# Verify
sysctl net.ipv6.conf.all.accept_ra
```

Checklist:

☐ IPv6 RA disabled: ☐ PASS ☐ FAIL ☐ N/A

Summary: Verify all network parameters

```
# Comprehensive network parameter check
echo "=== Network Parameters Summary ==="
sysctl -a | grep -E "(ip_forward|accept_source_route|accept_redirects|send_redirects|
    log_martians|icmp_echo_ignore_broadcasts|icmp_ignore_bogus_error_responses|rp_filter
    |tcp_syncookies|accept_ra)" | sort

# All should be correct values
# Expected output:
# net.ipv4.conf.all.accept_redirects = 0
# net.ipv4.conf.all.accept_source_route = 0
# net.ipv4.conf.all.log_martians = 1
# net.ipv4.conf.all.rp_filter = 1
# net.ipv4.conf.all.secure_redirects = 0
# net.ipv4.conf.all.send_redirects = 0
# net.ipv4.conf.default.* = [same as above]
# net.ipv4.icmp_echo_ignore_broadcasts = 1
# net.ipv4.icmp_ignore_bogus_error_responses = 1
# net.ipv4.ip_forward = 0
# net.ipv4.tcp_syncookies = 1
# net.ipv6.conf.all.accept_ra = 0
# net.ipv6.conf.all.accept_redirects = 0
# net.ipv6.conf.all.accept_source_route = 0
# net.ipv6.conf.all.forwarding = 0
```

Network Parameters Summary Checklist:

☐ IP forwarding disabled: ☐ PASS ☐ FAIL

☐ Send redirects disabled: ☐ PASS ☐ FAIL

☐ Source routed packets not accepted: ☐ PASS ☐ FAIL

☐ ICMP redirects not accepted: ☐ PASS ☐ FAIL

☐ Secure ICMP redirects not accepted: ☐ PASS ☐ FAIL

☐ Suspicious packets logged: ☐ PASS ☐ FAIL

☐ Broadcast ICMP ignored: ☐ PASS ☐ FAIL

☐ Bogus ICMP ignored: ☐ PASS ☐ FAIL

☐ Reverse Path Filtering enabled: ☐ PASS ☐ FAIL

☐ TCP SYN Cookies enabled: ☐ PASS ☐ FAIL

☐ IPv6 router advertisements not accepted: ☐ PASS ☐ FAIL

☐ Configuration persisted in sysctl.d: ☐ PASS ☐ FAIL

☐ Overall Result: ☐ PASS ☐ FAIL

Network Connectivity Test After Parameters:

☐ Ping test

- ping -c 4 8.8.8.8
- Result: ☐ PASS ☐ FAIL

☐ DNS resolution

- nslookup google.com
- Result: ☐ PASS ☐ FAIL

☐ Application connectivity

- All applications working: ☐ Yes ☐ Issues
- Details: _____

☐ SSH still accessible

- New SSH connection: ☐ Working ☐ Issues

5.3.3 3.3 Configure Firewall

FIREWALL CONFIGURATION IS THE HIGHEST LOCKOUT RISK!
ABSOLUTELY MANDATORY BEFORE PROCEEDING:

- ☐ Console/OOB access TESTED and WORKING
- ☐ Current SSH session kept open
- ☐ ALL required ports documented
- ☐ Rollback command ready: `ufw disable`
- ☐ Team on standby
- ☐ Change window confirmed

FAILURE TO PREPARE = GUARANTEED LOCKOUT

Pre-Firewall Preparation Checklist

MANDATORY Pre-Firewall Checklist:

☐ Console/OOB access

- Type: ☐ Physical ☐ iLO/iDRAC ☐ KVM ☐ Serial
- Tested NOW: ☐ Yes ☐ No
- **If NO: STOP! Do not proceed!**

☐ Document ALL required ports

- SSH: Port _____ From: _____
- HTTP: Port _____ From: _____
- HTTPS: Port _____ From: _____
- Application 1: Port _____ From: _____
- Application 2: Port _____ From: _____
- Monitoring: Port _____ From: _____
- Database: Port _____ From: _____
- Other: _____

☐ Current listening ports documented

- Command: ss -tulnp > /tmp/ports_before_firewall.txt
- Saved: ☐ Yes ☐ No

☐ Rollback plan

- Command ready: ufw disable
- Console access for rollback: ☐ Ready

☐ Team communication

- Team notified: ☐ Yes ☐ No
- Support on standby: ☐ Yes ☐ No

3.3.1 Ensure ufw is installed

```
# Install UFW
apt install -y ufw

# Verify installation
dpkg -s ufw
ufw version

# Check initial status (should be inactive)
ufw status verbose
```

Checklist:

☐ UFW installed: ☐ Yes ☐ No

☐ UFW version: _____

☐ Current status: ☐ Inactive (expected)

3.3.2 Ensure iptables-persistent is not installed with ufw

```
# Check for iptables-persistent
dpkg -s iptables-persistent 2>/dev/null

# If installed, remove (conflicts with UFW)
apt purge iptables-persistent -y

# Verify removal
dpkg -s iptables-persistent 2>/dev/null
echo "Exit code: $?"  # Should be non-zero
```

Checklist:

☐ iptables-persistent

- Status: ☐ Not Installed (PASS) ☐ Removed (PASS)

3.3.3 Ensure ufw service is enabled

```
# Enable UFW service to start on boot (but don't activate yet)
systemctl enable ufw.service

# Verify
systemctl is-enabled ufw.service
```

Checklist:

☐ UFW service enabled: ☐ Yes ☐ No

3.3.4 Ensure ufw loopback traffic is configured

```
# Configure loopback interface
ufw allow in on lo
ufw allow out on lo

# Deny traffic from loopback network on other interfaces
ufw deny in from 127.0.0.0/8
ufw deny in from ::1

# Show rules
ufw status numbered
```

Checklist:

☐ Loopback input allowed: ☐ Yes

☐ Loopback output allowed: ☐ Yes

☐ Non-loopback traffic from 127.0.0.0/8 denied: ☐ Yes

☐ Non-loopback traffic from ::1 denied: ☐ Yes

3.3.5 Ensure ufw outbound connections are configured

```
# Option 1: Allow all outbound (standard for most servers)
ufw default allow outgoing

# Option 2: Deny all outbound (restrictive, requires more config)
# Only use if you need extreme security
# ufw default deny outgoing
# Then allow specific outbound:
# ufw allow out 53/udp    # DNS
# ufw allow out 80/tcp    # HTTP
# ufw allow out 443/tcp   # HTTPS
# ufw allow out 123/udp   # NTP
# Add other required outbound ports

# Verify
ufw status verbose | grep "Default:"
```

Checklist:

☐ Outbound policy selected

- Policy: ☐ Allow outgoing (standard)
- Policy: ☐ Deny outgoing (restrictive)

☐ If deny outgoing, required ports configured

- DNS (53/udp): ☐ Allowed ☐ N/A
- HTTP (80/tcp): ☐ Allowed ☐ N/A
- HTTPS (443/tcp): ☐ Allowed ☐ N/A
- NTP (123/udp): ☐ Allowed ☐ N/A
- Other: _____

3.3.6 Ensure ufw firewall rules exist for all open ports

ADD SSH RULE BEFORE ANY OTHER RULES!
If you enable UFW without SSH rule, you will be locked out!

```
# CRITICAL: Add SSH rule FIRST (before anything else)
echo "Adding SSH rule (CRITICAL)..."
ufw allow 22/tcp comment 'SSH Access'

# Or restrict SSH to specific IPs (more secure)
# ufw allow from MANAGEMENT_IP to any port 22 proto tcp comment 'SSH from Management'

# Verify SSH rule added
```

```
ufw status numbered | grep 22

# NOW add other required services
echo "Adding other service rules..."

# HTTP (if web server)
ufw allow 80/tcp comment 'HTTP'

# HTTPS (if web server)
ufw allow 443/tcp comment 'HTTPS'

# Add application-specific ports
# Example: Application on port 8080
# ufw allow 8080/tcp comment 'Application'

# Example: Database from specific IP
# ufw allow from DATABASE_CLIENT_IP to any port 3306 proto tcp comment 'MySQL from App
    Server'

# Review all rules before enabling
ufw show added

# Compare with listening ports
echo "=== Listening Ports ==="
ss -tulnp | grep LISTEN

echo "=== UFW Rules ==="
ufw status numbered
```

CRITICAL Checklist - SSH Rule:

☐ SSH rule added FIRST

- Command executed: ufw allow 22/tcp
- Result: ☐ Rule added
- Verified: ufw status numbered | grep 22
- SSH rule present: ☐ YES (proceed) ☐ NO (STOP!)

Application Ports Checklist:

☐ HTTP (80/tcp)

- Required: ☐ Yes ☐ No
- Rule added: ☐ Yes ☐ N/A

☐ HTTPS (443/tcp)

- Required: ☐ Yes ☐ No
- Rule added: ☐ Yes ☐ N/A

☐ Custom Port 1

- Port: _____ Service: _____
- Rule added: ☐ Yes ☐ N/A

☐ Custom Port 2

- Port: _____ Service: _____
- Rule added: ☐ Yes ☐ N/A

☐ Custom Port 3

- Port: _____ Service: _____
- Rule added: ☐ Yes ☐ N/A

☐ Monitoring agent

- Port: _____
- Rule added: ☐ Yes ☐ N/A

☐ All listening ports have firewall rules

- Compared ss -tulnp with ufw rules
- All covered: ☐ Yes ☐ No
- Unexplained ports: _____

3.3.7 Ensure ufw default deny firewall policy

```
# Set default policies
ufw default deny incoming
ufw default allow outgoing
ufw default deny routed

# Verify policies
ufw status verbose | grep "Default:"

# Should show:
# Default: deny (incoming), allow (outgoing), deny (routed)
```

Checklist:

☐ Default incoming: deny

☐ Default outgoing: allow

☐ Default routed: deny

☐ Policies verified: ☐ PASS ☐ FAIL

3.3.8 CRITICAL: Enable UFW and Test (MOST CRITICAL STEP)

YOU ARE ABOUT TO ENABLE THE FIREWALL!
FINAL CHECKS before enabling:

☐ Console/OOB access is WORKING

☐ SSH rule is present: ufw status numbered | grep 22

☐ All required ports have rules

☐ Current SSH session is OPEN

☐ Team is on standby

☐ You know rollback command: ufw disable

IF ANY ITEM NOT CHECKED: DO NOT PROCEED!

```
# FINAL REVIEW before enabling
echo "=== FINAL REVIEW ==="
echo "Current UFW rules:"
ufw show added

echo ""
echo "Listening ports:"
ss -tulnp | grep LISTEN

echo ""
echo "Press ENTER to enable UFW (or Ctrl+C to abort)"
read

# Enable UFW
ufw --force enable

# Verify UFW is active
ufw status verbose

# Check service status
systemctl status ufw

# IMMEDIATELY test - DO NOT CLOSE CURRENT SSH SESSION YET!
```

CRITICAL: Immediate Testing (DO NOT SKIP):

☐ TEST 1: New SSH Connection (CRITICAL)

- ☐ Open NEW terminal window
- ☐ Attempt SSH: ssh user@server
  - Result: ☐ SUCCESS (proceed) ☐ FAIL (ROLLBACK NOW!)
  - **If FAIL: Execute immediately:**

```
# In current (working) SSH session:
ufw disable
# Then investigate and fix
```

☐ TEST 2: Sudo in New SSH Session

- In new SSH session: sudo -l
- Result: ☐ Working ☐ Not Working
- Can become root: ☐ Yes ☐ No

☐ TEST 3: Required Services Accessible

- HTTP/HTTPS: ☐ Working ☐ Not Working ☐ N/A
- Application ports: ☐ Working ☐ Not Working ☐ N/A
- Database: ☐ Working ☐ Not Working ☐ N/A
- Monitoring: ☐ Working ☐ Not Working ☐ N/A

☐ TEST 4: Blocked Ports Actually Blocked

- From external host: nc -zv <server> <blocked_port>
- Result: ☐ Connection refused/timeout (PASS)
- Result: ☐ Connection successful (FAIL - firewall not working!)

☐ **TEST 5: Application Functionality**

- Web application: ☐ Working ☐ Broken
- API endpoints: ☐ Working ☐ Broken
- Background jobs: ☐ Working ☐ Broken
- Database queries: ☐ Working ☐ Broken

☐ **TEST 6: Monitor UFW Logs**

- Command: tail -f /var/log/ufw.log
- Legitimate traffic blocked: ☐ No (PASS) ☐ Yes (add rules)
- Blocked traffic details: _____

IF ANY CRITICAL TEST FAILS

IMMEDIATE ACTION REQUIRED:
In your current (working) SSH session:

```
# Disable UFW immediately
sudo ufw disable

# Verify services restored
systemctl status <affected_service>

# Review UFW rules
ufw status numbered

# Identify missing rule
ss -tulnp | grep <port>

# Add missing rule
ufw allow <port>/<protocol>

# Re-enable UFW
ufw enable

# Re-test ALL critical tests
```

If still failing after 3 attempts:

1. Keep UFW disabled
2. Document the issue
3. Escalate to security team
4. Schedule troubleshooting session
5. Do NOT leave firewall in broken state

Post-Enable Validation:

☐ UFW active

- ufw status
- Status: ☐ active ☐ inactive

☐ UFW service running

- systemctl status ufw

- Status: ☐ active (running)

☐ UFW rules loaded

- ufw status numbered
- Rule count: _____
- SSH rule present: ☐ Yes (rule #_____)

☐ Firewall actually filtering

- Test blocked port from external
- Result: ☐ Blocked (PASS) ☐ Not blocked (FAIL)

☐ All critical tests passed

☐ No legitimate traffic blocked

☐ Applications functioning normally

☐ Can close old SSH session: ☐ Yes (all tests pass)

3.3.9 Final UFW Validation and Persistence Test

```
# Complete verification
echo "=== UFW Status ==="
ufw status verbose
ufw status numbered

echo ""
echo "=== UFW Service ==="
systemctl status ufw
systemctl is-enabled ufw

echo ""
echo "=== UFW Logs (last 20 lines) ==="
tail -20 /var/log/ufw.log

echo ""
echo "=== iptables rules (UFW manages these) ==="
iptables -L -n -v

# Test persistence (requires reboot)
echo ""
echo "Reboot test required to verify UFW persists"
echo "Schedule reboot: sudo reboot"
```

Persistence Test Checklist:

☐ Pre-reboot verification

- UFW active: ☐ Yes
- UFW enabled: ☐ Yes
- All rules present: ☐ Yes

☐ Reboot scheduled

- Reboot time: _____
- Stakeholders notified: ☐ Yes
- Console access ready: ☐ Yes

☐ POST-REBOOT: UFW verification

- System booted: ☐ Yes ☐ Issues
- UFW active after reboot: ufw status
- Result: ☐ active (PASS) ☐ inactive (FAIL)

☐ POST-REBOOT: Rules verification

- ufw status numbered
- All rules present: ☐ Yes ☐ No
- Rule count matches: ☐ Yes ☐ No

☐ POST-REBOOT: Connectivity test

- SSH accessible: ☐ Yes ☐ No
- Applications working: ☐ Yes ☐ No
- No firewall errors: ☐ Yes ☐ No

☐ POST-REBOOT: Service status

- systemctl status ufw
- Status: ☐ active (running)
- Enabled: ☐ enabled

☐ Overall firewall result: ☐ PASS ☐ FAIL

Final Firewall Documentation:

☐ UFW configuration documented

- Rules list: ☐ Saved (ufw status numbered > /docs/ufw_rules.txt)
- Exceptions documented: ☐ Yes
- Change approved: ☐ Yes

☐ Firewall testing complete

- All tests passed: ☐ Yes
- Issues resolved: ☐ Yes ☐ N/A
- Sign-off obtained: ☐ Yes

5.4 Section 4: Logging and Auditing

Audit Logging Importance

Comprehensive audit logging is critical for:

- Security incident detection and response
- Compliance requirements
- Forensic investigations
- Detecting unauthorized access
- Tracking privileged operations

This section implements ALL 14 CIS audit controls (many were missing in original script).

5.4.1 4.1 Configure System Accounting (auditd)

4.1.1 Ensure auditd is installed

```
# Install auditd and plugins
apt install -y auditd audispd-plugins

# Enable and start auditd
systemctl enable auditd
systemctl start auditd

# Verify installation
systemctl status auditd
dpkg -s auditd
auditctl -l
```

Checklist:

☐ auditd installed: ☐ Yes ☐ No

☐ auditd service enabled: ☐ Yes ☐ No

☐ auditd service running: ☐ Yes ☐ No

☐ Result: ☐ PASS ☐ FAIL

4.1.2 Ensure auditd service is enabled and active

```
# Enable and start auditd
systemctl enable auditd
systemctl start auditd

# Verify
systemctl is-enabled auditd
systemctl is-active auditd
systemctl status auditd
```

Checklist:

☐ auditd enabled: ☐ Yes ☐ No

☐ auditd active: ☐ Yes ☐ No

☐ Result: ☐ PASS ☐ FAIL

4.1.3 Ensure events that modify date and time information are collected

```
# Create audit rules for time changes
cat > /etc/audit/rules.d/50-time-change.rules << 'EOF'
# Monitor time changes
-a always,exit -F arch=b64 -S adjtimex,settimeofday -k time-change
-a always,exit -F arch=b32 -S adjtimex,settimeofday,stime -k time-change
-a always,exit -F arch=b64 -S clock_settime -k time-change
-a always,exit -F arch=b32 -S clock_settime -k time-change
-w /etc/localtime -p wa -k time-change
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep time-change
```

Checklist:

☐ Time change rules created

☐ Rules loaded successfully

☐ Rules verified: ☐ PASS ☐ FAIL

4.1.4 Ensure events that modify user/group information are collected

```
# Create audit rules for identity changes
cat > /etc/audit/rules.d/50-identity.rules << 'EOF'
# Monitor user/group changes
-w /etc/group -p wa -k identity
-w /etc/passwd -p wa -k identity
-w /etc/gshadow -p wa -k identity
-w /etc/shadow -p wa -k identity
-w /etc/security/opasswd -p wa -k identity
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep identity
```

Checklist:

☐ Identity rules created

☐ All identity files monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.5 Ensure events that modify system's network environment are collected

```
# Create audit rules for network changes
cat > /etc/audit/rules.d/50-system-locale.rules << 'EOF'
# Monitor network environment changes
-a always,exit -F arch=b64 -S sethostname,setdomainname -k system-locale
```

```
-a always,exit -F arch=b32 -S sethostname,setdomainname -k system-locale
-w /etc/issue -p wa -k system-locale
-w /etc/issue.net -p wa -k system-locale
-w /etc/hosts -p wa -k system-locale
-w /etc/network -p wa -k system-locale
-w /etc/netplan -p wa -k system-locale
-w /etc/systemd/network -p wa -k system-locale
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep system-locale
```

Checklist:

☐ Network environment rules created

☐ All network config files monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.6 Ensure events that modify system's Mandatory Access Controls are collected

```
# Create audit rules for MAC changes
cat > /etc/audit/rules.d/50-MAC-policy.rules << 'EOF'
# Monitor MAC policy changes (AppArmor)
-w /etc/apparmor/ -p wa -k MAC-policy
-w /etc/apparmor.d/ -p wa -k MAC-policy
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep MAC-policy
```

Checklist:

☐ MAC policy rules created

☐ AppArmor directories monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.7 Ensure login and logout events are collected

```
# Create audit rules for login/logout
cat > /etc/audit/rules.d/50-login.rules << 'EOF'
# Monitor login and logout events
-w /var/log/faillog -p wa -k logins
-w /var/log/lastlog -p wa -k logins
-w /var/log/tallylog -p wa -k logins
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep logins
```

Checklist:

☐ Login/logout rules created

☐ Login logs monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.8 Ensure session initiation information is collected

```
# Create audit rules for session information
cat > /etc/audit/rules.d/50-session.rules << 'EOF'
# Monitor session initiation
-w /var/run/utmp -p wa -k session
-w /var/log/wtmp -p wa -k logins
-w /var/log/btmp -p wa -k logins
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep -E "session|logins"
```

Checklist:

☐ Session rules created

☐ utmp/wtmp/btmp monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.9 Ensure discretionary access control permission modification events are collected

```
# Create audit rules for permission modifications
cat > /etc/audit/rules.d/50-perm_mod.rules << 'EOF'
# Monitor permission modifications
-a always,exit -F arch=b64 -S chmod,fchmod,fchmodat -F auid>=1000 -F auid!=unset -k
    perm_mod
-a always,exit -F arch=b32 -S chmod,fchmod,fchmodat -F auid>=1000 -F auid!=unset -k
    perm_mod
-a always,exit -F arch=b64 -S chown,fchown,fchownat,lchown -F auid>=1000 -F auid!=unset
    -k perm_mod
-a always,exit -F arch=b32 -S chown,fchown,fchownat,lchown -F auid>=1000 -F auid!=unset
    -k perm_mod
-a always,exit -F arch=b64 -S setxattr,lsetxattr,fsetxattr,removexattr,lremovexattr,
    fremovexattr -F auid>=1000 -F auid!=unset -k perm_mod
-a always,exit -F arch=b32 -S setxattr,lsetxattr,fsetxattr,removexattr,lremovexattr,
    fremovexattr -F auid>=1000 -F auid!=unset -k perm_mod
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep perm_mod
```

Checklist:

☐ Permission modification rules created

☐ chmod/chown/setxattr monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.10 Ensure unsuccessful file access attempts are collected

```
# Create audit rules for failed file access
cat > /etc/audit/rules.d/50-access.rules << 'EOF'
# Monitor unsuccessful file access attempts
-a always,exit -F arch=b64 -S open,openat,truncate,ftruncate -F exit=-EACCES -F auid
    >=1000 -F auid!=unset -k access
-a always,exit -F arch=b32 -S open,openat,truncate,ftruncate -F exit=-EACCES -F auid
    >=1000 -F auid!=unset -k access
-a always,exit -F arch=b64 -S open,openat,truncate,ftruncate -F exit=-EPERM -F auid
    >=1000 -F auid!=unset -k access
-a always,exit -F arch=b32 -S open,openat,truncate,ftruncate -F exit=-EPERM -F auid
    >=1000 -F auid!=unset -k access
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep access
```

Checklist:

☐ Failed access rules created

☐ EACCES and EPERM monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.11 Ensure use of privileged commands is collected

```
# Find all privileged commands (setuid/setgid)
echo "Finding privileged commands (this may take a few minutes)..."
find / -xdev \( -perm -4000 -o -perm -2000 \) -type f 2>/dev/null > /tmp/
    privileged_commands.txt

# Create audit rules for privileged commands
cat > /etc/audit/rules.d/50-privileged.rules << 'EOF'
# Monitor privileged command execution
EOF

# Add rule for each privileged command
while IFS= read -r cmd; do
    echo "-a always,exit -F path=$cmd -F perm=x -F auid>=1000 -F auid!=unset -k
    privileged" >> /etc/audit/rules.d/50-privileged.rules
done < /tmp/privileged_commands.txt

# Load rules
augenrules --load

# Verify
auditctl -l | grep privileged | wc -l
echo "Privileged commands monitored: $(auditctl -l | grep privileged | wc -l)"
```

Checklist:

☐ Privileged commands found

- Count: _____

☐ Audit rules generated

☐ Rules loaded successfully

☐ Verification: auditctl -l | grep privileged | wc -l

☐ Result: ☐ PASS ☐ FAIL

4.1.12 Ensure successful file system mounts are collected

```
# Create audit rules for mounts
cat > /etc/audit/rules.d/50-mounts.rules << 'EOF'
# Monitor file system mounts
-a always,exit -F arch=b64 -S mount -F auid>=1000 -F auid!=unset -k mounts
-a always,exit -F arch=b32 -S mount -F auid>=1000 -F auid!=unset -k mounts
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep mounts
```

Checklist:

☐ Mount rules created

☐ Result: ☐ PASS ☐ FAIL

4.1.13 Ensure file deletion events by users are collected

```
# Create audit rules for file deletions
cat > /etc/audit/rules.d/50-delete.rules << 'EOF'
# Monitor file deletion events
-a always,exit -F arch=b64 -S unlink,unlinkat,rename,renameat -F auid>=1000 -F auid!=
    unset -k delete
-a always,exit -F arch=b32 -S unlink,unlinkat,rename,renameat -F auid>=1000 -F auid!=
    unset -k delete
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep delete
```

Checklist:

☐ File deletion rules created

☐ unlink/rename monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.14 Ensure changes to system administration scope (sudoers) is collected

```
# Create audit rules for sudoers changes
cat > /etc/audit/rules.d/50-scope.rules << 'EOF'
# Monitor sudoers changes
-w /etc/sudoers -p wa -k scope
-w /etc/sudoers.d/ -p wa -k scope
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep scope
```

Checklist:

☐ Sudoers monitoring rules created

☐ Result: ☐ PASS ☐ FAIL

4.1.15 Ensure system administrator command executions (sudo) are collected

```
# Create audit rules for sudo command execution
cat > /etc/audit/rules.d/50-actions.rules << 'EOF'
# Monitor sudo command execution
-a always,exit -F arch=b64 -C euid!=uid -F euid=0 -F auid>=1000 -F auid!=unset -S execve
     -k actions
-a always,exit -F arch=b32 -C euid!=uid -F euid=0 -F auid>=1000 -F auid!=unset -S execve
     -k actions
EOF

# Load rules
augenrules --load

# Verify
auditctl -l | grep actions
```

Checklist:

☐ Sudo execution rules created

☐ Result: ☐ PASS ☐ FAIL

4.1.16 Ensure kernel module loading and unloading is collected

```
# Create audit rules for kernel modules
cat > /etc/audit/rules.d/50-modules.rules << 'EOF'
# Monitor kernel module operations
-w /sbin/insmod -p x -k modules
-w /sbin/rmmod -p x -k modules
-w /sbin/modprobe -p x -k modules
-a always,exit -F arch=b64 -S init_module,delete_module -F auid>=1000 -F auid!=unset -k
    modules
-a always,exit -F arch=b32 -S init_module,delete_module -F auid>=1000 -F auid!=unset -k
    modules
EOF

# Load rules
augenrules --load
```

```
# Verify
auditctl -l | grep modules
```

Checklist:

☐ Kernel module rules created

☐ insmod/rmmod/modprobe monitored

☐ Result: ☐ PASS ☐ FAIL

4.1.17 Ensure the audit configuration is immutable

WARNING: Making audit configuration immutable!
After this step:

- Audit rules CANNOT be changed without reboot
- auditctl commands will fail
- Only way to modify: reboot system

Ensure all rules are correct before proceeding!
Recommended: Monitor for 24-48 hours before making immutable.

```
# OPTION 1: Test rules first (RECOMMENDED)
# Monitor for 24-48 hours, review logs, ensure no issues
# Then proceed with immutable

# OPTION 2: Make immutable now (only if confident)
# Create immutable rule (MUST be last rule)
cat > /etc/audit/rules.d/99-finalize.rules << 'EOF'
# Make audit configuration immutable
# Reboot required to change audit rules after this
-e 2
EOF

# Load rules
augenrules --load

# Verify (-e 2 should be last rule)
auditctl -l | tail -5

# Test that rules cannot be modified
auditctl -D 2>&1  # Should fail with "permission denied"
```

Checklist:

☐ Decision on immutable

- ☐ Monitor first (24-48 hours) - RECOMMENDED
- ☐ Make immutable now (only if fully tested)

☐ If making immutable:

- Immutable rule created: ☐ Yes
- Rule loaded: ☐ Yes

- Verified (-e 2 present): ☐ Yes
- Tested cannot modify: ☐ Yes
- Team informed about reboot requirement: ☐ Yes

☐ Result: ☐ PASS ☐ DEFERRED

Summary: Verify all audit rules

```
# Comprehensive audit rule verification
echo "=== Audit Rules Summary ==="
echo "Total rules: $(auditctl -l | wc -l)"
echo ""

echo "Time change rules:"
auditctl -l | grep time-change | wc -l

echo "Identity rules:"
auditctl -l | grep identity | wc -l

echo "System locale rules:"
auditctl -l | grep system-locale | wc -l

echo "MAC policy rules:"
auditctl -l | grep MAC-policy | wc -l

echo "Login/session rules:"
auditctl -l | grep -E "logins|session" | wc -l

echo "Permission modification rules:"
auditctl -l | grep perm_mod | wc -l

echo "Access rules:"
auditctl -l | grep access | wc -l

echo "Privileged command rules:"
auditctl -l | grep privileged | wc -l

echo "Mount rules:"
auditctl -l | grep mounts | wc -l

echo "Delete rules:"
auditctl -l | grep delete | wc -l

echo "Scope rules:"
auditctl -l | grep scope | wc -l

echo "Actions rules:"
auditctl -l | grep actions | wc -l

echo "Module rules:"
auditctl -l | grep modules | wc -l

# Check audit log
echo ""
echo "=== Recent Audit Events ==="
ausearch -ts recent | tail -20

# Check audit log file size
echo ""
echo "=== Audit Log Status ==="
```

```
ls -lh /var/log/audit/audit.log
```

Complete Audit Summary Checklist:

☐ Time change monitoring: ☐ PASS ☐ FAIL

☐ User/group monitoring: ☐ PASS ☐ FAIL

☐ Network environment monitoring: ☐ PASS ☐ FAIL

☐ MAC policy monitoring: ☐ PASS ☐ FAIL

☐ Login/logout monitoring: ☐ PASS ☐ FAIL

☐ Session monitoring: ☐ PASS ☐ FAIL

☐ Permission modification monitoring: ☐ PASS ☐ FAIL

☐ Failed access monitoring: ☐ PASS ☐ FAIL

☐ Privileged command monitoring: ☐ PASS ☐ FAIL

☐ Mount monitoring: ☐ PASS ☐ FAIL

☐ File deletion monitoring: ☐ PASS ☐ FAIL

☐ Sudoers monitoring: ☐ PASS ☐ FAIL

☐ Sudo execution monitoring: ☐ PASS ☐ FAIL

☐ Kernel module monitoring: ☐ PASS ☐ FAIL

☐ Configuration immutable: ☐ PASS ☐ DEFERRED

☐ Total audit rules: _____

☐ Audit logs functioning: ☐ Yes ☐ No

☐ Overall Audit Result: ☐ PASS ☐ FAIL

5.4.2 4.2 Configure Logging

4.2.1 Ensure rsyslog is installed and running

```
# Install rsyslog
apt install -y rsyslog

# Enable and start
systemctl enable rsyslog
systemctl start rsyslog

# Verify
systemctl status rsyslog
```

Checklist:

☐ rsyslog installed and running: ☐ PASS ☐ FAIL

4.2.2 Ensure journald is configured to send logs to rsyslog

```
# Configure journald to forward to syslog
cat > /etc/systemd/journald.conf.d/99-cis.conf << 'EOF'
[Journal]
Storage=persistent
Compress=yes
ForwardToSyslog=yes
EOF

# Restart journald
systemctl restart systemd-journald

# Verify
journalctl --verify
grep "ForwardToSyslog" /etc/systemd/journald.conf.d/99-cis.conf
```

Checklist:

☐ journald configured: ☐ PASS ☐ FAIL

4.2.3 Ensure permissions on all logfiles are configured

```
# Set permissions on log files
find /var/log -type f -exec chmod g-wx,o-rwx {} \;

# Verify
ls -la /var/log/
```

Checklist:

☐ Log file permissions restricted: ☐ PASS ☐ FAIL

5.5 Section 5: Access, Authentication and Authorization

THIS SECTION HAS HIGHEST LOCKOUT RISK!
SSH and PAM changes can completely lock you out!
ABSOLUTELY MANDATORY:

- ☐ Console/OOB access WORKING
- ☐ NEVER close current SSH session until new one verified
- ☐ Test EVERY change immediately
- ☐ Know rollback commands
- ☐ Have root password set (not locked)

DO NOT PROCEED WITHOUT CONSOLE ACCESS!

5.5.1 5.1 Configure time-based job schedulers

5.1.1 Ensure cron daemon is enabled and running

```
# Enable and start cron
systemctl enable cron
systemctl start cron

# Verify
systemctl status cron
```

Checklist:

☐ cron enabled and running: ☐ PASS ☐ FAIL

5.1.2 Ensure permissions on /etc/crontab are configured

```
# Set permissions on crontab
chown root:root /etc/crontab
chmod og-rwx /etc/crontab

# Verify
stat /etc/crontab
```

Checklist:

☐ /etc/crontab permissions: 600, root:root

☐ Result: ☐ PASS ☐ FAIL

5.1.3-5.1.7 Ensure permissions on cron directories

```
# Set permissions on cron directories
chown root:root /etc/cron.hourly /etc/cron.daily /etc/cron.weekly /etc/cron.monthly /etc
    /cron.d
chmod og-rwx /etc/cron.hourly /etc/cron.daily /etc/cron.weekly /etc/cron.monthly /etc/
    cron.d

# Verify
ls -ld /etc/cron.*
```

Checklist:

☐ All cron directories: 700, root:root

☐ Result: ☐ PASS ☐ FAIL

5.1.8 Ensure cron is restricted to authorized users

```
# Create /etc/cron.allow
touch /etc/cron.allow
chmod 640 /etc/cron.allow
chown root:root /etc/cron.allow

# Remove /etc/cron.deny if exists
rm -f /etc/cron.deny

# Verify
ls -l /etc/cron.allow
ls -l /etc/cron.deny 2>&1  # Should not exist
```

Checklist:

☐ /etc/cron.allow created: 640, root:root

☐ /etc/cron.deny removed

☐ Result: ☐ PASS ☐ FAIL

5.1.9 Ensure at is restricted to authorized users

```
# Create /etc/at.allow
touch /etc/at.allow
chmod 640 /etc/at.allow
chown root:root /etc/at.allow

# Remove /etc/at.deny if exists
rm -f /etc/at.deny

# Verify
ls -l /etc/at.allow
```

Checklist:

☐ /etc/at.allow created: 640, root:root

☐ /etc/at.deny removed

☐ Result: ☐ PASS ☐ FAIL

5.5.2 5.2 SSH Server Configuration

SSH CONFIGURATION IS EXTREMELY DANGEROUS!
BEFORE STARTING SSH CONFIGURATION:

- ☐ Backup current SSH config: cp /etc/ssh/sshd_config /etc/ssh/sshd_config.cis-backup
- ☐ Console/OOB access is WORKING and TESTED
- ☐ Current SSH session is OPEN
- ☐ Root password is SET (not locked)
- ☐ Know how to restore: cp /etc/ssh/sshd_config.cis-backup /etc/ssh/sshd_config
- ☐ Team is INFORMED and ON STANDBY

TESTING PROTOCOL:

1. Test SSH config: `sshd -t`
2. Reload SSH: `systemctl reload sshd`
3. Open NEW terminal
4. Test SSH in new terminal
5. Only if successful, close old session

NEVER close your current SSH session until new connection verified!

Pre-SSH Configuration Backup

```
# Backup SSH configuration
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.cis-backup
cp -r /etc/ssh/sshd_config.d/ /etc/ssh/sshd_config.d.cis-backup/ 2>/dev/null

# Document current configuration
sshd -T > /tmp/sshd_config_before_cis.txt

# Verify backup
ls -l /etc/ssh/sshd_config*
```

Pre-SSH Checklist:

☐ SSH config backed up: ☐ Yes ☐ No

☐ Backup verified: ☐ Yes ☐ No

☐ Current config documented: ☐ Yes ☐ No

☐ Console access ready: ☐ Yes ☐ No

☐ If ANY NO: STOP! Do not proceed!

5.2.1 Ensure permissions on SSH private host key files are configured

```
# Find and set permissions on SSH private keys
find /etc/ssh -name 'ssh_host_*_key' -type f -exec chmod 0600 {} \;
find /etc/ssh -name 'ssh_host_*_key' -type f -exec chown root:root {} \;
```

```
# Verify
find /etc/ssh -name 'ssh_host_*_key' -type f -exec stat {} \;
```

Checklist:

☐ Private keys: 0600, root:root

☐ Result: ☐ PASS ☐ FAIL

5.2.2 Ensure permissions on SSH public host key files are configured

```
# Find and set permissions on SSH public keys
find /etc/ssh -name 'ssh_host_*_key.pub' -type f -exec chmod 0644 {} \;
find /etc/ssh -name 'ssh_host_*_key.pub' -type f -exec chown root:root {} \;

# Verify
find /etc/ssh -name 'ssh_host_*_key.pub' -type f -exec stat {} \;
```

Checklist:

☐ Public keys: 0644, root:root

☐ Result: ☐ PASS ☐ FAIL

5.2.3 Ensure permissions on /etc/ssh/sshd_config are configured

```
# Set permissions on sshd_config
chown root:root /etc/ssh/sshd_config
chmod og-rwx /etc/ssh/sshd_config

# Verify
stat /etc/ssh/sshd_config
```

Checklist:

☐ /etc/ssh/sshd_config: 0600, root:root

☐ Result: ☐ PASS ☐ FAIL

5.2.4 Ensure SSH access is limited

```
# Create SSH access control configuration
cat > /etc/ssh/sshd_config.d/01-access.conf << 'EOF'
# Limit SSH access to authorized users/groups
AllowUsers deployuser adminuser
# Or use AllowGroups
# AllowGroups sshusers administrators

# Deny specific users if needed
# DenyUsers baduser

# Deny specific groups if needed
# DenyGroups noremote
EOF

# Test configuration
sshd -t

# If test passes, reload SSH
```

```
systemctl reload sshd

# Verify
sshd -T | grep -E "^(allow|deny)(users|groups)"
```

Checklist:

☐ SSH access limited

- Method: ☐ AllowUsers ☐ AllowGroups
- Users/groups: _____

☐ Configuration tested: ☐ PASS ☐ FAIL

☐ SSH reloaded: ☐ Yes

5.2.5 Ensure SSH LogLevel is appropriate

```
# Set SSH LogLevel
cat > /etc/ssh/sshd_config.d/02-logging.conf << 'EOF'
LogLevel VERBOSE
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep loglevel

# Test logging
tail -f /var/log/auth.log &
# SSH login and check logs
```

Checklist:

☐ LogLevel set to VERBOSE or INFO

☐ Logging verified in /var/log/auth.log

☐ Result: ☐ PASS ☐ FAIL

5.2.6 Ensure SSH X11 forwarding is disabled

```
# Disable X11 forwarding
cat > /etc/ssh/sshd_config.d/03-x11.conf << 'EOF'
X11Forwarding no
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep x11forwarding
```

Checklist:

☐ X11Forwarding disabled

☐ Result: ☐ PASS ☐ FAIL

5.2.7 Ensure SSH MaxAuthTries is set to 4 or less

```
# Set MaxAuthTries
cat > /etc/ssh/sshd_config.d/04-maxauth.conf << 'EOF'
MaxAuthTries 4
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep maxauthtries
```

Checklist:

☐ MaxAuthTries ≤ 4

☐ Result: ☐ PASS ☐ FAIL

5.2.8 Ensure SSH IgnoreRhosts is enabled

```
# Enable IgnoreRhosts
cat > /etc/ssh/sshd_config.d/05-rhosts.conf << 'EOF'
IgnoreRhosts yes
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep ignorerhosts
```

Checklist:

☐ IgnoreRhosts enabled

☐ Result: ☐ PASS ☐ FAIL

5.2.9 Ensure SSH HostbasedAuthentication is disabled

```
# Disable HostbasedAuthentication
cat > /etc/ssh/sshd_config.d/06-hostbased.conf << 'EOF'
HostbasedAuthentication no
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep hostbasedauthentication
```

Checklist:

☐ HostbasedAuthentication disabled

☐ Result: ☐ PASS ☐ FAIL

5.2.10 Ensure SSH root login is disabled

```
# Disable root login
cat > /etc/ssh/sshd_config.d/07-root.conf << 'EOF'
PermitRootLogin no
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep permitrootlogin

# Test (should fail)
ssh root@localhost  # Should be denied
```

Checklist:

☐ PermitRootLogin no

☐ Root login test fails: ☐ Yes (PASS)

☐ Result: ☐ PASS ☐ FAIL

5.2.11 Ensure SSH PermitEmptyPasswords is disabled

```
# Disable empty passwords
cat > /etc/ssh/sshd_config.d/08-emptypass.conf << 'EOF'
PermitEmptyPasswords no
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep permitemptypasswords
```

Checklist:

☐ PermitEmptyPasswords disabled

☐ Result: ☐ PASS ☐ FAIL

5.2.12 Ensure SSH PermitUserEnvironment is disabled

```
# Disable PermitUserEnvironment
cat > /etc/ssh/sshd_config.d/09-userenv.conf << 'EOF'
PermitUserEnvironment no
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep permituserenvironment
```

Checklist:

☐ PermitUserEnvironment disabled

☐ Result: ☐ PASS ☐ FAIL

5.2.13 Ensure only strong Ciphers are used

```
# Configure strong ciphers only
cat > /etc/ssh/sshd_config.d/10-ciphers.conf << 'EOF'
# Use strong ciphers only
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,
    aes256-ctr,aes192-ctr,aes128-ctr
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep ciphers
```

Checklist:

☐ Strong ciphers configured

☐ Weak ciphers removed (3des-cbc, aes128-cbc, etc.)

☐ Result: ☐ PASS ☐ FAIL

5.2.14 Ensure only strong MAC algorithms are used

```
# Configure strong MACs only
cat > /etc/ssh/sshd_config.d/11-macs.conf << 'EOF'
# Use strong MAC algorithms only
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512,hmac-sha2
    -256
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep macs
```

Checklist:

☐ Strong MACs configured

☐ Weak MACs removed (hmac-md5, hmac-sha1, etc.)

☐ Result: ☐ PASS ☐ FAIL

5.2.15 Ensure only strong Key Exchange algorithms are used

```
# Configure strong KexAlgorithms only
cat > /etc/ssh/sshd_config.d/12-kex.conf << 'EOF'
# Use strong Key Exchange algorithms only
KexAlgorithms curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-
    sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group-exchange-sha256
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep kexalgorithms
```

Checklist:

☐ Strong Kex algorithms configured

☐ Weak algorithms removed (diffie-hellman-group14-sha1, etc.)

☐ Result: ☐ PASS ☐ FAIL

5.2.16 Ensure SSH Idle Timeout Interval is configured

```
# Configure idle timeout
cat > /etc/ssh/sshd_config.d/13-timeout.conf << 'EOF'
ClientAliveInterval 300
ClientAliveCountMax 3
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep clientalive
```

Checklist:

☐ ClientAliveInterval 300

☐ ClientAliveCountMax 3

☐ Total timeout: 15 minutes (300 × 3)

☐ Result: ☐ PASS ☐ FAIL

5.2.17 Ensure SSH LoginGraceTime is set to one minute or less

```
# Set LoginGraceTime
cat > /etc/ssh/sshd_config.d/14-logingrace.conf << 'EOF'
LoginGraceTime 60
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep logingracetime
```

Checklist:

☐ LoginGraceTime $\leq$ 60

☐ Result: ☐ PASS ☐ FAIL

5.2.18 Ensure SSH warning banner is configured

```
# Configure banner
cat > /etc/ssh/sshd_config.d/15-banner.conf << 'EOF'
Banner /etc/issue.net
EOF

# Ensure banner file exists
test -f /etc/issue.net || cat > /etc/issue.net << 'EOF'
################################################################################
```

```
#                          AUTHORIZED ACCESS ONLY                                #
################################################################################
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep banner

# Test - banner should appear
ssh localhost  # Banner should display
```

Checklist:

☐ Banner configured

☐ Banner file exists

☐ Banner displays on connection

☐ Result: ☐ PASS ☐ FAIL

5.2.19 Ensure SSH PAM is enabled

```
# Enable PAM
cat > /etc/ssh/sshd_config.d/16-pam.conf << 'EOF'
UsePAM yes
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep usepam
```

Checklist:

☐ UsePAM enabled

☐ Result: ☐ PASS ☐ FAIL

5.2.20 Ensure SSH AllowTcpForwarding is disabled

```
# Disable TCP forwarding (if not needed)
cat > /etc/ssh/sshd_config.d/17-tcpforward.conf << 'EOF'
AllowTcpForwarding no
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep allowtcpforwarding
```

Checklist:

☐ AllowTcpForwarding disabled

☐ TCP forwarding needed: ☐ No ☐ Yes (exception)

☐ Result: ☐ PASS ☐ FAIL ☐ EXCEPTION

5.2.21 Ensure SSH MaxStartups is configured

```
# Configure MaxStartups
cat > /etc/ssh/sshd_config.d/18-maxstartups.conf << 'EOF'
MaxStartups 10:30:60
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep maxstartups
```

Checklist:

☐ MaxStartups configured

☐ Result: ☐ PASS ☐ FAIL

5.2.22 Ensure SSH MaxSessions is limited

```
# Set MaxSessions
cat > /etc/ssh/sshd_config.d/19-maxsessions.conf << 'EOF'
MaxSessions 10
EOF

# Test and reload
sshd -t && systemctl reload sshd

# Verify
sshd -T | grep maxsessions
```

Checklist:

☐ MaxSessions $\leq$ 10

☐ Result: ☐ PASS ☐ FAIL

CRITICAL: Final SSH Testing (MANDATORY)

DO NOT SKIP THIS TESTING!
You have made 22 SSH configuration changes. ONE mistake = lockout!
Test EVERY aspect before closing current session!

```
# Final comprehensive SSH test
echo "=== SSH Configuration Summary ==="
sshd -T | grep -E "(permitrootlogin|passwordauthentication|pubkeyauthentication|
    x11forwarding|maxauthtries|clientalive|logingracetime|ciphers|macs|kexalgorithms|
    usepam|allowtcpforwarding|maxstartups|maxsessions|loglevel|ignorerhosts|
    hostbasedauthentication|permitemptypasswords|permituserenvironment|banner|allowusers
    |allowgroups)"

echo ""
echo "=== Final SSH Configuration Test ==="
sshd -t

if [ $? -eq 0 ]; then
```

```
    echo "SSH configuration syntax: PASS"
else
    echo "SSH configuration syntax: FAIL - DO NOT PROCEED!"
    exit 1
fi

echo ""
echo "=== SSH Service Status ==="
systemctl status sshd

echo ""
echo "=== SSH Listening ==="
ss -tulnp | grep sshd
```

CRITICAL: Final SSH Testing Checklist:

☐ Configuration syntax valid

- sshd -t
- Result: ☐ PASS ☐ FAIL (STOP if FAIL!)

☐ **TEST 1: New SSH Connection (CRITICAL)**

- ☐ Open NEW terminal
- ☐ SSH with regular user: ssh user@server
- Result: ☐ SUCCESS ☐ FAIL
- **If FAIL: ROLLBACK NOW!**

☐ **TEST 2: Sudo in New Session**

- In new session: sudo -l
- Result: ☐ Working ☐ Not Working

☐ **TEST 3: Root Login Blocked**

- Try: ssh root@server
- Result: ☐ Denied (PASS) ☐ Allowed (FAIL)

☐ **TEST 4: Key-based Authentication**

- SSH with key: ☐ Working ☐ Not Working

☐ **TEST 5: Banner Display**

- Banner appears: ☐ Yes ☐ No

☐ **TEST 6: Crypto Algorithms**

- SSH from various clients: ☐ All work ☐ Some fail
- Incompatible clients: _____

☐ **TEST 7: Multiple Connections**

- Open 3 SSH sessions simultaneously
- Result: ☐ All work ☐ Issues

☐ **TEST 8: Idle Timeout**

- Leave session idle 15+ minutes

- Disconnects: ☐ Yes (PASS) ☐ No

☐ All SSH tests passed

☐ Can close old session: ☐ Yes (only if all tests pass)

☐ Overall SSH Result: ☐ PASS ☐ FAIL

IF SSH TESTS FAIL

IMMEDIATE ROLLBACK:
In current (working) SSH session:

```
# Restore backup
cp /etc/ssh/sshd_config.cis-backup /etc/ssh/sshd_config
rm -rf /etc/ssh/sshd_config.d/
cp -r /etc/ssh/sshd_config.d.cis-backup/ /etc/ssh/sshd_config.d/ 2>/dev/null

# Restart SSH
systemctl restart sshd

# Verify
sshd -T | head -20

# Re-test
ssh user@localhost
```

Then:

1. Identify which setting caused the issue
2. Research the problem
3. Fix the specific setting
4. Re-apply incrementally
5. Test after EACH change

5.5.3 5.3 Configure PAM

5.3.1 Ensure password creation requirements are configured

```
# Install pwquality
apt install -y libpam-pwquality

# Configure password requirements
cat > /etc/security/pwquality.conf << 'EOF'
# Password quality requirements
minlen = 14
dcredit = -1
ucredit = -1
ocredit = -1
lcredit = -1
minclass = 4
maxrepeat = 3
maxsequence = 3
gecoscheck = 1
dictcheck = 1
usercheck = 1
enforcing = 1
```

```
EOF

# Verify
grep -E '^[^#]' /etc/security/pwquality.conf
```

Checklist:

☐ pwquality installed

☐ Minimum length: 14

☐ Complexity requirements configured

☐ Result: ☐ PASS ☐ FAIL

5.3.2 Ensure lockout for failed password attempts is configured

```
# Configure faillock
cat > /etc/security/faillock.conf << 'EOF'
# Account lockout policy
deny = 5
fail_interval = 900
unlock_time = 600
EOF

# Update PAM configuration for common-auth
cat > /etc/pam.d/common-auth << 'EOF'
# Account lockout
auth required pam_faillock.so preauth
auth [success=1 default=ignore] pam_unix.so nullok
auth [default=die] pam_faillock.so authfail
auth sufficient pam_faillock.so authsucc
EOF

# Verify
grep faillock /etc/pam.d/common-auth
```

Checklist:

☐ faillock configured

☐ Lockout after 5 attempts

☐ Auto-unlock after 10 minutes

☐ Result: ☐ PASS ☐ FAIL

5.3.3 Ensure password reuse is limited

```
# Configure password history in common-password
sed -i '/pam_unix.so/s/$/ remember=5/' /etc/pam.d/common-password

# Verify
grep remember /etc/pam.d/common-password
```

Checklist:

☐ Password history configured (5 passwords)

☐ Result: ☐ PASS ☐ FAIL

5.3.4 Ensure password hashing algorithm is SHA-512

```
# Verify SHA-512 is used
grep -E "^password.*pam_unix.so.*sha512" /etc/pam.d/common-password

# If not present, add
sed -i '/pam_unix.so/s/$/ sha512/' /etc/pam.d/common-password

# Verify
grep sha512 /etc/pam.d/common-password
```

Checklist:

☐ SHA-512 configured

☐ Result: ☐ PASS ☐ FAIL

5.5.4 5.4 User Accounts and Environment

5.4.1 Set Shadow Password Suite Parameters

```
# Configure password aging in login.defs
sed -i 's/^PASS_MAX_DAYS.*/PASS_MAX_DAYS   90/' /etc/login.defs
sed -i 's/^PASS_MIN_DAYS.*/PASS_MIN_DAYS   1/' /etc/login.defs
sed -i 's/^PASS_WARN_AGE.*/PASS_WARN_AGE   7/' /etc/login.defs

# Apply to existing users
for user in $(awk -F: '$3 >= 1000 {print $1}' /etc/passwd); do
    chage -M 90 -m 1 -W 7 "$user"
done

# Verify
grep -E '^PASS_(MAX|MIN|WARN)' /etc/login.defs
```

Checklist:

☐ PASS_MAX_DAYS: 90

☐ PASS_MIN_DAYS: 1

☐ PASS_WARN_AGE: 7

☐ Applied to existing users

☐ Result: ☐ PASS ☐ FAIL

5.4.2 Ensure system accounts are secured

```
# Lock system accounts and set nologin shell
for user in $(awk -F: '($3 < 1000) {print $1}' /etc/passwd | grep -v "^root$"); do
    if [ "$user" != "sync" ] && [ "$user" != "shutdown" ] && [ "$user" != "halt" ]; then
        usermod -L "$user" 2>/dev/null
        usermod -s /usr/sbin/nologin "$user" 2>/dev/null
    fi
done

# Verify
awk -F: '($3 < 1000 && $1 != "root") {print $1":"$7}' /etc/passwd
```

Checklist:

☐ System accounts locked

☐ Shell set to /usr/sbin/nologin

☐ Result: ☐ PASS ☐ FAIL

5.4.3 Ensure default group for root is GID 0

```
# Check root's GID
id root | grep "gid=0"

# Verify in passwd
grep "^root:" /etc/passwd | cut -d: -f4

# Should be 0
```

Checklist:

☐ Root GID is 0

☐ Result: ☐ PASS ☐ FAIL

5.4.4 Ensure default user umask is configured

```
# Set umask to 027
sed -i 's/umask 022/umask 027/g' /etc/profile
sed -i 's/umask 022/umask 027/g' /etc/bash.bashrc

# Set in login.defs
grep -q "^UMASK" /etc/login.defs && \
sed -i 's/^UMASK.*/UMASK           027/' /etc/login.defs || \
echo "UMASK           027" >> /etc/login.defs

# Verify
grep umask /etc/profile /etc/bash.bashrc
grep UMASK /etc/login.defs
```

Checklist:

☐ umask 027 configured

☐ Result: ☐ PASS ☐ FAIL

5.4.5 Ensure default user shell timeout is configured

```
# Configure shell timeout
cat > /etc/profile.d/timeout.sh << 'EOF'
# Shell timeout - 15 minutes
TMOUT=900
readonly TMOUT
export TMOUT
EOF

chmod 644 /etc/profile.d/timeout.sh

# Verify
grep TMOUT /etc/profile.d/timeout.sh
```

Checklist:

☐ TMOUT=900 configured

☐ TMOUT set readonly

☐ Result: ☐ PASS ☐ FAIL

5.5.5 5.5 Ensure root login is restricted to system console

```
# Restrict root login to console only
cat > /etc/securetty << 'EOF'
console
EOF

# Verify
cat /etc/securetty
```

Checklist:

☐ Root console login restricted

☐ Result: ☐ PASS ☐ FAIL

5.5.6 5.6 Ensure access to su command is restricted

```
# Configure PAM to restrict su
grep -E '^\s*auth\s+required\s+pam_wheel.so' /etc/pam.d/su || \
echo "auth required pam_wheel.so use_uid group=sudo" >> /etc/pam.d/su

# Verify
grep pam_wheel.so /etc/pam.d/su
```

Checklist:

☐ pam_wheel configured

☐ su restricted to sudo group

☐ Result: ☐ PASS ☐ FAIL

5.6 Section 6: System Maintenance

5.6.1 6.1 System File Permissions

6.1.1 Audit system file permissions

```
# Set permissions on critical files
chmod 644 /etc/passwd
chmod 000 /etc/shadow
chmod 644 /etc/group
chmod 000 /etc/gshadow

chown root:root /etc/passwd /etc/shadow /etc/group /etc/gshadow

# Verify
stat /etc/passwd /etc/shadow /etc/group /etc/gshadow
```

Checklist:

☐ /etc/passwd: 644, root:root

☐ /etc/shadow: 000, root:root

☐ /etc/group: 644, root:root

☐ /etc/gshadow: 000, root:root

☐ Result: ☐ PASS ☐ FAIL

6.1.2 Ensure no world writable files exist

```
# Find world-writable files
df --local -P | awk '{if (NR!=1) print $6}' | \
xargs -I '{}' find '{}' -xdev -type f -perm -0002 2>/dev/null

# Save results
df --local -P | awk '{if (NR!=1) print $6}' | \
xargs -I '{}' find '{}' -xdev -type f -perm -0002 2>/dev/null > /tmp/world-writable-
    files.txt

# Count
wc -l /tmp/world-writable-files.txt
```

Checklist:

☐ World-writable files found: _____

☐ Each file reviewed: ☐ Yes

☐ Unnecessary write permissions removed: ☐ Yes

☐ Result: ☐ PASS ☐ FAIL

6.1.3 Ensure no unowned files or directories exist

```
# Find unowned files
df --local -P | awk '{if (NR!=1) print $6}' | \
xargs -I '{}' find '{}' -xdev -nouser 2>/dev/null

# Save results
df --local -P | awk '{if (NR!=1) print $6}' | \
```

```
xargs -I '{}' find '{}' -xdev -nouser 2>/dev/null > /tmp/unowned-files.txt

wc -l /tmp/unowned-files.txt
```

Checklist:

☐ Unowned files found: _____

☐ Each file reviewed and ownership assigned

☐ Result: ☐ PASS ☐ FAIL

6.1.4 Ensure no ungrouped files or directories exist

```
# Find ungrouped files
df --local -P | awk '{if (NR!=1) print $6}' | \
xargs -I '{}' find '{}' -xdev -nogroup 2>/dev/null

# Save results
df --local -P | awk '{if (NR!=1) print $6}' | \
xargs -I '{}' find '{}' -xdev -nogroup 2>/dev/null > /tmp/ungrouped-files.txt

wc -l /tmp/ungrouped-files.txt
```

Checklist:

☐ Ungrouped files found: _____

☐ Each file reviewed and group assigned

☐ Result: ☐ PASS ☐ FAIL

5.6.2 6.2 User and Group Settings

6.2.1 Ensure accounts in /etc/passwd use shadowed passwords

```
# Check for non-shadowed passwords
awk -F: '($2 != "x" ) { print $1 }' /etc/passwd

# Should return nothing
```

Checklist:

☐ All accounts use shadowed passwords

☐ Result: ☐ PASS ☐ FAIL

6.2.2 Ensure /etc/shadow password fields are not empty

```
# Check for empty passwords
awk -F: '($2 == "" ) { print $1 }' /etc/shadow

# Should return nothing
# If found, lock account: passwd -l <username>
```

Checklist:

☐ No empty password fields

☐ Result: ☐ PASS ☐ FAIL

6.2.3 Ensure all groups in /etc/passwd exist in /etc/group

```
# Check group consistency
for group in $(cut -d: -f4 /etc/passwd | sort -u); do
    grep -q "^.*:.*:$group:" /etc/group || echo "Group $group is referenced but does not
     exist"
done
```

Checklist:

☐ All groups exist

☐ Result: ☐ PASS ☐ FAIL

6.2.4 Ensure no duplicate UIDs exist

```
# Check for duplicate UIDs
cut -d: -f3 /etc/passwd | sort | uniq -d
```

Checklist:

☐ No duplicate UIDs

☐ Result: ☐ PASS ☐ FAIL

6.2.5 Ensure no duplicate GIDs exist

```
# Check for duplicate GIDs
cut -d: -f3 /etc/group | sort | uniq -d
```

Checklist:

☐ No duplicate GIDs

☐ Result: ☐ PASS ☐ FAIL

6.2.6 Ensure no duplicate user names exist

```
# Check for duplicate usernames
cut -d: -f1 /etc/passwd | sort | uniq -d
```

Checklist:

☐ No duplicate usernames

☐ Result: ☐ PASS ☐ FAIL

6.2.7 Ensure no duplicate group names exist

```
# Check for duplicate group names
cut -d: -f1 /etc/group | sort | uniq -d
```

Checklist:

☐ No duplicate group names

☐ Result: ☐ PASS ☐ FAIL

6.2.8 Ensure root PATH Integrity

```
# Check root PATH
echo $PATH | grep -q "::" && echo "Empty directory in PATH"
echo $PATH | grep -q ":$" && echo "Trailing colon in PATH"
echo $PATH | grep -q "^\." && echo "Current directory in PATH"

# Check for writable directories in PATH
for dir in $(echo $PATH | tr ":" " "); do
    if [ -d "$dir" ]; then
        ls -ld "$dir" | awk '$1 ~ /^d....w/ {print "World writable: " $NF}'
    fi
done
```

Checklist:

☐ No empty directories in PATH

☐ No trailing colon

☐ Current directory not in PATH

☐ No world-writable directories

☐ Result: ☐ PASS ☐ FAIL

6.2.9 Ensure all users' home directories exist

```
# Check home directories exist
awk -F: '($3>=1000) {print $1":"$6}' /etc/passwd | while IFS=: read -r user dir; do
    if [ ! -d "$dir" ]; then
        echo "Home directory $dir does not exist for user $user"
    fi
done
```

Checklist:

☐ All home directories exist

☐ Missing directories created

☐ Result: ☐ PASS ☐ FAIL

6.2.10 Ensure users own their home directories

```
# Check home directory ownership
awk -F: '($3>=1000) {print $1":"$6}' /etc/passwd | while IFS=: read -r user dir; do
    if [ -d "$dir" ]; then
        owner=$(stat -L -c "%U" "$dir")
        if [ "$owner" != "$user" ]; then
            echo "Directory $dir is owned by $owner, should be $user"
        fi
    fi
done
```

Checklist:

☐ Home directory ownership correct

☐ Result: ☐ PASS ☐ FAIL

6.2.11 Ensure users' home directory permissions are 750 or more restrictive

```
# Check and fix home directory permissions
for dir in /home/*; do
    if [ -d "$dir" ]; then
        chmod 750 "$dir"
    fi
done

# Verify
ls -ld /home/*
```

Checklist:

☐ Home directory permissions $\leq$ 750

☐ Result: ☐ PASS ☐ FAIL

6.2.12 Ensure users' dot files are not group or world writable

```
# Check and fix dot file permissions
for dir in /home/*; do
    if [ -d "$dir" ]; then
        find "$dir" -name ".*" -type f -exec chmod go-w {} \;
    fi
done

# Verify
find /home/*/.[A-Za-z0-9]* -type f -perm /g+w,o+w 2>/dev/null
```

Checklist:

☐ Dot files not group/world writable

☐ Result: ☐ PASS ☐ FAIL

6.2.13 Ensure no users have .forward files

```
# Check for .forward files
find /home -name .forward 2>/dev/null

# Remove if found
find /home -name .forward -exec rm {} \;
```

Checklist:

☐ No .forward files exist

☐ Result: ☐ PASS ☐ FAIL

6.2.14 Ensure no users have .netrc files

```
# Check for .netrc files
find /home -name .netrc 2>/dev/null

# Remove if found
find /home -name .netrc -exec rm {} \;
```

Checklist:

☐ No .netrc files exist

☐ Result: ☐ PASS ☐ FAIL

6.2.15 Ensure no users have .rhosts files

```
# Check for .rhosts files
find /home -name .rhosts 2>/dev/null

# Remove if found
find /home -name .rhosts -exec rm {} \;
```

Checklist:

☐ No .rhosts files exist

☐ Result: ☐ PASS ☐ FAIL

6 Phase 4: Post-Implementation Validation

6.1 Complete System Validation

Final Validation

This phase validates that all CIS controls are implemented correctly and the system is functioning properly.

6.1.1 System Stability Check

```
# System uptime and load
uptime

# Check failed services
systemctl --failed

# Check errors in journal
journalctl -p err -b

# Check system resources
free -h
df -h

# Check network connectivity
ping -c 4 8.8.8.8
curl -I https://google.com
```

System Stability Checklist:

☐ System uptime: _____

☐ Load average acceptable: ☐ Yes ☐ No

☐ No failed services: ☐ Yes ☐ Some (list below)

☐ No critical errors in journal: ☐ Yes ☐ Some

☐ Memory usage normal: ☐ Yes ☐ High

☐ Disk space adequate: ☐ Yes ☐ Low

☐ Network connectivity working: ☐ Yes ☐ Issues

☐ Overall Stability: ☐ PASS ☐ FAIL

6.1.2 Application Functionality Test

```
# Test web application
curl http://localhost
curl https://localhost

# Test database
# MySQL: mysql -u root -p -e "SELECT 1;"
# PostgreSQL: psql -c "SELECT 1;"

# Test application endpoints
# curl http://localhost:8080/health

# Check background jobs
```

```
crontab -l
systemctl list-timers

# Check monitoring agents
# systemctl status <monitoring-agent>
```

Application Functionality Checklist:

☐ Web application responding: ☐ Yes ☐ No

☐ Database accessible: ☐ Yes ☐ No

☐ Background jobs running: ☐ Yes ☐ No

☐ Monitoring agents active: ☐ Yes ☐ No

☐ All critical functions working: ☐ Yes ☐ No

☐ Overall Functionality: ☐ PASS ☐ FAIL

6.1.3 Security Validation

```
# Verify firewall active
ufw status verbose

# Verify audit logging
auditctl -l | wc -l
tail -20 /var/log/audit/audit.log

# Verify AppArmor active
aa-status

# Check SSH hardening
sshd -T | grep -E "(permitrootlogin|passwordauthentication)"

# Check failed login attempts
tail -50 /var/log/auth.log | grep -i failed
```

Security Validation Checklist:

☐ Firewall active and enforcing: ☐ Yes ☐ No

☐ Audit rules loaded: _____ rules

☐ AppArmor profiles enforcing: ☐ Yes ☐ Partial

☐ SSH hardened: ☐ Yes ☐ No

☐ Failed logins logged: ☐ Yes ☐ No

☐ Overall Security: ☐ PASS ☐ FAIL

6.1.4 Compliance Re-Assessment

```
# Run automated CIS assessment (if available)
# CIS-CAT Pro:
# ./Assessor-CLI.sh -i -rd /reports -b benchmarks/CIS_Ubuntu_Linux_24.04_Benchmark_v1
    .0.0-xccdf.xml

# Or run Lynis
lynis audit system --quick
```

```
# Save report
lynis audit system > /tmp/lynis-post-cis-report.txt
```

Compliance Re-Assessment Checklist:

☐ Automated assessment run: ☐ CIS-CAT ☐ Lynis ☐ Manual

☐ Score/Results: _____

☐ Improvement from baseline: _____%

☐ Critical findings: _____

☐ Overall Compliance: ☐ PASS ☐ FAIL

7 Phase 5: Documentation & Sign-Off

7.1 Implementation Summary

CIS Benchmark Implementation Summary

Metric	Value
Server	_____
Implementation Date	_____
Tester	_____
Total Duration	_____ hours
Controls Tested	
Section 1 - Initial Setup	☐ Complete ☐ Partial ☐ Not Applicable
Section 2 - Services	☐ Complete ☐ Partial ☐ Not Applicable
Section 3 - Network	☐ Complete ☐ Partial ☐ Not Applicable
Section 4 - Logging/Audit	☐ Complete ☐ Partial ☐ Not Applicable
Section 5 - Access Control	☐ Complete ☐ Partial ☐ Not Applicable
Section 6 - System Maintenance	☐ Complete ☐ Partial ☐ Not Applicable
Results	
Total Controls	_____
Passed	_____ (_____%)
Failed	_____ (_____%)
Exceptions	_____ (_____%)
Not Applicable	_____ (_____%)
Compliance Score	_____%

7.2 Exception Documentation

Document all exceptions:

Control	Reason	Compensating Control	Approved By
squashfs (1.1.1.6)	Snap packages required	Monitor snap packages	_____
Wireless (3.1.5)	Laptop/mobile device	WPA3, 802.1X auth	_____
AppArmor profile	Application compatibility	Enhanced logging	_____

7.3 Issues Encountered

Document any issues:

1. Issue: _____
   - Resolution: _____
   - Time to resolve: _____
2. Issue: _____
   - Resolution: _____
   - Time to resolve: _____

3. Issue: _____
   - Resolution: _____
   - Time to resolve: _____

7.4 Lessons Learned

What went well:

- _____
- _____
- _____

What could be improved:

- _____
- _____
- _____

Recommendations for future implementations:

- _____
- _____
- _____

7.5 Final Sign-Off

Final Sign-Off

Role	Approval
Security Engineer	Name: _____ Signature: _____ Date: _____
System Administrator	Name: _____ Signature: _____ Date: _____
Application Owner	Name: _____ Signature: _____ Date: _____
IT Manager	Name: _____ Signature: _____ Date: _____
Security Manager	Name: _____ Signature: _____ Date: _____

7.6 Next Steps

Immediate actions (within 24 hours):

☐ Document configuration in CMDB

☐ Update runbooks with new procedures

☐ Brief operations team on changes

☐ Schedule post-implementation review

Short-term actions (within 1 week):

☐ Monitor logs for issues

☐ Review audit findings

☐ Address any exceptions

☐ Update disaster recovery procedures

Long-term actions (within 1 month):

☐ Schedule quarterly CIS compliance reviews

☐ Plan for other systems

☐ Develop automation scripts

☐ Train additional staff

8 Appendix A: Quick Reference Commands

8.1 Emergency Commands

```
# SSH Lockout Recovery
# Via console:
cp /etc/ssh/sshd_config.cis-backup /etc/ssh/sshd_config
systemctl restart sshd

# Firewall Lockout Recovery
ufw disable

# View Failed Services
systemctl --failed

# Check Recent Errors
journalctl -p err -n 50

# Restore from Backup
cd /backup/cis_pre_YYYYMMDD/
tar -xzf etc_backup.tar.gz -C /

# Emergency Reboot
reboot
```

8.2 Verification Commands

```
# Quick CIS Status Check
echo "=== CIS Implementation Status ==="
echo "Firewall: $(ufw status | grep Status)"
echo "AppArmor: $(aa-status | grep 'loaded' | head -1)"
echo "Auditd: $(systemctl is-active auditd)"
echo "SSH: $(systemctl is-active ssh)"
echo "Failed Services: $(systemctl --failed | grep -c 'loaded')"

# Comprehensive Check
lynis audit system --quick
```

8.3 Monitoring Commands

```
# Monitor Firewall Logs
tail -f /var/log/ufw.log

# Monitor Authentication Logs
tail -f /var/log/auth.log

# Monitor Audit Logs
tail -f /var/log/audit/audit.log

# Monitor System Logs
journalctl -f
```

9 Appendix B: Rollback Procedures

9.1 Complete System Rollback

If complete rollback is needed:

1. **Restore VM Snapshot (Fastest)**

```
# VMware
vim-cmd vmsvc/snapshot.revert <vm_id> <snapshot_id>

# KVM
virsh snapshot-revert <domain> <snapshot_name>
```

2. **Or Restore from Backup**

```
# Restore configuration
cd /backup/cis_pre_*/
tar -xzf etc_backup.tar.gz -C /

# Restart services
systemctl daemon-reload
systemctl restart ssh
systemctl restart networking

# Reboot
reboot
```

9.2 Selective Rollback

```
# Rollback SSH only
cp /etc/ssh/sshd_config.cis-backup /etc/ssh/sshd_config
systemctl restart sshd

# Rollback Firewall only
ufw disable

# Rollback Network Parameters
rm /etc/sysctl.d/60-net*.conf
sysctl --system

# Rollback Audit Rules
auditctl -D
```

10 Appendix C: Troubleshooting Guide

10.1 SSH Issues

Problem: Cannot SSH to server

- Check firewall: `ufw status`
- Check SSH service: `systemctl status ssh`
- Check SSH config: `sshd -t`
- Check logs: `tail -50 /var/log/auth.log`

10.2 Firewall Issues

Problem: Service not accessible

- Check UFW rules: `ufw status numbered`
- Check listening ports: `ss -tulnp`
- Add missing rule: `ufw allow <port>/<protocol>`
- Check logs: `tail -f /var/log/ufw.log`

10.3 AppArmor Issues

Problem: Application fails after AppArmor enforcement

- Check denials: `grep DENIED /var/log/syslog`
- Set to complain mode: `aa-complain /etc/apparmor.d/profile`
- Generate custom profile: `aa-genprof /path/to/app`

10.4 Performance Issues

Problem: System slow after CIS implementation

- Check AIDE: May be running scan (CPU intensive)
- Check audit: `auditctl -l | wc -l` (many rules = overhead)
- Check AppArmor: `aa-status` (all profiles = overhead)
- Monitor: `top`, `iotop`

11 Appendix D: Additional Resources

11.1 Official Documentation

- CIS Benchmarks: https://www.cisecurity.org/cis-benchmarks/
- Ubuntu Security: https://ubuntu.com/security
- AppArmor Wiki: https://gitlab.com/apparmor/apparmor/-/wikis/home
- Auditd Documentation: https://github.com/linux-audit/audit-documentation

11.2 Tools

- CIS-CAT Pro: https://www.cisecurity.org/cybersecurity-tools/cis-cat-pro/
- Lynis: https://cisofy.com/lynis/
- OpenSCAP: https://www.open-scap.org/
- Wazuh: https://wazuh.com/
- Ubuntu Security Guide (USG): https://documentation.ubuntu.com/security/compliance/usg/

11.3 Support Contacts

Contact	Email	Phone
Security Team	_____	_____
Operations	_____	_____
24/7 Support	_____	_____

Conclusion and Recommendations

Executive Summary

This document provides a comprehensive implementation guide for the CIS Ubuntu Linux 24.04 LTS Benchmark v1.0.0. The checklist format ensures systematic coverage of all security controls while maintaining practical applicability in enterprise environments.

Key Achievements

- **100% Coverage** of CIS Ubuntu 24.04 LTS controls
- **Enhanced Safety Procedures** with exception handling for operational requirements
- **Comprehensive Verification** steps for each control implementation
- **Practical Implementation** guidance with real-world considerations
- **Integrated Testing Methodology** for validation of security controls

Security Posture Improvements

Implementation of controls documented in this checklist will result in:

- Reduced attack surface through filesystem hardening
- Enhanced access control via AppArmor enforcement
- Improved system integrity through AIDE monitoring
- Secure boot configuration with password protection
- Comprehensive logging and auditing capabilities
- Network security hardening and service management
- Proper user authentication and authorization controls

Implementation Recommendations

1. **Phased Implementation**: Deploy controls in phases to minimize operational impact
2. **Thorough Testing**: Validate all changes in non-production environments first
3. **Documentation**: Maintain detailed records of all exceptions and modifications
4. **Regular Monitoring**: Establish continuous monitoring and periodic reassessment
5. **Staff Training**: Ensure operational teams understand security controls
6. **Backup Strategy**: Maintain system backups before implementing major changes

Ongoing Maintenance

- **Monthly Review**: Update package repositories and security patches
- **Quarterly Assessment**: Re-run AIDE integrity checks and review audit logs
- **Annual Update**: Review and update this checklist against new CIS releases
- **Continuous Monitoring**: Monitor system logs for security events and anomalies

Compliance Framework

This implementation addresses requirements from multiple compliance frameworks:

- CIS Controls v8
- NIST Cybersecurity Framework
- ISO/IEC 27001
- PCI DSS (where applicable)
- Industry-specific security requirements

Document Control

CIS Ubuntu Linux 24.04 LTS Benchmark
Complete Implementation and Testing Checklist

Document Version: 1.0.0
CIS Benchmark Version: v1.0.0 (08-26-2024)
Implementation Date: December 2, 2025

Prepared by:
Csalab Security Assessment Team

Document Classification: Confidential
Distribution: Authorized Security Personnel Only

This document contains proprietary security implementation guidance.
Unauthorized distribution is strictly prohibited.

DOCUMENT CONCLUSION

"Security is not a product, but a process."

– Anonymous

Implementation Status

Sections Completed:

- ☐ Section 1: Initial Setup
- ☐ Section 2: Services
- ☐ Section 3: Network Configuration
- ☐ Section 4: Logging and Auditing
- ☐ Section 5: Access, Authentication and Authorization
- ☐ Section 6: System Maintenance

Implementation Ready: All controls verified and tested
Next Phase: Production deployment with ongoing monitoring

— END OF DOCUMENT —