Linux Security Commands Cheatsheet

Essential security commands every Linux administrator must master. This comprehensive reference covers the critical tools for hardening systems, detecting intrusions, and responding to security incidents. The commands and configurations below reflect industry best practices from CIS Benchmarks, NIST guidelines, and recommendations from security professionals who emphasize that "configuring systems in compliance eliminates 80-95% of known vulnerabilities." github

---

File permissions form your first line of defense

Proper permission management prevents the most common privilege escalation attacks. The principle of least privilege should guide every permission decision.

chmod, chown, and chgrp essentials

Command	Security Use	Critical Flags
chmod 600 file	Private keys, sensitive configs	Never use 777
chmod 750 directory	Group-accessible directories	-R with caution
chown -h user:group file	Change ownership	-h prevents symlink attacks
chgrp -R group directory	Set group ownership	Verify before recursive ops

SUID/SGID/Sticky bit commands warrant special attention. LinuxConfig SUID files (chmod u+s or 4755) run with owner privileges The CyberSec Guru +2 and represent primary privilege escalation vectors when misconfigured. The sticky bit (chmod +t or 1777) on /tmp prevents users from deleting others' files—The CyberSec Guru +2never remove it.

```bash
# Audit SUID files (CIS requirement)
find / -perm -4000 -type f 2>/dev/null

# Find world-writable files
find / -type f -perm -002 2>/dev/null
```

umask secures new files by default

CIS Benchmark 5.4.4 mandates **umask 027 or more restrictive**. CIS Center for Internet Security This ensures new

files get `640` permissions and directories get `750`.

```bash
# Set in /etc/profile or ~/.bashrc
umask 027
```

ACLs provide granular access control

```bash
# Grant specific user read-write access
setfacl -m u:contractor:rw /srv/project/docs

# Set default ACL for new files in directory
setfacl -dm g:developers:rwx /srv/shared

# View ACLs (files with ACLs show + in ls -l)
getfacl /path/to/file

# Always test before recursive operations
setfacl -R --test -m g:team:rwx /srv/
```

User management requires security-first defaults

Creating secure accounts

```bash
```

```
# Service account (non-interactive, no home directory)
useradd -r -s /sbin/nologin -M -c "Web Server" nginx

# Temporary contractor with expiration
useradd -m -s /bin/bash -e 2025-03-01 -G developers contractor01

# CRITICAL: Always use -aG to append groups (not -G alone)
usermod -aG sudo username    # Correct
usermod -G sudo username     # WRONG - removes all other groups!

# Lock/unlock accounts
usermod -L username          # Lock
passwd -l username           # Alternative lock
usermod -U username          # Unlock
```

Password policy enforcement

bash

```
# Force password change on next login
passwd -e username

# Set password aging (max 90 days, warn 7 days before)
chage -M 90 -W 7 username

# Check password status
passwd -S username
```

sudo configuration best practices

Always use **visudo** to edit sudoers files—it validates syntax and prevents lockouts. The Network Logician

sudoers

```
# /etc/sudoers.d/security-defaults
Defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
Defaults    logfile="/var/log/sudo.log"
Defaults    log_input, log_output
Defaults    passwd_tries=3
Defaults    timestamp_timeout=5
Defaults    requiretty

# Principle of least privilege - specific commands only
%developers localhost=/usr/bin/systemctl restart nginx, /usr/bin/systemctl status nginx
```

Warning about command escalation: Allowing sudo access to editors, interpreters, or `find` enables trivial root shell escapes via GTFOBins techniques.

Log analysis reveals security incidents

journalctl powers modern log investigation

```bash
# Security-critical events by priority (err and above)
journalctl -p err -b

# SSH authentication events with time filter
journalctl -u ssh.service --since "1 hour ago"

# Filter by user ID (root activity)
journalctl _UID=0 --since today

# Real-time monitoring
journalctl -f -u ssh.service

# Export for forensic preservation (JSON for SIEM)
journalctl --since="2025-06-01" -o json > incident_logs.json
```

Critical log files by attack type

Attack Type	Log Files	Key Patterns
Brute force SSH	`/var/log/auth.log` (Debian) or `/var/log/secure` (RHEL)	"Failed password", "Invalid user"

Attack Type	Log Files	Key Patterns
Privilege escalation	auth.log, audit.log	sudo commands, setuid/setgid syscalls
Unauthorized access	`/var/log/audit/audit.log`	Permission denied, file access attempts
Login history	`/var/log/wtmp` (use `last`), `/var/log/btmp` (use `lastb`)	Unusual times, unknown IPs

bash

```
# Count failed SSH attempts by IP (brute force detection)
grep "Failed password" /var/log/auth.log | awk '{print $11}' | sort | uniq -c | sort -rn | head

# Enable persistent journald logging for forensics
mkdir -p /var/log/journal
systemctl restart systemd-journald
```

Process monitoring detects active threats

ps and lsof for security investigation

bash

```
# Full process tree (spot suspicious child processes)
ps auxf

# Show full command lines
ps auxww

# Find processes by user
ps -u www-data

# Network connections with process info
lsof -i -n

# Find what's listening on specific port
lsof -i :22

# Deleted files still open (malware hiding technique)
lsof +L1

# Files opened by suspicious PID
lsof -p <PID>
```

Detecting hidden processes and rootkits

```
bash

# Install and run rootkit hunters
chkrootkit
rkhunter --update && rkhunter --check

# Detect hidden processes
unhide proc
unhide brute

# Compare ps output with /proc (manual check)
ps aux | wc -l
ls -d /proc/[0-9]* | wc -l

# Check for promiscuous interfaces (sniffing)
ip link | grep PROMISC
```

Network commands secure your perimeter

ss replaces netstat for modern systems

```bash
# Listening TCP ports with process info
ss -tlnp

# Established connections
ss -tan state established

# Connections to/from specific port
ss -tan '( dport = :22 or sport = :22 )'

# Find connections from suspicious IP
ss dst 192.168.1.100

# Summary statistics (detect abnormal patterns)
ss -s
```

iptables rules for common security scenarios

```bash
```

```
# Block specific IP
iptables -A INPUT -s 192.168.1.100 -j DROP

# Rate limit SSH (brute force prevention)
iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --name ssh --set
iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --name ssh --update --seconds 60 --hitcount 6 -j DROP

# Log then drop suspicious packets
iptables -N LOGDROP
iptables -A LOGDROP -m limit --limit 5/min -j LOG --log-prefix "FIREWALL-DROP: "
iptables -A LOGDROP -j DROP

# Block port scan signatures (NULL, XMAS)
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP

# Stateful firewall baseline
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT
iptables -P INPUT DROP
```

firewalld (RHEL/CentOS) and ufw (Ubuntu)

```bash
# firewalld: Add service permanently
firewall-cmd --permanent --add-service=https
firewall-cmd --reload

# firewalld: Rich rule to block IP
firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="10.0.0.50" drop'

# ufw: Rate limit SSH (6 connections/30 seconds)
ufw limit ssh

# ufw: Allow from specific subnet
ufw allow from 192.168.1.0/24 to any port 22
```

tcpdump captures suspicious traffic

```bash
```

```
# Detect SYN scans
tcpdump -i eth0 'tcp[tcpflags] & tcp-syn != 0 and tcp[tcpflags] & tcp-ack == 0'

# Large packets (potential exfiltration)
tcpdump -i eth0 'greater 1000'

# DNS queries (detect tunneling—watch for large packets)
tcpdump -i eth0 port 53

# Capture to file for Wireshark analysis
tcpdump -i eth0 -nn -w capture.pcap

# HTTP credentials in plaintext (audit only)
tcpdump port http -l -A | egrep -i 'pass=|user='
```

Security auditing tools provide compliance and detection

auditd monitors system calls at kernel level

bash

```
# Check audit status
auditctl -s

# CIS-recommended audit rules (/etc/audit/audit.rules)
-w /etc/passwd -p wa -k identity
-w /etc/shadow -p wa -k identity
-w /etc/sudoers -p wa -k sudoers_changes
-w /var/log/lastlog -p wa -k logins
-a always,exit -F arch=b64 -S execve -F euid=0 -F auid>=1000 -k privilege_escalation

# Search today's events by key
ausearch -k passwd_changes -i --start today

# Generate summary reports
aureport --login --failed --summary
aureport --auth --summary
```

AIDE detects file integrity changes

bash

```
# Initialize database (on clean system)
aide --init
mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz

# Check for modifications
aide --check

# Update after legitimate changes
aide --update
```

Lynis performs comprehensive security audits

bash

```
# Full system audit
lynis audit system

# Quick scan with cronjob mode
lynis audit system --quick --cronjob

# Audit specific category
lynis audit system --tests-from-category "authentication"

# View specific test details
lynis show details AUTH-9328
```

Lynis hardening index scores 0-100; higher means more hardened. Warnings require immediate attention; suggestions improve overall posture. cisofy

---

SELinux and AppArmor enforce mandatory access control

SELinux essential commands

bash

```
# Check mode
getenforce              # Enforcing, Permissive, or Disabled
sestatus -v             # Detailed status

# Temporary mode change
setenforce 1            # Enforcing
setenforce 0            # Permissive

# Manage file contexts
semanage fcontext -a -t httpd_sys_content_t "/srv/web(/.*)?"
restorecon -Rv /srv/web

# Allow custom port
semanage port -a -t http_port_t -p tcp 8080

# Troubleshoot denials
ausearch -m AVC -ts recent | audit2why
```

AppArmor profile management

```
bash

# Check status
aa-status

# Set profile to enforce mode
aa-enforce /etc/apparmor.d/usr.sbin.nginx

# Set to complain mode (learning/debugging)
aa-complain /etc/apparmor.d/usr.sbin.nginx

# Generate new profile interactively
aa-genprof /usr/sbin/nginx

# Reload profile
apparmor_parser -r /etc/apparmor.d/usr.sbin.nginx
```

Security hardening hardens the attack surface

SSH hardening (/etc/ssh/sshd_config)

```bash
PermitRootLogin no
PasswordAuthentication no      # Key-based only
MaxAuthTries 3
AllowUsers adminuser
ClientAliveInterval 300
ClientAliveCountMax 2
X11Forwarding no
```

Kernel security parameters (/etc/sysctl.conf)

```bash
# Network hardening
net.ipv4.tcp_syncookies = 1            # SYN flood protection
net.ipv4.conf.all.rp_filter = 1        # IP spoofing protection
net.ipv4.conf.all.accept_redirects = 0  # Disable ICMP redirects
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.all.log_martians = 1      # Log suspicious packets

# Kernel hardening
kernel.randomize_va_space = 2          # Enable ASLR
kernel.dmesg_restrict = 1              # Restrict dmesg access
kernel.kptr_restrict = 2               # Hide kernel pointers
```

Apply with: `sysctl -p`

Package integrity verification

```bash
# RPM-based (RHEL/CentOS)
rpm -Va                    # Verify all packages (S=size, 5=checksum, M=mode)

# Debian/Ubuntu
debsums -c                 # Show only changed files
```

Security audit find commands

```bash
# SUID/SGID binaries
find / -perm /6000 -type f -ls 2>/dev/null

# Files with no owner (orphaned)
find / -nouser -o -nogroup 2>/dev/null

# Recently modified files (incident response)
find / -mtime -1 -type f -ls 2>/dev/null

# World-writable directories (excluding /tmp)
find / -type d -perm -002 ! -path "/tmp/*" 2>/dev/null
```

Quick reference for incident response

Scenario	Command
Who logged in recently	`last -20`
Failed login attempts	`lastb` or `grep "Failed" /var/log/auth.log`
Current connections	`ss -tulnpe`
Listening services	`ss -tlnp`
Running processes tree	`ps auxf`
Open files by PID	`lsof -p <PID>`
Recent sudo commands	`grep sudo /var/log/auth.log`
SELinux denials	`ausearch -m AVC -ts recent`
Modified system files	`rpm -Va` or `debsums -c`
Check for rootkits	`rkhunter --check`

Conclusion

Linux security requires a defense-in-depth approach combining preventive controls, detection mechanisms, and response capabilities. The most critical takeaways: **never use 777 permissions**, **always disable root SSH login**, **enable auditd with CIS-recommended rules**, and **run regular integrity checks with AIDE or Lynis**. Real-world incidents consistently show that misconfigurations—particularly SUID bit abuse and weak sudo configurations—enable the majority of privilege escalation attacks. (Red Hat) (SecureFlag) Organizations following CIS Benchmarks eliminate 80-95% of known vulnerabilities, making compliance-based hardening the most effective security investment for Linux systems.