Table of Contents

01	`apparmor_parser`	Load and manage AppArmor security profiles
02	`auditctl`	Control the Linux audit system
03	`certbot`	Lets Encrypt client for obtaining SSL/TLS certificates
04	`chcon`	Change SELinux security context of files
05	`fail2ban-client`	Fail2ban client to manage ban rules
06	`firewall-cmd`	firewalld command line client (RHEL/CentOS)
07	`gpg`	GNU Privacy Guard - encrypt and sign data
08	`iptables`	Configure Linux kernel packet filtering firewall rules
09	`lynis`	Security auditing and hardening tool
10	`md5sum`	Compute and verify MD5 message digests

$ apparmor_parser

Advanced

Load and manage AppArmor security profiles

The apparmor_parser command loads, replaces, removes, and manages AppArmor security profiles. AppArmor is a Mandatory Access Control (MAC) system that confines programs to a limited set of resources - restricting file access, network capabilities, and other system operations based on per-program ...

Options & Flags

`-a`	Add and load a new profile
`-r`	Replace (reload) an existing profile
`-R`	Remove a profile from the kernel
`-C`	Set profile to complain mode
`-p`	Preprocess and dump profile (debug)
`aa-status`	Show AppArmor status and loaded profiles
`aa-enforce`	Set profile to enforce mode
`aa-genprof`	Generate a new profile interactively

Practical Examples

Example: Check AppArmor status

```
$ sudo aa-status
```

Show how many profiles are loaded, how many are in enforce vs complain mode, and which processes are confined.

Example: Load a profile

```
$ sudo apparmor_parser -a /etc/apparmor.d/usr.sbin.nginx
```

Load a new profile into the kernel. The profile takes effect immediately for new process invocations.

Example: Reload after editing

```
$ sudo apparmor_parser -r /etc/apparmor.d/usr.sbin.nginx
```

Replace the kernel profile with the updated file version. Use after editing a profile.

Example: Set to complain mode for testing

```
$ sudo aa-complain /etc/apparmor.d/usr.sbin.nginx
```

Switch to complain mode: violations are logged but not blocked. Use for testing new profiles.

Example: Generate profile for a program

```
$ sudo aa-genprof /usr/sbin/nginx
```

Interactive profile generator. Run the program normally in another terminal, then scan for access events and build the profile.

Tips & Best Practices

Pro Tip: Start with complain mode: When creating or modifying profiles, start in complain mode (aa-complain). Monitor logs for denials, adjust the profile, then switch to enforce mode (aa-enforce).

Note: Profile naming convention: Profile files in /etc/apparmor.d/ are named after the program path with dots: /usr/sbin/nginx becomes usr.sbin.nginx.

Warning: Test before enforcing: An incorrectly configured enforce-mode profile can break services. Always test in complain mode first and review denials in dmesg or /var/log/syslog.

Pro Tip: Use aa-logprof for updates: After running a program in complain mode, use aa-logprof to scan logs and update the profile with the observed access patterns.

$ auditctl

Advanced

Control the Linux audit system

The auditctl command is the primary administration tool for the Linux Audit System (auditd). It configures audit rules that track system calls, file access, user actions, and security-relevant events. The audit system is essential for security compliance, forensic investigation, and intrusion det...

Options & Flags

`-w PATH`	Watch a file or directory for access
`-p PERMISSIONS`	Set permissions to watch: r(ead), w(rite), x(ecute), a(tt...
`-k KEY`	Set a filter key for searching audit logs
`-a FILTER,ACTION`	Add a syscall audit rule
`-l`	List all current audit rules
`-D`	Delete all audit rules
`-d FILTER,ACTION`	Delete a specific rule
`-s`	Show audit system status
`-e 0|1|2`	Enable/disable audit (2=immutable)

Practical Examples

Example: Watch password file changes

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

Track writes and attribute changes to /etc/passwd. Search with: ausearch -k passwd_changes

Example: Monitor SSH config

```
$ sudo auditctl -w /etc/ssh/sshd_config -p wa -k sshd_config
```

Track any modifications to SSH server configuration. Critical for security auditing.

Example: Audit all command executions

```
$ sudo auditctl -a always,exit -F arch=b64 -S execve -k all_commands
```

Log every command executed on the system. Generates significant volume - use on sensitive servers only.

Example: Monitor user creation/deletion

```
$ sudo auditctl -a always,exit -F arch=b64 -S execve -F path=/usr/sbin/useradd -k user_management && sudo auditctl -a always,exit
```

Track when users are created or deleted. Essential for compliance.

Example: Watch sensitive directories

```
$ sudo auditctl -w /etc/cron.d/ -p wa -k cron_changes && sudo auditctl -w /etc/sudoers.d/ -p wa -k sudo_changes
```

Monitor changes to cron jobs and sudo rules - common targets for privilege escalation.

Tips & Best Practices

Warning: Performance impact: Auditing every execve call generates massive log volume. Be selective - watch specific files and critical syscalls. Monitor /var/log/audit/ disk usage.

Pro Tip: Use keys for organization: Always set -k (key) on rules. This makes searching with ausearch -k KEY fast and organized. Group related rules with the same key.

Note: Persistent rules: auditctl rules are lost on reboot. Save to /etc/audit/rules.d/*.rules for persistence. Run augenrules --load to reload without restart.

Pro Tip: Use aureport for summaries: aureport provides summary reports: aureport --auth (authentication), aureport --login (logins), aureport --file (file access). Great for daily security reviews.

$ certbot

Intermediate

Lets Encrypt client for obtaining SSL/TLS certificates

certbot is the Let's Encrypt client that automatically obtains and renews free SSL/TLS certificates. It is the standard tool for adding HTTPS to web servers with minimal effort.

certbot supports automatic configuration for Apache and Nginx, standalone certificate generation, and DNS-based challe...

Options & Flags

`--nginx`	Automatic Nginx configuration
`--apache`	Automatic Apache configuration
`certonly`	Obtain certificate without installing
`--standalone`	Use standalone web server for challenge
`--webroot`	Use existing web server for challenge
`renew`	Renew all certificates
`certificates`	List installed certificates

Practical Examples

Example: Setup Nginx HTTPS

```
$ sudo certbot --nginx -d example.com -d www.example.com
```

Obtains certificate and automatically configures Nginx.

Example: Certificate only

```
$ sudo certbot certonly --standalone -d example.com
```

Gets certificate without modifying web server config.

Example: Webroot method

```
$ sudo certbot certonly --webroot -w /var/www/html -d example.com
```

Gets certificate using existing web server (no downtime).

Example: Wildcard certificate

```
$ sudo certbot certonly --manual --preferred-challenges dns -d "*.example.com"
```

Gets a wildcard certificate via DNS challenge.

Example: Renew all certificates

```
$ sudo certbot renew --dry-run
```

Tests renewal process without actually renewing.

Tips & Best Practices

Pro Tip: Auto-renewal: certbot installs a systemd timer for auto-renewal. Verify with: systemctl status certbot.timer.

Note: Dry run first: Always test with --dry-run before actual renewal: certbot renew --dry-run. This avoids rate limit issues.

Warning: Rate limits: Let's Encrypt has rate limits: 50 certificates per domain per week. Use --dry-run or staging server for testing.

$ chcon

Advanced

Change SELinux security context of files

The chcon command changes the SELinux security context of files and directories. SELinux (Security-Enhanced Linux) is a mandatory access control (MAC) system that adds an additional layer of security beyond traditional Unix permissions. It is enabled by default on RHEL, Fedora, CentOS, AlmaLinux,...

Options & Flags

`-t TYPE`	Change the SELinux type of a file
`-R`	Change context recursively
`-u USER`	Change the SELinux user
`-r ROLE`	Change the SELinux role
`-l LEVEL`	Change the MLS/MCS security level
`--reference=FILE`	Set context to match another file
`-v`	Verbose - show each change

Practical Examples

Example: Fix web server file access

```
$ sudo chcon -Rt httpd_sys_content_t /var/www/html/
```

Set proper SELinux type for web content. Fixes "403 Forbidden" when Apache/Nginx cannot read files.

Example: Allow web server to write

```
$ sudo chcon -Rt httpd_sys_rw_content_t /var/www/html/uploads/
```

Allow web server to write to an uploads directory. httpd_sys_rw_content_t permits read-write.

Example: Check current context

```
$ ls -lZ /var/www/html/
system_u:object_r:httpd_sys_content_t:s0 index.html
```

View SELinux context of files. The -Z flag shows the security context alongside standard permissions.

Example: Copy context from reference file

```
$ sudo chcon --reference=/var/www/html/index.html /var/www/html/new-page.html
```

Set the context of new-page.html to match the existing index.html. Useful when adding files.

Example: Restore default contexts

```
$ sudo restorecon -Rv /var/www/html/
```

Reset file contexts to the default policy values. Undoes any chcon changes. Always use this before chcon for troubleshooting.

Tips & Best Practices

Warning: chcon changes are temporary: chcon changes are lost when restorecon runs or the filesystem is relabeled. For permanent changes, use semanage fcontext followed by restorecon.

Pro Tip: Troubleshoot with audit2why: When SELinux blocks access, check: sudo ausearch -m AVC --start today | audit2why. This tells you exactly what context change is needed.

Note: Common web types: httpd_sys_content_t (read-only web content), httpd_sys_rw_content_t (read-write), httpd_sys_script_exec_t (CGI scripts), httpd_log_t (log files).

Pro Tip: Check SELinux status: getenforce shows current mode: Enforcing (active), Permissive (logging only), Disabled. Use sestatus for detailed info.

$ fail2ban-client

Intermediate

Fail2ban client to manage ban rules

fail2ban-client controls the fail2ban service, which monitors log files for malicious activity (failed logins, scanning attempts) and automatically bans offending IP addresses using the firewall.

fail2ban reads log files, matches entries against configurable patterns (filters), and triggers acti...

Options & Flags

`status`	Show fail2ban status
`status JAIL`	Show jail status and banned IPs
`set JAIL banip`	Manually ban an IP
`set JAIL unbanip`	Unban an IP
`reload`	Reload configuration
`banned`	Show all banned IPs

Practical Examples

Example: Check status

```
$ sudo fail2ban-client status
Status
|- Number of jail:      2
`- Jail list:   sshd, nginx-botsearch
```

Shows fail2ban status and active jails.

Example: View jail details

```
$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Currently banned: 3
`- Banned IP list: 1.2.3.4 5.6.7.8 9.10.11.12
```

Shows SSH jail details including banned IPs.

Example: Ban IP manually

```
$ sudo fail2ban-client set sshd banip 203.0.113.50
```

Manually bans an IP in the SSH jail.

Example: Unban IP

```
$ sudo fail2ban-client set sshd unbanip 203.0.113.50
```

Removes a ban for a specific IP.

Example: Reload after config change

```
$ sudo fail2ban-client reload
```

Reloads jail configurations without restarting.

Tips & Best Practices

Pro Tip: Essential for public servers: fail2ban is one of the first services to configure on any internet-facing server. It dramatically reduces brute force attacks.

Note: Custom jails: Create custom jails in /etc/fail2ban/jail.local for your applications. Define filter patterns in /etc/fail2ban/filter.d/.

Warning: Whitelist your IP: Add your IP to ignoreip in jail.local to prevent accidentally banning yourself.

$ firewall-cmd

Intermediate

firewalld command line client (RHEL/CentOS)

firewall-cmd is the command-line client for firewalld, the default firewall management tool on RHEL, CentOS, and Fedora systems. It uses zones and services for organized rule management.

firewalld operates with zones (trusted, public, internal, etc.) that define trust levels for network connecti...

Options & Flags

`--state`	Check if firewalld is running
`--list-all`	List all rules for default zone
`--add-service`	Allow a service
`--add-port`	Allow a port
`--remove-service`	Remove a service
`--reload`	Reload to apply permanent changes
`--permanent`	Make change persistent

Practical Examples

Example: Check status

```
$ sudo firewall-cmd --state
running
```

Shows if firewalld is running.

Example: List all rules

```
$ sudo firewall-cmd --list-all
public (active)
  services: ssh dhcpv6-client http https
```

Shows all rules for the default zone.

Example: Allow HTTP/HTTPS

```
$ sudo firewall-cmd --permanent --add-service=http && sudo firewall-cmd --permanent --add-service=https && sudo firewall-cmd --rel
```

Permanently allows web traffic and reloads.

Example: Allow custom port

```
$ sudo firewall-cmd --permanent --add-port=8080/tcp && sudo firewall-cmd --reload
```

Opens port 8080 permanently.

Example: Allow from specific IP

```
$ sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="10.0.0.0/24" port port="3306" protocol="tcp" a
```

Allows MySQL access from a specific network.

Tips & Best Practices

Warning: --permanent + --reload: Changes without --permanent are temporary. Always add --permanent for persistent rules, then --reload to apply.

Pro Tip: Use services over ports: firewall-cmd --add-service=http is more readable and maintainable than --add-port=80/tcp.

Note: Zones: firewall-cmd --get-zones lists available zones. --zone=trusted is fully open. --zone=public is default (restrictive).

$ gpg

Advanced

GNU Privacy Guard - encrypt and sign data

gpg (GNU Privacy Guard) provides encryption, digital signatures, and key management using public-key cryptography. It implements the OpenPGP standard for secure communication and file protection.

gpg is used for encrypting files, signing documents, verifying software signatures, secure email, an...

Options & Flags

`-c`	Symmetric encryption (password only)
`-e`	Encrypt for a recipient
`-d`	Decrypt
`-s`	Sign a file
`--verify`	Verify a signature
`--gen-key`	Generate a key pair
`--list-keys`	List public keys
`--import`	Import a key

Practical Examples

Example: Encrypt with password

```
$ gpg -c sensitive_data.tar.gz
```

Encrypts with a symmetric password. Creates sensitive_data.tar.gz.gpg.

Example: Decrypt file

```
$ gpg -d backup.tar.gz.gpg > backup.tar.gz
```

Decrypts a file using the password or your private key.

Example: Generate key pair

```
$ gpg --full-gen-key
```

Creates a new PGP key pair with interactive prompts.

Example: Verify downloaded file

```
$ gpg --verify file.sig file.tar.gz
gpg: Good signature from "Author Name"
```

Verifies that a downloaded file signature is valid.

Example: Export public key

```
$ gpg --armor --export alice@example.com > alice_public.asc
```

Exports your public key in ASCII format for sharing.

Tips & Best Practices

Pro Tip: Use -c for simple encryption: gpg -c file uses symmetric encryption with a password. No key management needed. Good for personal backups.

Warning: Backup your private key: If you lose your private key, encrypted data is unrecoverable. Export with gpg --export-secret-keys > backup.gpg.

Note: Verifying software: Many Linux downloads include .sig files. Import the developer key, then gpg --verify file.sig file to verify authenticity.

$ iptables

Advanced

Configure Linux kernel packet filtering firewall rules

iptables is the traditional Linux firewall tool that manages packet filtering rules in the kernel. It controls incoming, outgoing, and forwarded network traffic using chains (INPUT, OUTPUT, FORWARD) and tables (filter, nat, mangle).

iptables is powerful but complex - each rule must be specified ...

Options & Flags

`-A`	Append rule to chain
`-I`	Insert rule at position
`-D`	Delete rule
`-L`	List rules
`-F`	Flush (delete all rules)
`-P`	Set default policy
`-j`	Jump target (ACCEPT, DROP, REJECT)
`-s/-d`	Source/destination IP

Practical Examples

Example: List all rules

```
$ sudo iptables -L -v -n --line-numbers
```

Shows all firewall rules with line numbers, verbose, and numeric output.

Example: Allow SSH

```
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Allows incoming SSH connections.

Example: Allow web traffic

```
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT && sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Allows HTTP and HTTPS traffic.

Example: Drop all other incoming

```
$ sudo iptables -P INPUT DROP
```

Sets default policy to drop all incoming traffic not matching a rule.

Example: Allow established connections

```
$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Allows return traffic for connections you initiated.

Tips & Best Practices

Warning: Rules are not persistent: iptables rules are lost on reboot. Save with iptables-save and restore with iptables-restore. Install iptables-persistent on Debian/Ubuntu.

Pro Tip: Order matters: Rules are evaluated top-down. Place specific rules before general ones. A DROP at the top blocks everything below.

Note: Use ufw or firewall-cmd instead: For simple firewall needs, ufw (Ubuntu) or firewall-cmd (RHEL) are much easier. Learn iptables for advanced scenarios.

$ lynis

Intermediate

Security auditing and hardening tool

The lynis command performs comprehensive security audits on Linux, macOS, and Unix systems. It checks system configuration, installed software, security settings, and potential vulnerabilities, then provides a hardening index score and actionable recommendations.

Lynis examines hundreds of secur...

Options & Flags

`audit system`	Perform a full system security audit
`audit system --quick`	Run audit without waiting for user input
`show details TEST-ID`	Show details of a specific test
`show profiles`	List available audit profiles
`--pentest`	Non-privileged scan (penetration test mode)
`--log-file FILE`	Specify custom log file location
`--report-file FILE`	Custom report file location
`--cronjob`	Run for automation (no colors, no prompts)

Practical Examples

Example: Full system audit

```
$ sudo lynis audit system
```

Run a comprehensive security audit. Checks boot, kernel, memory, users, networking, firewall, SSH, files, and more. Interactive - press Enter to continue.

Example: Quick non-interactive audit

```
$ sudo lynis audit system --quick
```

Run the full audit without user prompts. Outputs results directly - ideal for scripts and scheduled runs.

Example: View hardening score

```
$ sudo lynis audit system --quick 2>/dev/null | grep "Hardening index"
Hardening index : 72 [##############        ]
```

Extract the hardening index (0-100). Higher is better. Aim for 80+ on production servers.

Example: Show warnings and suggestions

```
$ sudo grep -E "^warning|^suggestion" /var/log/lynis-report.dat
```

Extract all warnings and suggestions from the report file for systematic remediation.

Example: Scheduled audit with cron

```
$ 0 3 * * 0 /usr/bin/lynis audit system --cronjob --report-file /var/log/lynis-weekly.dat
```

Weekly Sunday 3 AM audit. --cronjob disables colors and prompts for clean automated output.

Tips & Best Practices

Pro Tip: Install lynis: Install: apt install lynis (Debian/Ubuntu), dnf install lynis (Fedora/RHEL), or clone from GitHub: git clone https://github.com/CISOfy/lynis. Run from git clone with ./lynis.

Note: Report and log files: Audit log: /var/log/lynis.log (detailed). Report: /var/log/lynis-report.dat (machine-readable). These files contain the full audit results.

Warning: Not all suggestions apply: Lynis suggestions are generic. Not every recommendation applies to your use case. Evaluate each suggestion in your specific context before implementing.

Pro Tip: Track improvements: After implementing hardening changes, re-run lynis to see your score improve. Aim to address all warnings first, then work through suggestions.

$ md5sum

Beginner

Compute and verify MD5 message digests

md5sum calculates and verifies MD5 message digests (128-bit hash). It produces a unique fingerprint of a file that can be used to verify data integrity - any change to the file produces a different hash.

md5sum is commonly used to verify downloaded files, check file integrity after transfers, an...

Options & Flags

`file`	Calculate MD5 hash
`-c`	Check hashes from file
`-b`	Binary mode
`--quiet`	Only show failures when checking

Practical Examples

Example: Calculate hash

```
$ md5sum ubuntu.iso
e8a123b456c789... ubuntu.iso
```

Computes the MD5 hash of a file.

Example: Verify download

```
$ md5sum -c MD5SUMS
ubuntu.iso: OK
```

Checks all files listed in MD5SUMS against their expected hashes.

Example: Compare files

```
$ md5sum file1.txt file2.txt
d41d8cd98f00... file1.txt\nd41d8cd98f00... file2.txt
```

Generates hashes to compare if two files are identical.

Example: Hash a string

```
$ echo -n "hello" | md5sum
5d41402abc4b... -
```

Calculates MD5 of a string (use -n to avoid newline).

Example: Create checksum file

```
$ md5sum *.iso > checksums.md5
```

Generates a checksum file for all ISOs.

Tips & Best Practices

Warning: MD5 is not secure: MD5 is cryptographically broken - collisions can be crafted. Use sha256sum for security-critical verification.

Pro Tip: Quick file comparison: To check if two files are identical without comparing content: md5sum file1 file2. Same hash = same content.

Note: Use -n with echo: echo "text" | md5sum includes a newline in the hash. Use echo -n "text" | md5sum for the string without newline.