Cloud Web Server Security Report

Name: Mitchelle Anduru
Date: 22/01/2026
Role: Junior DevOps Engineer

Executive Summary

This engagement assessed a client's locally hosted Docker-based Ubuntu 22.04 web server (Apache 2.4.52) and identified key limitations affecting scalability, security assurance, and disaster recovery. A baseline Lynis audit scored **59**, reflecting gaps in password/permission defaults, web attack resilience, integrity monitoring, and patching maturity—risks that increase compromise likelihood and make compliance evidence harder to produce. Targeted hardening actions were implemented (stronger /etc/login.defs defaults, reduced Apache information leakage, DoS/WAF protections via mod_evasive and mod_security2, disabled core dumps, initialized AIDE, improved patch workflow tooling, and configured brute-force controls), improving the post-hardening score to **73**. Despite these improvements, local-only hosting remains a single point of failure with limited scaling and recovery automation; therefore, a **public cloud PaaS** deployment with managed load balancing, WAF, centralized monitoring, automated backups, and infrastructure-as-code is recommended to meet availability, security, and DR objectives sustainably.

Key Findings

- Local-only Docker hosting introduces **single-point-of-failure risk**, limited horizontal scaling, and weak recovery automation.
- Baseline hardening score **59** indicated gaps in **password/permission hygiene**, **integrity monitoring**, and **web attack protections** (compliance evidence risk).
- Implemented layered defenses (Apache hardening + WAF/DoS modules + AIDE + safer OS defaults) improved score to **73**.
- Missing/limited DR readiness remains the biggest business risk without multi-instance deployment, automated backups, and health-checked failover.
- A **public cloud PaaS** model best reduces operational burden while improving scalability, auditability, and recovery speed.

Before → After Snapshot (mini-table)

Area	Before	After
Lynis hardening index	59	73
Web exposure resilience	Basic Apache defaults	Reduced server info leakage + mod_security2 + mod_evasive

Integrity monitoring	None	AIDE initialized + baseline created
Sensitive data leakage	Core dumps enabled	Core dumps disabled (ulimit -c = 0)
Brute-force mitigation	Weak/absent	fail2ban configured (environment-dependent runtime)
Patch maturity	Limited	unattended upgrades + needrestart + apt-listchanges

Assumptions & Limitations

This assessment was performed in a **Docker-based lab environment**, which may lack full service management (e.g., systemctl). Some controls (such as fail2ban runtime enforcement) are fully valid in standard VM/cloud deployments but may be limited in containers without init systems. Findings and remediation evidence assume access to system configurations (/etc/*), Apache configs, and audit tools (Lynis/AIDE), and reflect a controlled lab scenario rather than a live production incident.

Cloud Solution

Recommended PaaS Public Cloud Architecture

- Managed **Load Balancer** + **Autoscaling** web tier (multi-zone)
- Managed **WAF** + rate limiting (replace/enhance mod_security2/mod_evasive)
- Centralized logging (access/error + security logs) into SIEM/log analytics
- Metrics + alerting (latency, 5xx, auth failures, WAF blocks, CPU/memory)
- Automated backups + versioned object storage + retention policies (RPO/RTO aligned)
- Infrastructure-as-code for repeatable secure deployments (baseline configs)
- Secrets management (no secrets in images/repos)
- IAM least-privilege + MFA for admin access

Understanding Customer Challenges and Requirements

Client Challenges

- **Current Setup and Issues**: Document the client's current webserver setup (a local webserver using Docker) and describe its limitations.
  - *Instructions*: Explain why this local-only setup presents challenges for scalability, compliance, or disaster recovery.

Answer

The client is hosting their application on a local webserver using Docker, running Ubuntu 22.04 with Apache 2.4.52. This setup is functional but creates major business risks:

- Scalability limitation: A single locally hosted server has fixed resources. When traffic spikes, the client cannot automatically scale horizontally (more servers) or use a load balancer easily. This causes slow performance and downtime.

- Security risk: The baseline hardening posture showed missing/weak controls (patch management hygiene, account/password controls, brute-force protection, integrity monitoring, and web attack protections). This increases the risk of compromise and service disruption.

- Compliance challenges: Local-only setup typically lacks standardized security controls and audit evidence (e.g., stronger configuration baselines, consistent logging/auditing, WAF protections). This makes compliance harder to demonstrate.

- Disaster recovery weakness: If this host fails, the service fails. With no **redundancy, backups, or failover, recovery becomes manual and slow.**

Audit Results

- **Baseline Security Posture**: Record the **hardening index score** from the Lynis audit and summarize the vulnerabilities identified.
    - *Instructions*: Connect the vulnerabilities to the client's business challenges. For example, how does missing HTTPS impact compliance or disaster recovery?

Answer

Baseline Security Posture (Lynis)

- Baseline hardening index (pre-hardening): 59

- Final hardening index (post-hardening): 73

Key vulnerabilities / gaps identified and linked to business challenges:

- Weak webserver protection controls → increased exposure to DoS and web attacks (business risk: outage + data compromise).

- Weak/insufficient system hardening defaults (password aging/umask/hashing rounds) → higher chance of account compromise (business risk: unauthorized access).

- No integrity monitoring baseline → harder to detect unauthorized changes (business risk: hidden compromise + unreliable recovery).

- Core dumps enabled (risk) → potential sensitive data leakage in crashes (business risk: data exposure).

- Patch workflow maturity needed → delayed security updates increases breach likelihood (business risk: known vulnerability exploitation).

Evaluating Cloud Service Models

Service Model Comparison

- **Analysis of Models**: Compare the client's responsibilities under IaaS, PaaS, and SaaS models.
    - *Instructions*: Identify which vulnerabilities (e.g., missing file permissions or insecure configurations) would remain the client's responsibility under each model.

Answer

IaaS (Infrastructure as a Service)

- Client manages: OS hardening, patching strategy, Apache config, security modules, logging/auditing, monitoring, backups.

- In our lab, everything we did (Lynis hardening actions) is exactly what the client must do on IaaS VMs.

PaaS (Platform as a Service)

- Cloud provider manages more of the underlying OS/runtime maintenance, scaling primitives, and managed security options.

- Client still manages: app security, configuration, identity/access policies, logging, and secure deployment patterns.

- Many OS-level hardening burdens reduce, which is ideal for a small team.

SaaS (Software as a Service)

- Cloud provider handles almost everything.
- Client only configures the application and user access.
- Only suitable if the client's "web app" can be replaced by a standard SaaS product (often not the case for custom apps).

Recommended Model

- **Chosen Service Model**: Recommend a cloud service model (e.g., IaaS, PaaS, or SaaS) based on the client's goals.
  - *Instructions*: Justify your recommendation by explaining how this model

addresses the client's scalability, compliance, or security concerns. **Answer**

Why PaaS fits the client:

- Scalability: built-in autoscaling and managed load balancing is easier than managing it manually.
- Security: reduces OS-level maintenance burden while still allowing secure configuration patterns and managed security controls (e.g., WAF, logging).
- Disaster recovery: easier to implement backups, multi-zone deployment, and automated recovery options.

Exploring Deployment Options

Deployment Choices

- **Analysis of Deployment Options**: Compare public, private, and hybrid cloud options.
  - *Instructions*: Explain how each option impacts scalability, security, and disaster recovery. For example, discuss why a public cloud might require stricter security measures.

Answer

Public Cloud

Best for rapid scaling (autoscaling, load balancers, managed DB/storage).
Requires strong security controls because it’s internet-facing (WAF, least privilege, monitoring).
Best overall fit for the client’s scalability + DR needs.

Private Cloud

More control and potentially easier for strict data rules, but expensive and harder to scale fast.
Higher operational workload.

Hybrid Cloud

Good if sensitive data must stay on-prem but web front-end needs public cloud scalability. More complexity (networking + governance).

Recommended Deployment Option

- **Chosen Deployment Option**: Recommend a deployment option for the client (e.g., public cloud for scalability, hybrid for flexibility).
    - *Instructions*: Justify your choice based on the client’s challenges and the vulnerabilities identified in the audit.

Answer

Solves the core pain fastest: scaling + availability + DR.
Security controls you implemented locally (WAF-style modules, integrity monitoring concept, stronger configs) translate well into public-cloud best practices.

Developing a Cloud Solution Recommendation

Hardening Actions

- **Actions Taken**: Document the steps you took to remediate vulnerabilities on the local web server.
    - *Instructions*: For each vulnerability, provide the following:
        - **The Issue**: Describe the vulnerability (e.g., missing HTTPS, weak file permissions).
        - **The Action**: Document the exact commands or configurations used to address the issue.

- **The Result**: Explain how the remediation improves security.

Answer

1) Weak account/password defaults

Issue: Password aging and default file permissions were not strict enough.
Action: Updated /etc/login.defs with:

- PASS_MAX_DAYS 90, PASS_MIN_DAYS 1, PASS_WARN_AGE 7
- UMASK 027
- SHA_CRYPT_MIN_ROUNDS 5000, SHA_CRYPT_MAX_ROUNDS 10000

Result: Stronger password lifecycle controls + safer default file permissions, reducing account and data exposure risk.

2) Apache information leakage

Issue: Webserver could reveal too much version/signature information.
Action: Updated /etc/apache2/conf-available/security.conf:

- ServerTokens Prod
- ServerSignature Off

Result: Reduces attacker reconnaissance and lowers targeted exploit risk.

3) Missing DoS / web attack protections

Issue: Lynis recommended additional Apache protections against DoS and web attacks. Action: Enabled modules:

- mod_evasive (DoS/bruteforce mitigation)
- mod_security2 (WAF-like protections)

  Verified enabled modules: evasive20_module, security2_module, unique_id_module.

Result: Adds layered protection against common web threats and improves resilience.

4) Core dumps enabled (sensitive data leakage risk)

Issue: Core dumps can expose sensitive memory content.
Action: Disabled core dumps with:

- `* hard core 0` in `/etc/security/limits.conf`
- Verified `ulimit -c` returns `0`

Result: Reduces risk of secrets being leaked during crashes.

5) Integrity monitoring missing

Issue: No way to detect unauthorized changes to critical files.
Action: Initialized AIDE database and verified checks detect system changes (e.g., new Apache module configs).Result: Enables tamper detection and supports incident response and recovery validation.

6) Brute-force mitigation (configured)

Issue: Lynis recommended fail2ban.

Action: Installed `fail2ban` and configured jail override:

- `/etc/fail2ban/jail.d/sshd.local`

Result: Provides automated banning logic for repeated authentication failures (note: service start may be limited in container environments without systemd, but configuration is in place and would be enabled on a VM/cloud host).

7) Patch workflow hardening (operational security)

Issue: Upgrades can be risky and services may not restart after patched libraries. Action: Installed `apt-listchanges`, `libpam-tmpdir`, `needrestart`; enabled unattended upgrades via `/etc/apt/apt.conf.d/20auto-upgrades`.

Result: More reliable patch management workflow and reduced exposure time to known vulnerabilities.

Cloud Solution Recommendation

- **Recommendation**: Propose a cloud solution that integrates the lessons learned from the hardening process.
  - *Instructions*: Justify your recommendation by explaining how the chosen cloud solution (e.g., public cloud with additional hardening) addresses the client's scalability, security, and disaster recovery needs.

Answer

Managed load balancer + autoscaling for web tier
Managed WAF (cloud-native) to replicate/enhance mod_security benefits
Centralized logging/monitoring + alerts
Automated patching and secure configuration baselines
Infrastructure-as-code for repeatable secure deployments

Justification: My hardening work improved security (59→73), but local hosting still limits scaling and DR. A public cloud PaaS approach reduces operational burden while improving availability and recovery readiness.

Disaster Recovery Planning

Proposed Recovery Plan

- **Backups**: Outline a plan for backing up the web server's configurations and data.
  - *Instructions*: Explain how regular backups protect the client from data loss and improve recovery time.

Back up web configs (`/etc/apache2`, application configs), application data, and deployment artifacts.

Schedule automated backups to offsite storage with retention (e.g., daily + weekly copies).

Define basic targets:

- **RPO:** 1 hour (max data loss)

- **RTO:** 1 hour (max downtime)
- **Failover Mechanisms**: Describe strategies for high availability (e.g., redundant web servers, load balancing).
  - *Instructions*: Explain how these mechanisms ensure the web server remains available during a failure.

Deploy at least 2 web instances across different zones.

Use a load balancer with health checks.
Auto-replace unhealthy instances (autoscaling group or platform scaling).

- **Monitoring and Alerts**: Propose logging and monitoring tools to detect and respond to issues proactively.

- *Instructions*: Explain how monitoring contributes to disaster recovery and reduces downtime.

Centralize logs (web access/error logs + system security logs).
Alert on: high error rate, latency spikes, repeated auth failures, suspicious traffic patterns.

Use dashboards to detect issues early and reduce downtime.

Connection to Hardening Actions

- **How Hardening Supports Recovery**: Explain how the security remediations you applied (e.g., enabling HTTPS, configuring permissions) contribute to disaster recovery goals.
  - *Instructions*: Tie each remediation to a recovery objective (e.g., secure backups, faster recovery).

AIDE integrity checks help confirm restored systems weren't altered (secure recovery validation).
mod_security/mod_evasive reduces successful attacks that cause outages and reduces incident frequency.
Stronger password policies/umask reduces risk of credential compromise and data exposure before/after recovery.
Core dump disabled reduces sensitive data leakage during incident or crash recovery.
Unattended upgrades + patch workflow tools reduce the chance of avoidable incidents from unpatched vulnerabilities.

Reflection on the Cloud Process

Reflection on Each Phase

- **Understanding Challenges**:
  - *Instructions*: Reflect on how analyzing customer challenges helped you understand their needs and guided your actions.

Analyzing the client's local setup clarified that the biggest business risks are limited scaling, weak recovery capability, and inconsistent security controls. That guided the decision to start with a baseline audit and prioritize the highest-impact fixes.

- **Evaluating Service Models**:
  - *Instructions*: Reflect on what you learned about cloud service models and how they impact security responsibilities.

I learned that **IaaS requires the client to own most hardening tasks** (like we did), while **PaaS reduces operational overhead** and improves scalability/DR readiness for small teams.

- **Exploring Deployment Options**:
    - *Instructions*: Reflect on how considering deployment options helped you align your recommendations with customer goals.

Public cloud best addresses scaling and DR quickly, but requires strong security guardrails. Hybrid can be justified if the client has strict data constraints.

- **Hardening and Recommendations**:
    - *Instructions*: Reflect on the impact of the hardening process and how it informed your recommendation for a scalable cloud solution.

Hardening turned audit findings into measurable improvement (hardening index 59→73) and gave evidence-based justification for moving to a cloud model that scales and standardizes security.

- **Disaster Recovery**:
    - *Instructions*: Reflect on the importance of disaster recovery and how it connects to the hardening actions you applied.

DR planning is not separate from security: hardening controls (integrity monitoring, WAF protections, patching) reduce incident frequency and make recovery more trustworthy and faster.

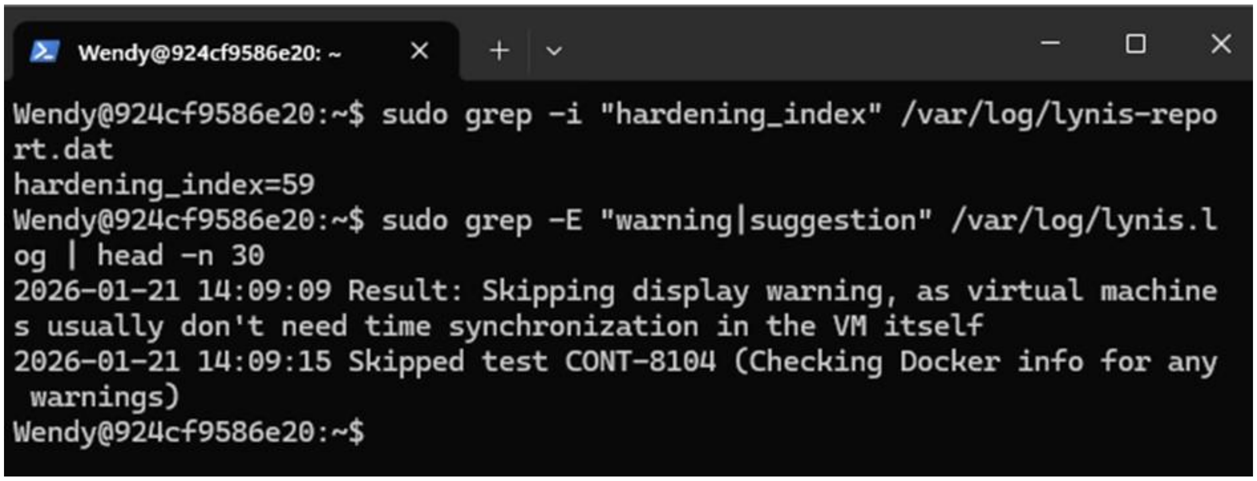
Overall Reflection

- Summarize your key takeaways from the lab.
- *Instructions*: Discuss how this process can be applied to real-world cloud security scenarios, and what you learned about balancing technical tasks with strategic decision-making.

This lab showed a real-world process: **baseline** → **prioritize** → **remediate** → **re-audit** → **recommend cloud architecture**. In real cloud projects, the same approach applies, but with stronger automation (IaC), managed services (logging/WAF), and multi-zone design to balance technical execution with strategic business goals.

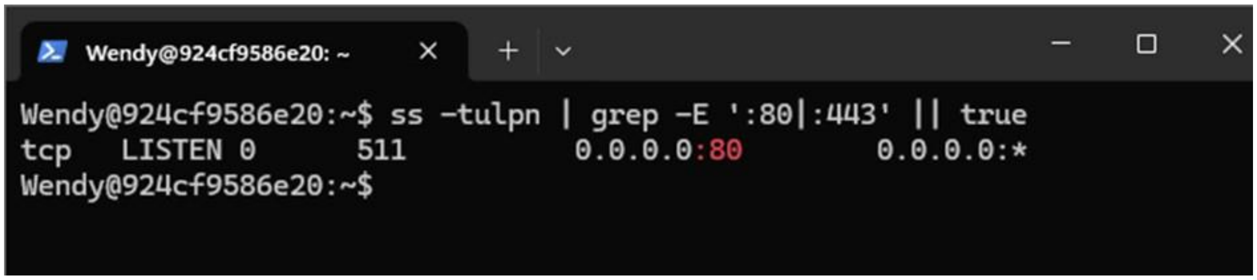
Screenshots

Immediately after the audit, extract the key evidence



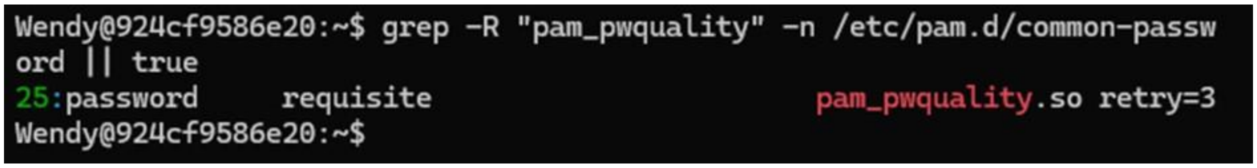
```
Wendy@924cf9586e20:~$ sudo grep -i "hardening_index" /var/log/lynis-repo
rt.dat
hardening_index=59
Wendy@924cf9586e20:~$ sudo grep -E "warning|suggestion" /var/log/lynis.l
og | head -n 30
2026-01-21 14:09:09 Result: Skipping display warning, as virtual machine
s usually don't need time synchronization in the VM itself
2026-01-21 14:09:15 Skipped test CONT-8104 (Checking Docker info for any
 warnings)
Wendy@924cf9586e20:~$
```

Ports check



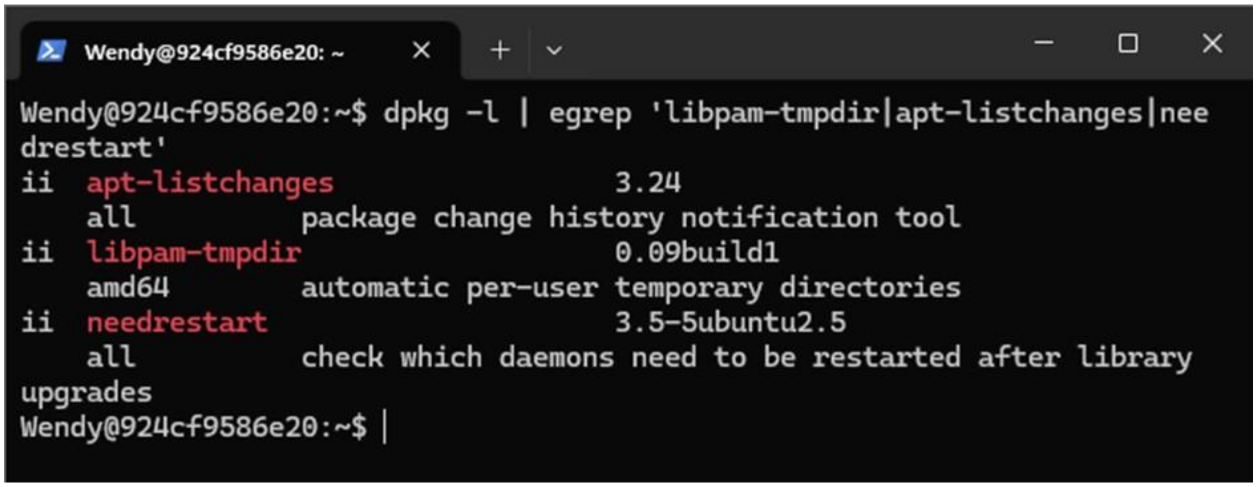
```
Wendy@924cf9586e20:~$ ss -tulpn | grep -E ':80|:443' || true
tcp   LISTEN 0      511          0.0.0.0:80        0.0.0.0:*
Wendy@924cf9586e20:~$
```

Install password quality module



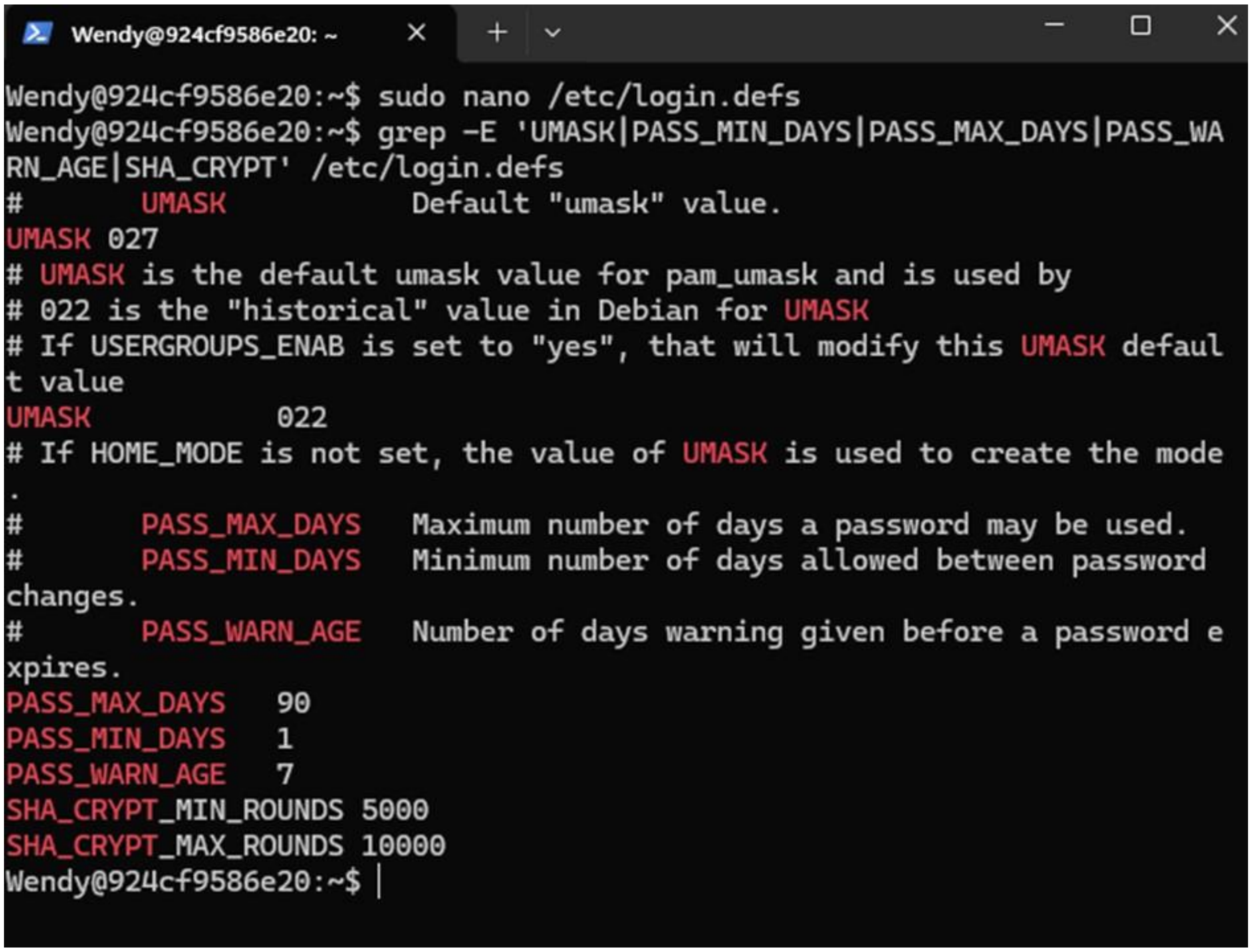
```
Wendy@924cf9586e20:~$ grep -R "pam_pwquality" -n /etc/pam.d/common-passw
ord || true
25:password     requisite                       pam_pwquality.so retry=3
Wendy@924cf9586e20:~$
```

Report (Issue → Action → Result)



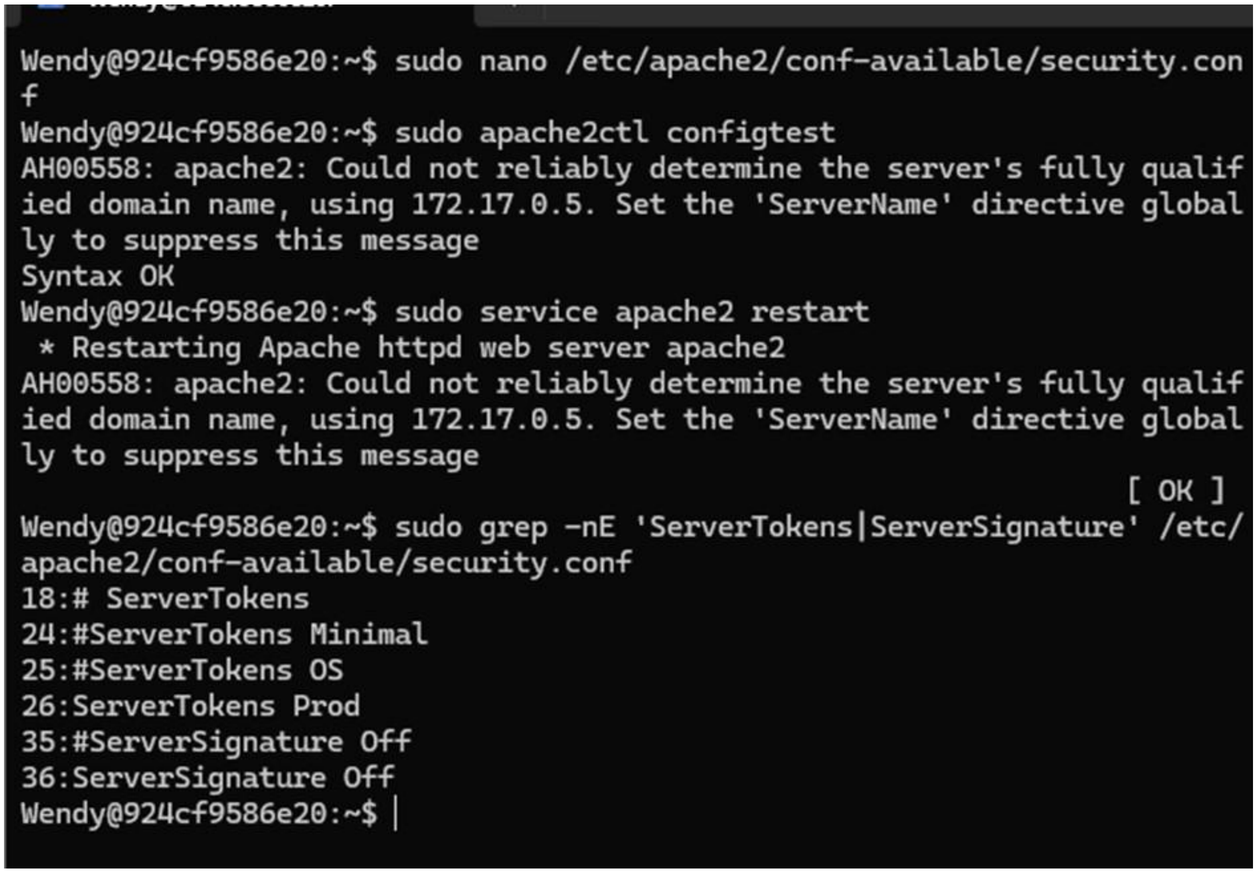
```
Wendy@924cf9586e20:~$ dpkg -l | egrep 'libpam-tmpdir|apt-listchanges|nee
drestart'
ii  apt-listchanges                  3.24
    all          package change history notification tool
ii  libpam-tmpdir                    0.09build1
    amd64        automatic per-user temporary directories
ii  needrestart                      3.5-5ubuntu2.5
    all          check which daemons need to be restarted after library
upgrades
Wendy@924cf9586e20:~$ |
```

Harden /etc/login.defs

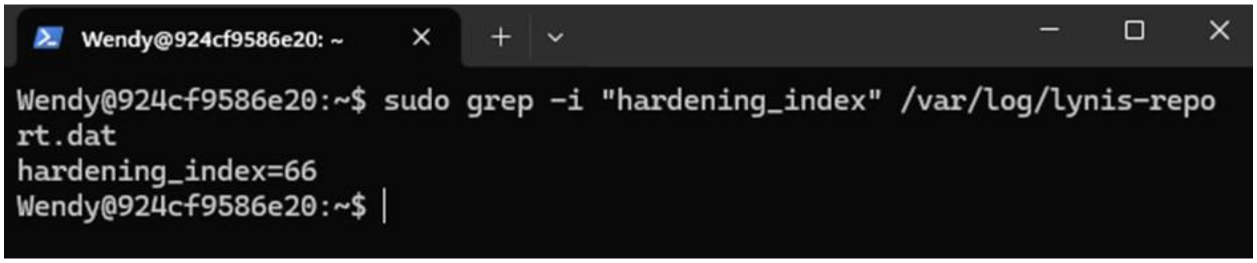


```
Wendy@924cf9586e20:~$ sudo nano /etc/login.defs
Wendy@924cf9586e20:~$ grep -E 'UMASK|PASS_MIN_DAYS|PASS_MAX_DAYS|PASS_WA
RN_AGE|SHA_CRYPT' /etc/login.defs
#       UMASK           Default "umask" value.
UMASK 027
# UMASK is the default umask value for pam_umask and is used by
# 022 is the "historical" value in Debian for UMASK
# If USERGROUPS_ENAB is set to "yes", that will modify this UMASK defaul
t value
UMASK           022
# If HOME_MODE is not set, the value of UMASK is used to create the mode
.
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password
changes.
#       PASS_WARN_AGE   Number of days warning given before a password e
xpires.
PASS_MAX_DAYS   90
PASS_MIN_DAYS   1
PASS_WARN_AGE   7
SHA_CRYPT_MIN_ROUNDS 5000
SHA_CRYPT_MAX_ROUNDS 10000
Wendy@924cf9586e20:~$ |
```

Apache quick hardening

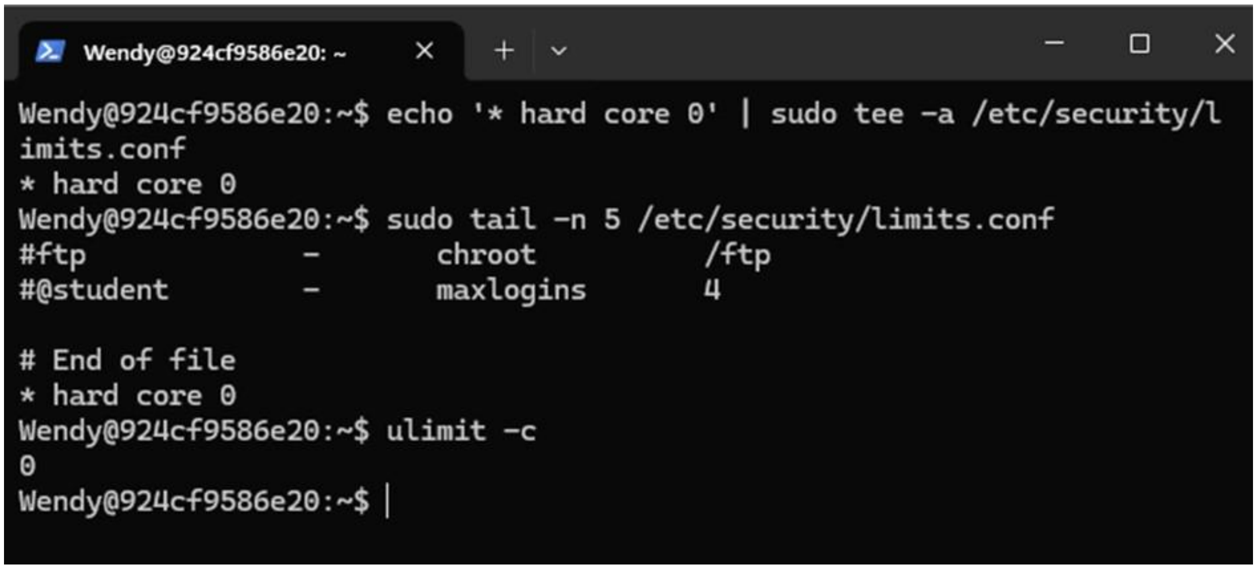


```
Wendy@924cf9586e20:~$ sudo nano /etc/apache2/conf-available/security.con
f
Wendy@924cf9586e20:~$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualif
ied domain name, using 172.17.0.5. Set the 'ServerName' directive global
ly to suppress this message
Syntax OK
Wendy@924cf9586e20:~$ sudo service apache2 restart
 * Restarting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully qualif
ied domain name, using 172.17.0.5. Set the 'ServerName' directive global
ly to suppress this message
                                                                  [ OK ]
Wendy@924cf9586e20:~$ sudo grep -nE 'ServerTokens|ServerSignature' /etc/
apache2/conf-available/security.conf
18:# ServerTokens
24:#ServerTokens Minimal
25:#ServerTokens OS
26:ServerTokens Prod
35:#ServerSignature Off
36:ServerSignature Off
Wendy@924cf9586e20:~$ |
```



```
Wendy@924cf9586e20:~$ sudo grep -i "hardening_index" /var/log/lynis-repo
rt.dat
hardening_index=66
Wendy@924cf9586e20:~$ |
```

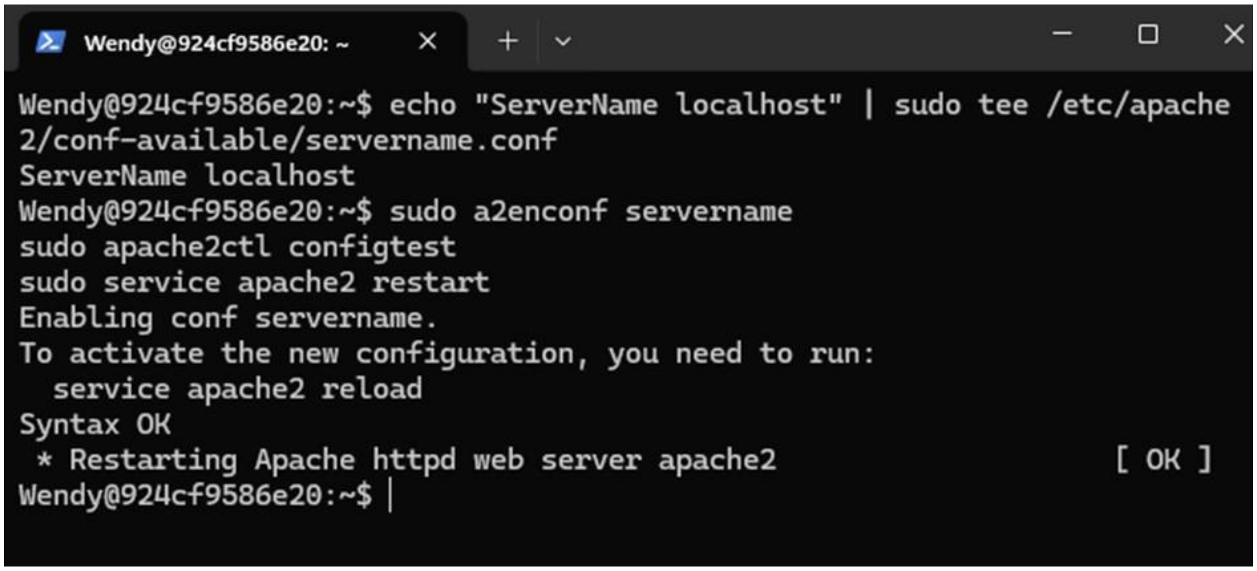
Disable core dumps



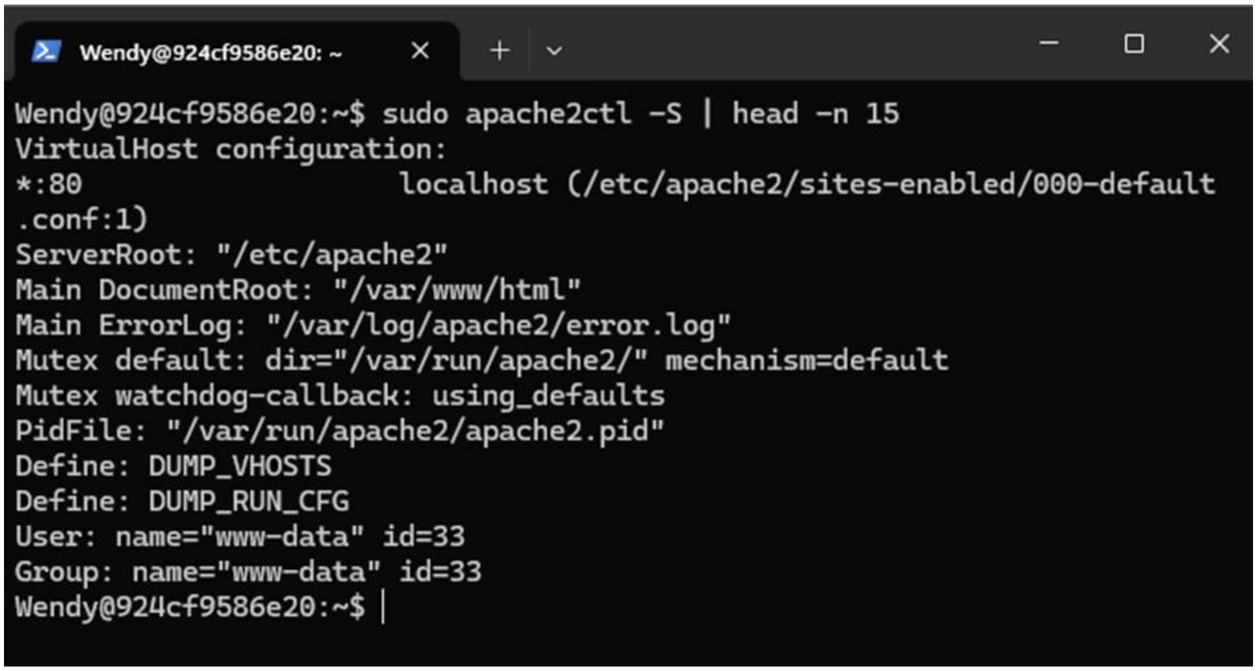
```
Wendy@924cf9586e20:~$ echo '* hard core 0' | sudo tee -a /etc/security/l
imits.conf
* hard core 0
Wendy@924cf9586e20:~$ sudo tail -n 5 /etc/security/limits.conf
#ftp             -       chroot          /ftp
#@student        -       maxlogins       4

# End of file
* hard core 0
Wendy@924cf9586e20:~$ ulimit -c
0
Wendy@924cf9586e20:~$ |
```

Fix the Apache “ServerName” warning

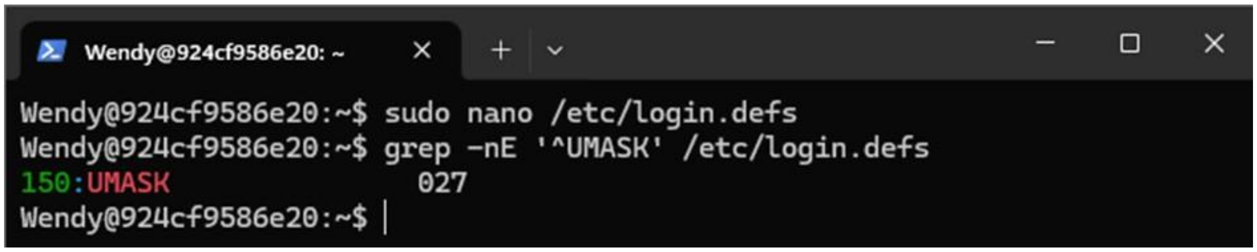


```
Wendy@924cf9586e20:~$ echo "ServerName localhost" | sudo tee /etc/apache
2/conf-available/servername.conf
ServerName localhost
Wendy@924cf9586e20:~$ sudo a2enconf servername
sudo apache2ctl configtest
sudo service apache2 restart
Enabling conf servername.
To activate the new configuration, you need to run:
  service apache2 reload
Syntax OK
 * Restarting Apache httpd web server apache2                    [ OK ]
Wendy@924cf9586e20:~$ |
```

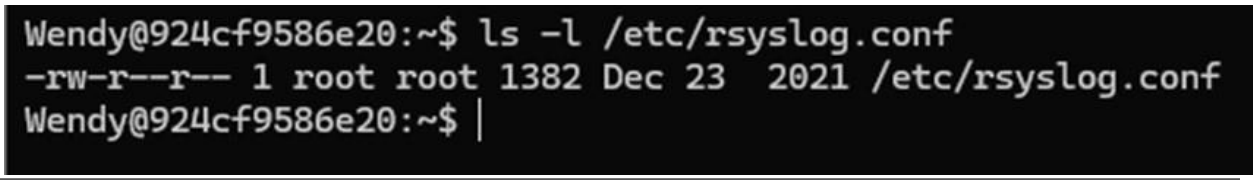


```
Wendy@924cf9586e20:~$ sudo apache2ctl -S | head -n 15
VirtualHost configuration:
*:80                   localhost (/etc/apache2/sites-enabled/000-default
.conf:1)
ServerRoot: "/etc/apache2"
Main DocumentRoot: "/var/www/html"
Main ErrorLog: "/var/log/apache2/error.log"
Mutex default: dir="/var/run/apache2/" mechanism=default
Mutex watchdog-callback: using_defaults
PidFile: "/var/run/apache2/apache2.pid"
Define: DUMP_VHOSTS
Define: DUMP_RUN_CFG
User: name="www-data" id=33
Group: name="www-data" id=33
Wendy@924cf9586e20:~$
```

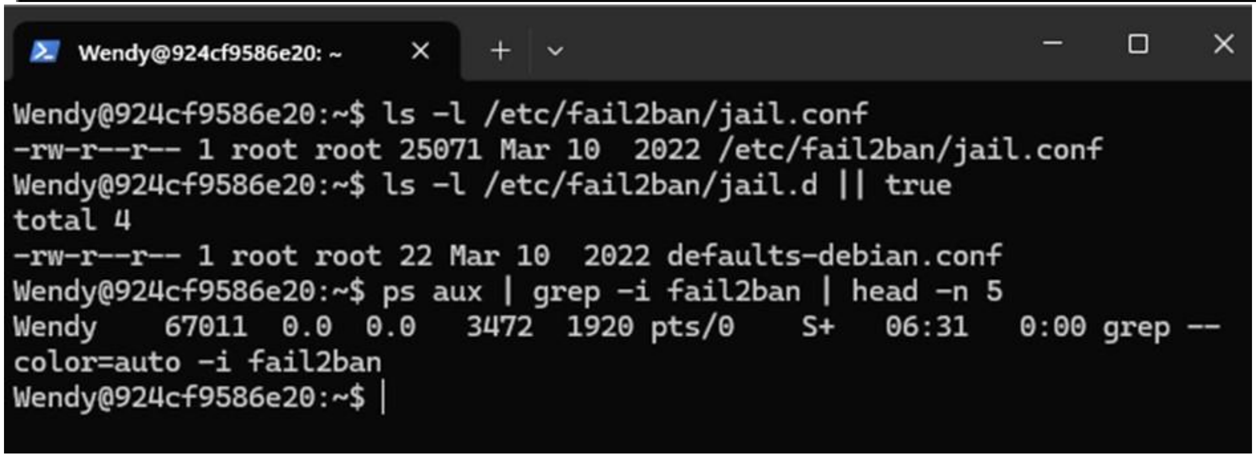
Clean up login.defs UMASK



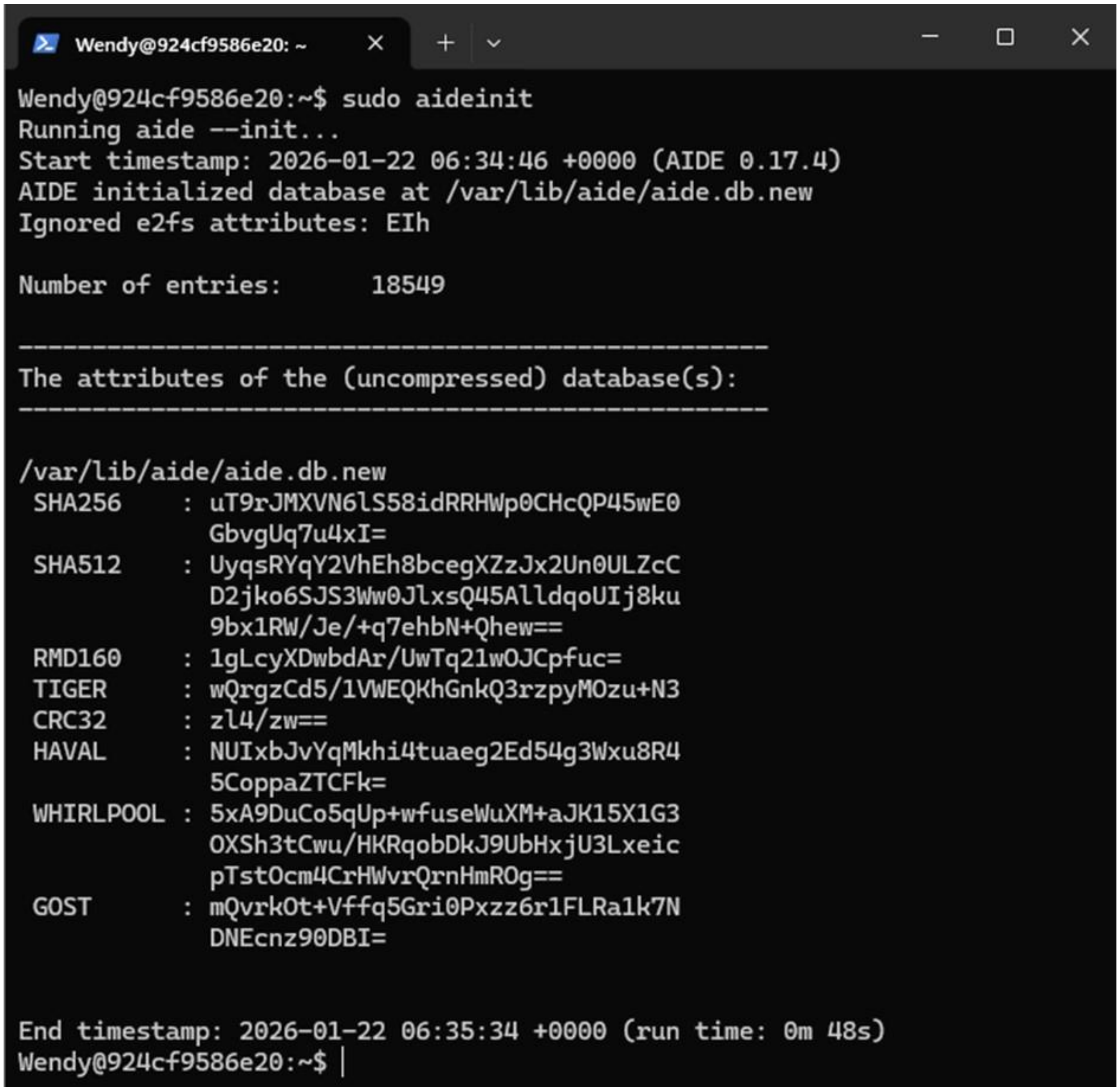
```
Wendy@924cf9586e20:~$ sudo nano /etc/login.defs
Wendy@924cf9586e20:~$ grep -nE '^UMASK' /etc/login.defs
150:UMASK                027
Wendy@924cf9586e20:~$
```



```
Wendy@924cf9586e20:~$ ls -l /etc/rsyslog.conf
-rw-r--r-- 1 root root 1382 Dec 23  2021 /etc/rsyslog.conf
Wendy@924cf9586e20:~$
```



```
Wendy@924cf9586e20:~$ ls -l /etc/fail2ban/jail.conf
-rw-r--r-- 1 root root 25071 Mar 10  2022 /etc/fail2ban/jail.conf
Wendy@924cf9586e20:~$ ls -l /etc/fail2ban/jail.d || true
total 4
-rw-r--r-- 1 root root 22 Mar 10  2022 defaults-debian.conf
Wendy@924cf9586e20:~$ ps aux | grep -i fail2ban | head -n 5
Wendy    67011  0.0  0.0   3472  1920 pts/0    S+   06:31   0:00 grep --
color=auto -i fail2ban
Wendy@924cf9586e20:~$
```



Wendy@924cf9586e20: ~

```
Wendy@924cf9586e20:~$ sudo aideinit
Running aide --init...
Start timestamp: 2026-01-22 06:34:46 +0000 (AIDE 0.17.4)
AIDE initialized database at /var/lib/aide/aide.db.new
Ignored e2fs attributes: EIh

Number of entries:      18549

---------------------------------------------------
The attributes of the (uncompressed) database(s):
---------------------------------------------------

/var/lib/aide/aide.db.new
 SHA256    : uT9rJMXVN6lS58idRRHWp0CHcQP45wE0
             GbvgUq7u4xI=
 SHA512    : UyqsRYqY2VhEh8bcegXZzJx2Un0ULZcC
             D2jko6SJS3Ww0JlxsQ45AlldqoUIj8ku
             9bx1RW/Je/+q7ehbN+Qhew==
 RMD160    : 1gLcyXDwbdAr/UwTq21wOJCpfuc=
 TIGER     : wQrgzCd5/1VWEQKhGnkQ3rzpyMOzu+N3
 CRC32     : zl4/zw==
 HAVAL     : NUIxbJvYqMkhi4tuaeg2Ed54g3Wxu8R4
             5CoppaZTCFk=
 WHIRLPOOL : 5xA9DuCo5qUp+wfuseWuXM+aJK15X1G3
             OXSh3tCwu/HKRqobDkJ9UbHxjU3Lxeic
             pTstOcm4CrHWvrQrnHmROg==
 GOST      : mQvrkOt+Vffq5Gri0Pxzz6r1FLRa1k7N
             DNEcnz90DBI=


End timestamp: 2026-01-22 06:35:34 +0000 (run time: 0m 48s)
Wendy@924cf9586e20:~$
```