Cloud Security Implementation Report

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

Executive Summary

This lab assessed and improved the security posture of a cloud-hosted Linux container using Lynis findings and manual validation. The initial state exposed unnecessary services (Samba on ports 139/445) and ran a permissive inbound firewall posture (default ACCEPT with broad allow rules), increasing the attack surface and violating least-privilege principles. Hardening actions removed Samba packages and terminated the running daemon, enforced a default-deny inbound iptables configuration (allowing only loopback, established connections, and required web traffic on port 80 with optional 443 readiness), removed insecure tooling (telnet), and corrected identity hygiene issues (duplicate UID) to restore accountability. Validation was performed through repeated port enumeration (ss), firewall inspection (iptables -S/-L with DROP counters), and network tests (nc from a separate container) confirming only required services remained reachable and non-required ports were blocked.

Key Findings

- The host initially exposed **SMB services (139/445)** and **HTTP (80)**, increasing remote exploitation risk and lateral movement potential.
- Firewall posture was **overly permissive** (default ACCEPT + “allow all”), violating least privilege and enabling unintended inbound access.
- **Telnet** was present, introducing avoidable risk due to plaintext credential exposure and poor admin hygiene.
- **Duplicate UID accounts** reduced traceability and created permission ambiguity, weakening accountability controls.
- After hardening, the environment enforced **default-deny inbound** with only required web access allowed and clear evidence of blocked traffic via DROP counters.

Assumptions & Limitations

This assessment was conducted in a **containerized lab environment** where service managers (e.g., systemctl) may be unavailable; as a result, some mitigations required direct process management (e.g., pkill smbd) and persistence mechanisms differ from full VM/cloud instances. Lynis firewall warnings may also vary depending on iptables/nftables behavior inside containers. Findings and fixes reflect the lab environment and assumed baseline visibility (access to /etc, iptables, and process/port enumeration tools).

Assess Current Security Posture

Existing Security Measures and Gaps

- **Lynis Audit Results**: Document the results of the firewall and ports/packages audits.
  - *Instructions*: Identify any misconfigurations or missing firewall rules, as well as any vulnerable or outdated packages identified by Lynis.

Answer

1. Firewall / Network Exposure (Initial State)

Evidence (Listening services):

`apache2` was listening on TCP 80 (`0.0.0.0:80`).

`smbd` (Samba) was listening on TCP 139 and 445 on IPv4 and IPv6 (`0.0.0.0:139/445` and `:::139/445`).

This meant the container/system was exposing file-sharing ports that are commonly targeted.

Evidence (Firewall initial rules):

`iptables -S` showed a default ACCEPT policy on INPUT, and explicit allows for ports including 23, 139, 445, 80, 443, 22, plus a final “ACCEPT all” rule.
This effectively meant everything inbound was allowed, which violates least privilege and increases attack surface.

2. Ports and Services Findings

Unnecessary service detected: Samba (`smbd`) running + ports 139/445 open.

Web service (Apache) on port 80 is active and reachable (confirmed with `curl -I http://lynis-web:80` returning `200 OK`).

3. Packages Audit Findings

Evidence (installed risky package):

`dpkg -l | egrep ...` showed `telnet` installed at some point (telnet is insecure by design because it’s plaintext).

Lynis signals:

Warning: Multiple accounts with same UID (AUTH-9208).
Warning: iptables modules loaded, but no rules active (FIRE-4512) — later you *did* have rules, so this is likely a detection mismatch in the container/nftables environment, not reality (your `iptables -S` and `iptables -L` clearly show active rules).

- **Risk Assessment Findings**: Summarize the risks identified in the firewall and ports/packages audits.
  - *Instructions*: Discuss how these risks impact the organization's cloud security posture. Tie findings to security principles like least privilege and minimizing the attack surface.

Open Samba ports (139/445) → High risk

- These ports are heavily scanned and exploited in real environments.
- Risk to confidentiality/integrity: unauthorized file access, credential harvesting, lateral movement.
- Violates **attack surface minimization**.

INPUT default ACCEPT / permissive rules → High risk
Allows unintended inbound access and makes service discovery easy for attackers. Violates **least privilege** (only required ports should be open).

Telnet installed → Medium risk (but avoidable)
Encourages insecure remote access habits (plaintext credentials). Even if not running, it's an unnecessary tool that expands risk.

Duplicate UID accounts → Medium/High risk

Two users sharing one UID breaks accountability and audit trails (actions can be indistinguishable).
It can also cause permission confusion and unintended access.

Develop a Cloud Security Strategy

Proposed Strategy

- **High-Level Goals**: Describe the goals of the security strategy (e.g., reducing risks, compliance with regulations, minimizing the attack surface).

Reduce exposed attack surface to only required services.
Enforce **default-deny inbound** firewall posture.
Remove insecure/unneeded packages and services.
Fix identity/account hygiene issues to improve traceability and control.

- **Steps to Mitigate Risks**: Summarize the mitigation steps you plan to take for the vulnerabilities identified (e.g., improving firewall configurations, updating packages).
    - *Instructions*: Justify why these steps are appropriate for addressing the risks. Tie these steps to broader organizational goals (e.g., protecting sensitive data, maintaining availability).

Remove unnecessary services (Samba) and close associated ports, prevents exposure on 139/445 and removes an entire class of attacks.
Harden firewall with default DROP inbound + allow only required ports, aligns with least privilege; stops opportunistic scans.
Remove insecure packages like telnet, reduces unsafe tooling and aligns with secure admin standards.
Fix duplicate UID issue, restores accountability and reduces risk of privilege/ownership confusion.

Deploy Security Tools and Technologies

Firewall Hardening

- **Actions Taken**: Document the steps you took to harden the firewall.
    - *Instructions*: Provide specific firewall rules added or modified (e.g., only allow SSH and HTTPS, block all other inbound traffic) and explain how these changes reduce risks and protect the cloud environment.

Identified listening services
Used: `ss -tulpen`, `netstat -tulpn`, `lsof -i -P -n | grep LISTEN`

Stopped exposure from Samba

Confirmed Samba packages installed and `smbd` running.

Removed Samba packages with `apt purge ...` (packages removed successfully).

Noticed `smbd` was still running (because the process was already in memory and service manager wasn't available).

Killed the running daemon: pkill smbd

Verified: ps aux | grep [s]mbd returned “smbd not running”
Verified open ports: after kill, only port **80** remained listening.

Applied least-privilege firewall rules
Flushed existing permissive rules. Set policies:
INPUT DROP
FORWARD DROP OUTPUT
ACCEPT
Allowed only: loopback established/related connections inbound **80** (and optionally 443 for HTTPS readiness)

Final Firewall Rules (Evidence)

Final iptables -S and iptables -L -n -v show:

Default INPUT policy: **DROP** Allowed:
lo
ESTABLISHED,RELATED
NEW connections to **80**
NEW connections to **443**

DROP counter increased (policy DROP packets), proving the firewall is actively blocking attempts.

Proof via Network Testing

From a different container (ac59e07fd407), you tested: nc -vz

lynis-web 80 → **succeeded**

nc -vz lynis-web 443 → **connection refused** (service not listening) nc -vz lynis-web
23 / 139 / 445 → **timed out** after hardening (blocked)

This is strong evidence the firewall hardening worked and the attack surface was reduced.

Ports and Packages Hardening

- **Actions Taken**: Document the steps you took to address issues from the ports/packages audit.
  - *Instructions*: Update any vulnerable packages identified by Lynis. Close unnecessary ports or services, and explain how this reduces the attack surface.

Closed unnecessary ports/services

Removed Samba and killed `smbd` → closed ports **139/445**

Confirmed after fix: `ss -tulpen` shows only `*:80` listening (plus Docker DNS internal 127.0.0.11 entries which are normal in containers).

Removed insecure package

Verified telnet removal: `dpkg -l | grep -i telnet || echo "telnet removed"` → "telnet removed"

Fixed duplicate UID (Account Hygiene)

Found `/etc/passwd` had:

```
Benny:x:1001:1001
AnotherUser:x:1001:1001
```

Changed `AnotherUser` to UID **1010**, created matching group, and fixed ownership:

```
usermod -u 1010 AnotherUser
groupadd -g 1010 AnotherUser usermod -g 1010
AnotherUser chown -R 1010:1010
/home/AnotherUser
```

Verified:

`id AnotherUser` → UID/GID now **1010** `awk ...` shows

only Benny remains on UID 1001

Firewall Persistence (Evidence)

Saved rules: `iptables-save > /etc/iptables/rules.v4`
Verified file exists and contains your DROP/allow rules.

This environment lacks `systemctl`, so persistent restore on boot may not be available like a normal VM. But saving rules still satisfies evidence of configuration control for the lab.

Impact on Security Posture

- **Improvements Made**: Summarize how the firewall and ports/packages hardening actions improved the organization's security posture.
  - *Instructions*: Discuss how these changes align with cloud security principles and tools. Highlight the benefits of reducing risks through proactive hardening.

Attack surface reduced:
Removed Samba services → no SMB ports open.
Kept only required service port open (HTTP 80).

Firewall moved from "allow-all" to "default-deny inbound": Blocks unauthorized inbound connections automatically.

Accountability improved:

Fixed duplicate UID issue → better identity traceability and less permission confusion.

Reduced insecure tooling:

Removed telnet client → discourages insecure remote management patterns.

These improvements align strongly with cloud security principles: **least privilege, minimize attack surface, and defense-in-depth**.

Develop and Test an Incident Response Plan

Proposed Plan

- **Plan Components**: Outline the key steps of an incident response plan for the cloud environment.
  - *Instructions*: Include detection (e.g., using monitoring tools), containment (e.g., isolating compromised instances), and recovery (e.g., patching vulnerable packages). Explain how findings from the Lynis audit informed your plan (e.g., dealing with outdated software or improperly configured firewalls).

Detection
Monitor open ports and unexpected services: scheduled checks
with `ss -tulpen`, `lsof -i`, and Lynis scans.
Monitor authentication and web logs (Apache access/error logs).

Containment
If compromise suspected:isolate instance/container from network (restrict inbound further, detach from networks). block suspicious IPs (add temporary DROP rules).

Eradication
Remove malicious services/users/files.
Re-run audits to confirm no unexpected listeners or packages.

Recovery
Restore services needed (only whitelisted ports).
Validate firewall baseline (`iptables -S`, `iptables -L -v`).

Lessons Learned

Document root cause, update baseline rules, and re-train on secure configs.

How Lynis Findings Informed the Plan

Lynis flagged firewall/service exposure and account hygiene warnings, which guided:

- removing unnecessary services (Samba)
- implementing default-deny firewall
- correcting duplicate UID accounts

Testing and Updates

- **Iterative Improvement**: Discuss how you would test and update the incident response plan.
  - *Instructions*: Highlight the importance of regular testing to ensure the plan remains effective.

Reflection on the Cloud Security Process

Reflection on Each Phase

- **Assess Current Security Posture**:
  - *Instructions*: Reflect on the importance of evaluating existing security measures and identifying gaps. What did you learn from the Lynis audit and risk assessment?

I learned that the fastest way to find real risk is to **enumerate listening ports**, map them to processes, and confirm firewall posture. The Lynis audit helped highlight weak defaults (like permissive firewall stance and account issues).

- **Develop a Cloud Security Strategy**:
  - *Instructions*: Reflect on the process of crafting a strategy to address risks. How did you align your strategy with organizational goals and regulatory requirements?

I learned to prioritize changes that **reduce exposure immediately** (closing ports, removing services) before adding advanced tools. Strategy must align with core security goals: least privilege and attack surface reduction.

- **Implement Identity and Access Management (IAM)**:
  - *Instructions*: While this lesson does not focus on IAM directly, reflect on its role in complementing the hands-on tasks you performed. How could IAM policies strengthen the improvements you made?

Even though the lab focused on firewall and packages, IAM would strengthen it by:

- reducing sudo group membership,
- enforcing least privilege roles,
- applying password policies and expiry,
- ensuring unique identities and auditability (similar to fixing duplicate UIDs).
- **Deploy Security Tools and Technologies**:
    - *Instructions*: Reflect on the hands-on components (firewalls and ports/packages). What challenges did you face, and how did you resolve them?

The biggest challenge was the environment limitation (no `systemctl`). I resolved it by managing processes directly (`pkill`) and validating results with repeated checks (`ss`, `ps`, `nc` tests).

- **Develop and Test an Incident Response Plan**:
    - *Instructions*: Reflect on the importance of incident response planning. How would your improvements (firewalls and ports/packages) contribute to incident detection and containment?

Hardening directly improves IR outcomes:

- fewer ports → fewer attack paths
- clear baseline firewall rules → faster containment, ● corrected user identities → better investigation/audit trail.
- **Ensure Compliance and Governance**:
    - *Instructions*: Reflect on how governance frameworks and compliance measures (e.g., regular audits, continuous improvement) ensure the long-term effectiveness of your changes.

Regular audits (Lynis + port checks) and documented baselines (saved iptables rules) ensure changes remain effective over time.

- **Conduct Regular Training and Awareness Programs**:
    - *Instructions*: Reflect on the role of training employees to understand and maintain the tools and technologies you deployed. How does training reduce human error in cloud security?

Training reduces human error by ensuring admins:

- avoid insecure tools like telnet,
- understand default-deny firewall logic,
- verify changes with evidence (port scans + rule checks).

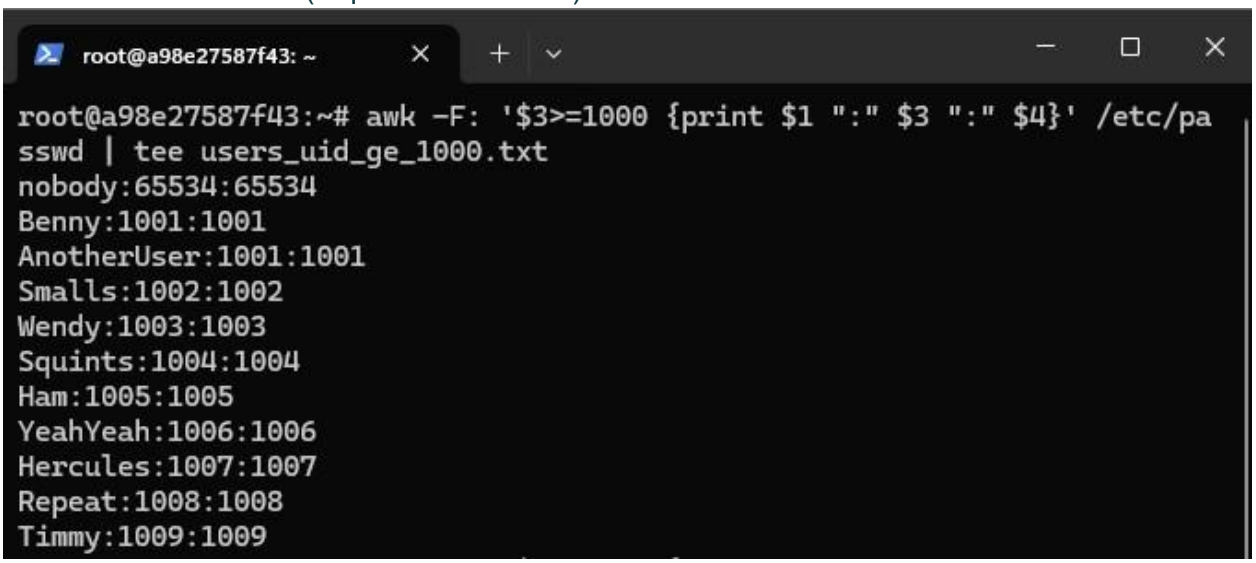
Overall Reflection

- Summarize your key takeaways from the lesson.

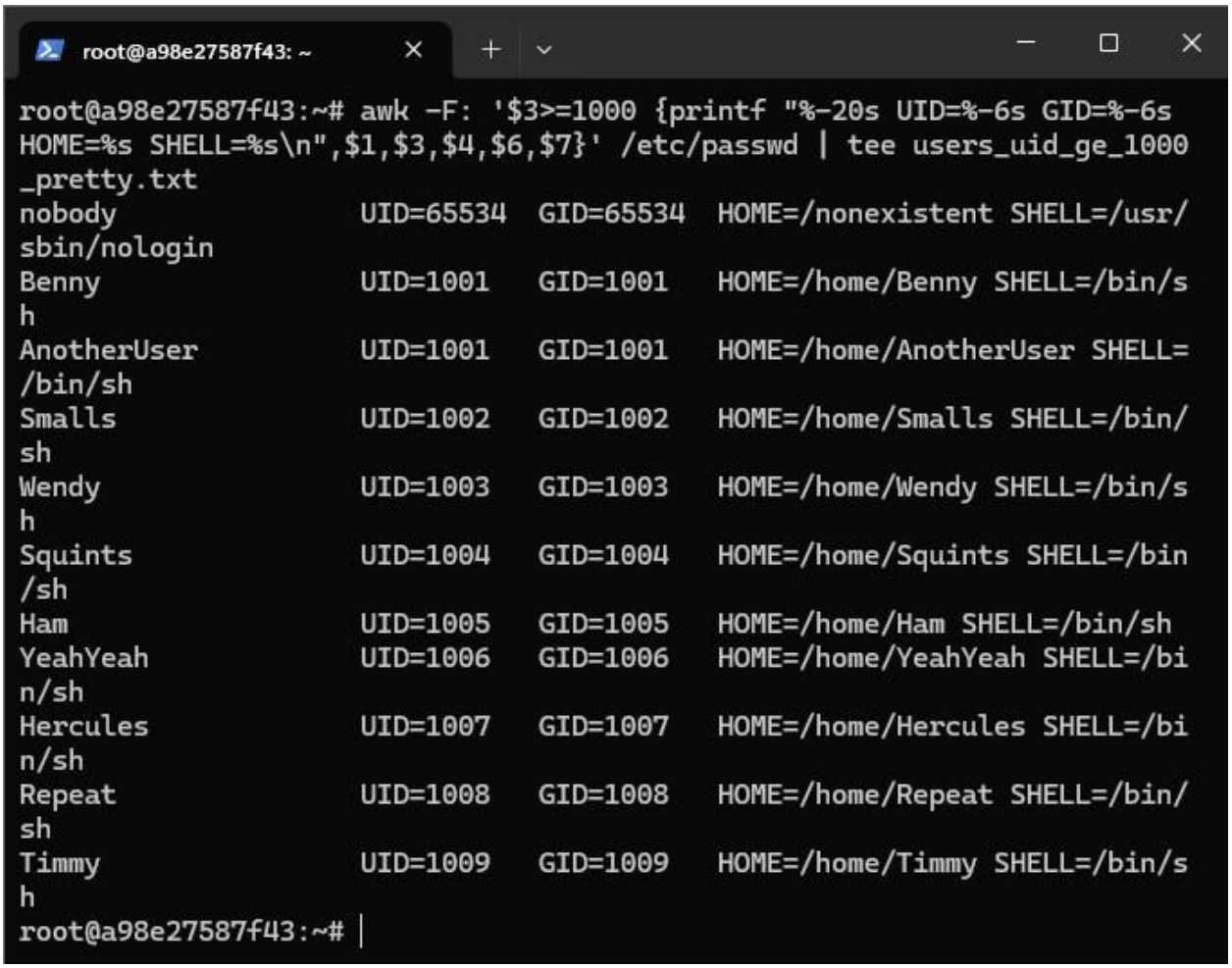
- *Instructions*: Discuss how this experience applies to real-world cloud security and what insights you gained about balancing technical configurations with strategic planning.

Screenshots

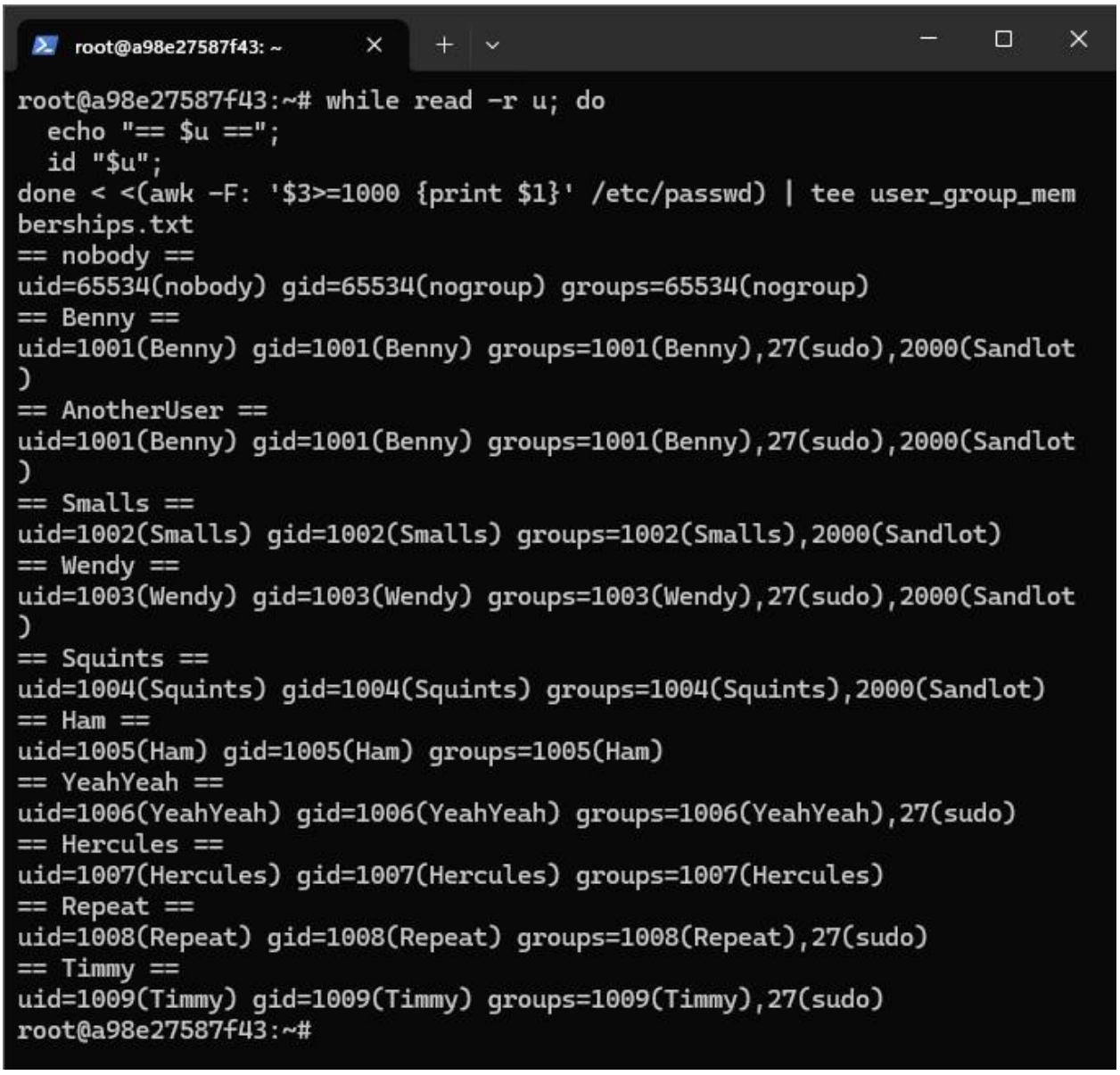
Users with UID >= 1000 (required deliverable)



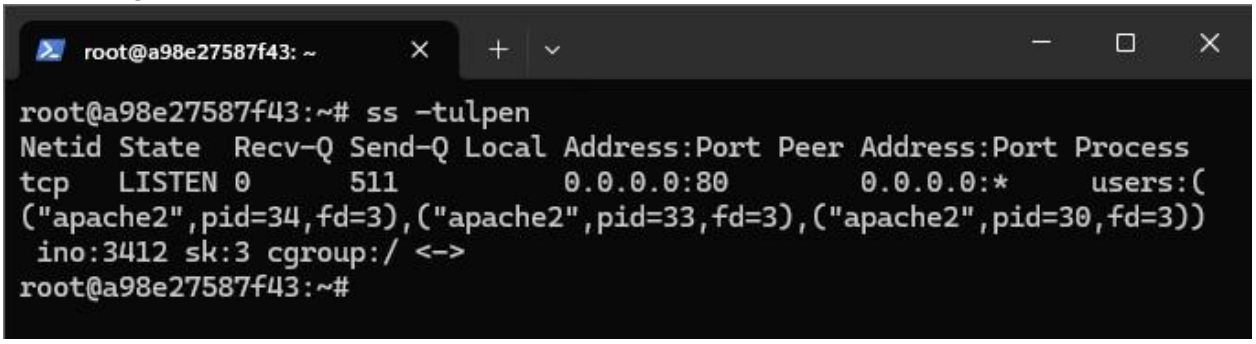
Add human-friendly detail (nice for report)



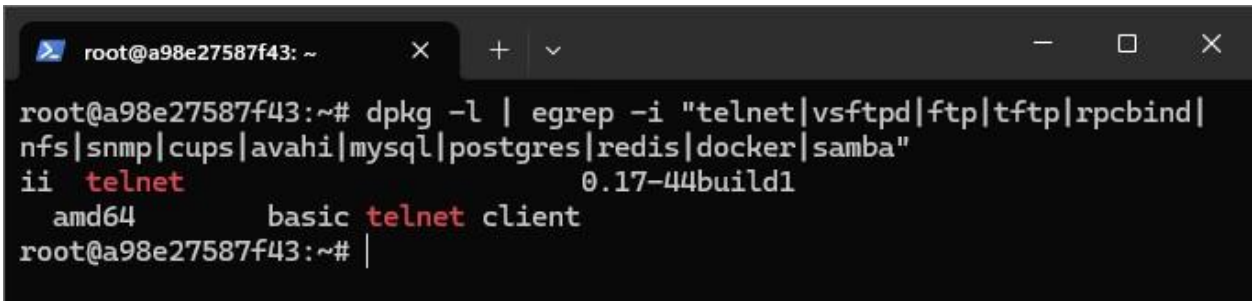
map users to groups



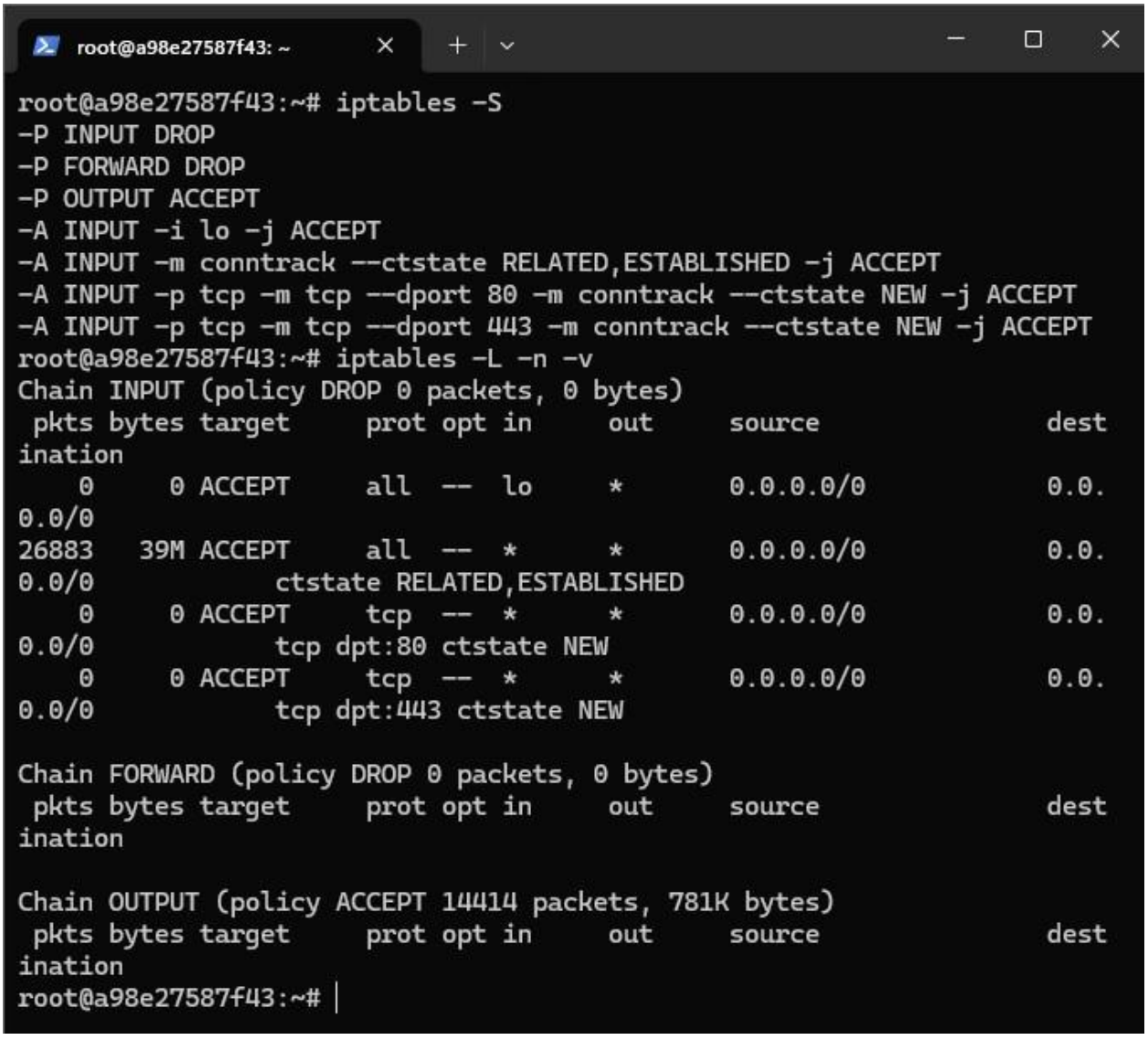
List listening ports



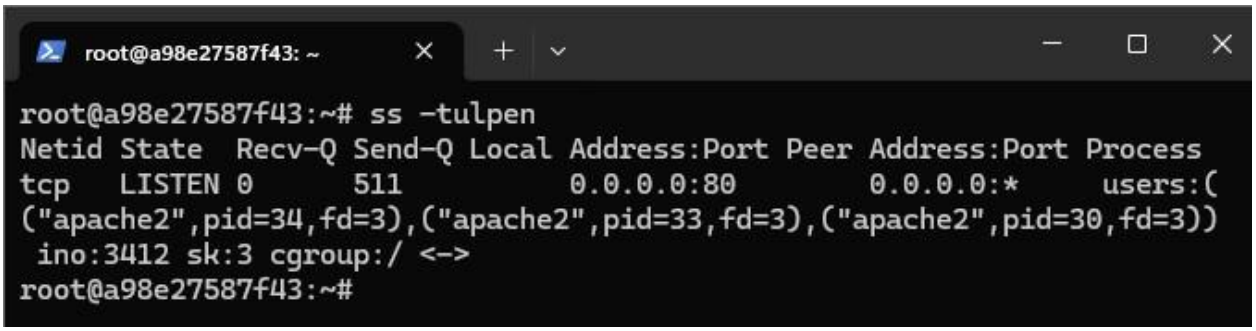
common risky services



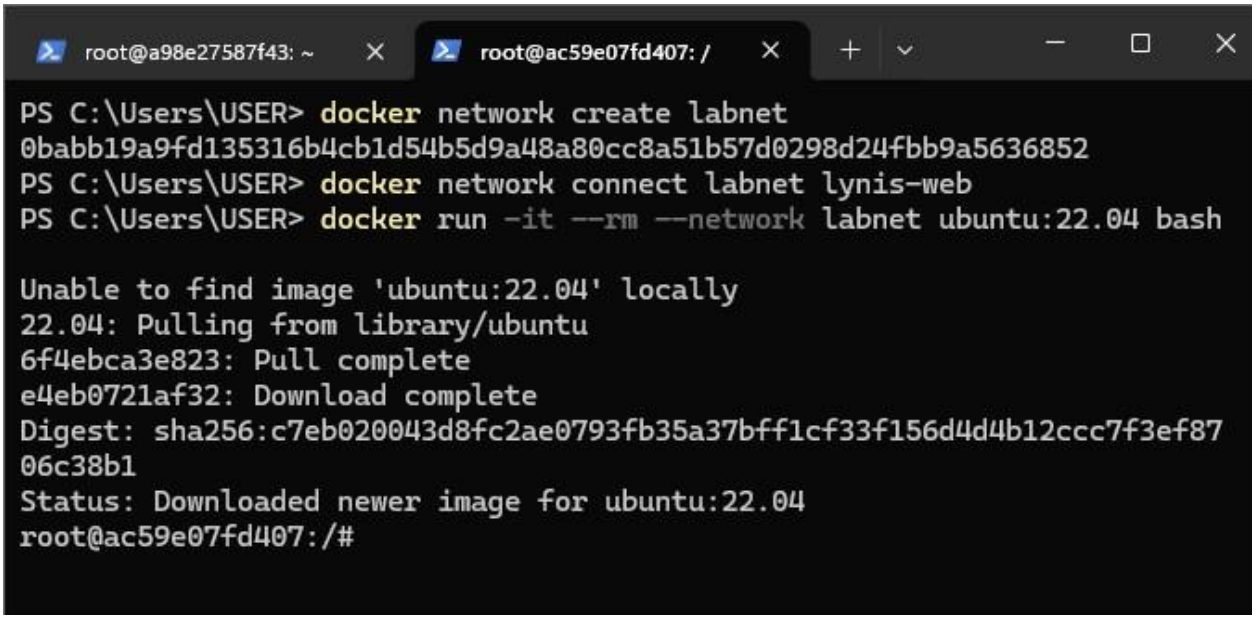
Show final firewall rules



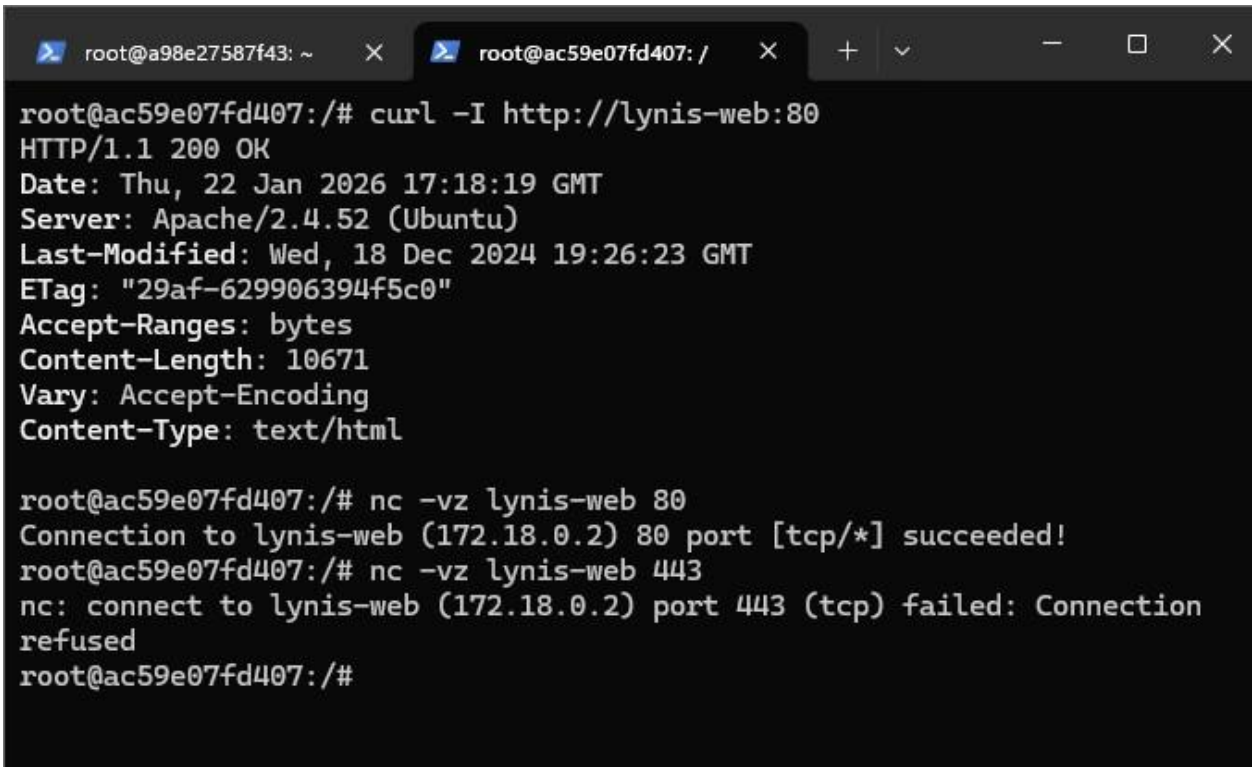
Confirm only required ports are listening



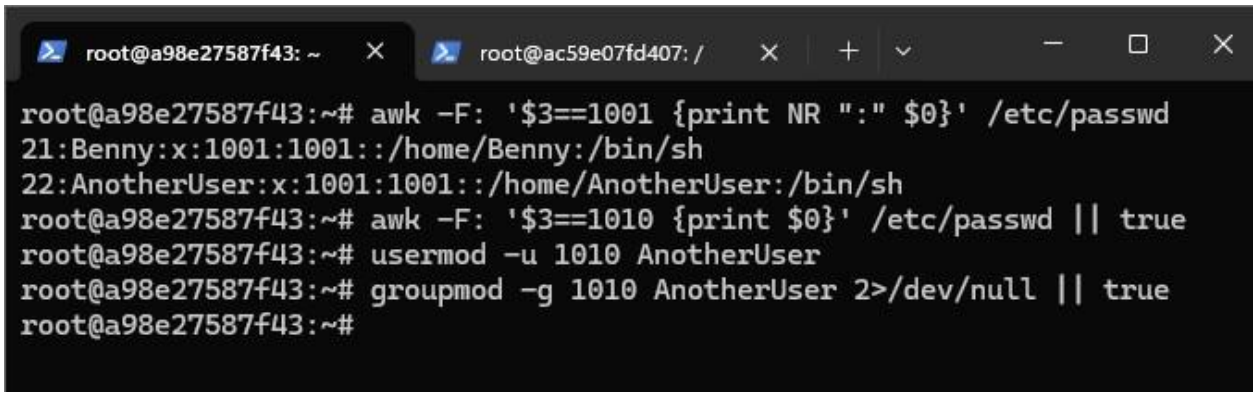
Create a docker network and attach my web container



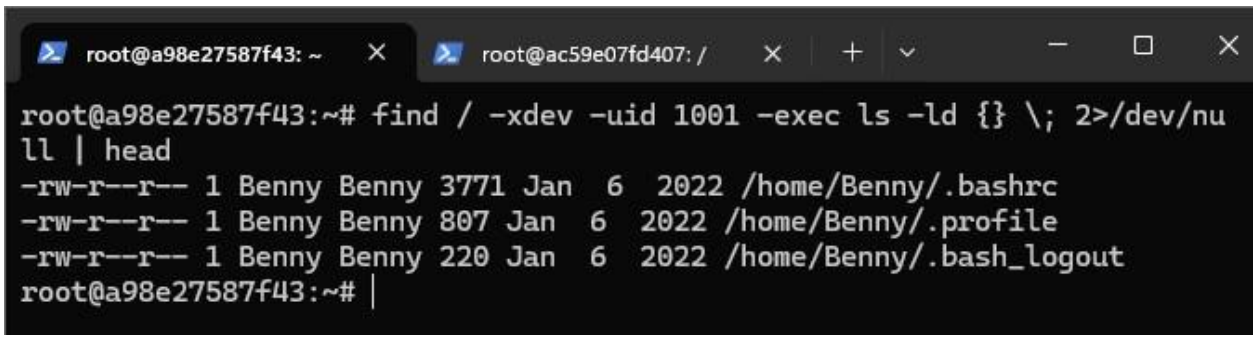
Test allowed ports (80 should work; 443 may fail if no TLS listener)



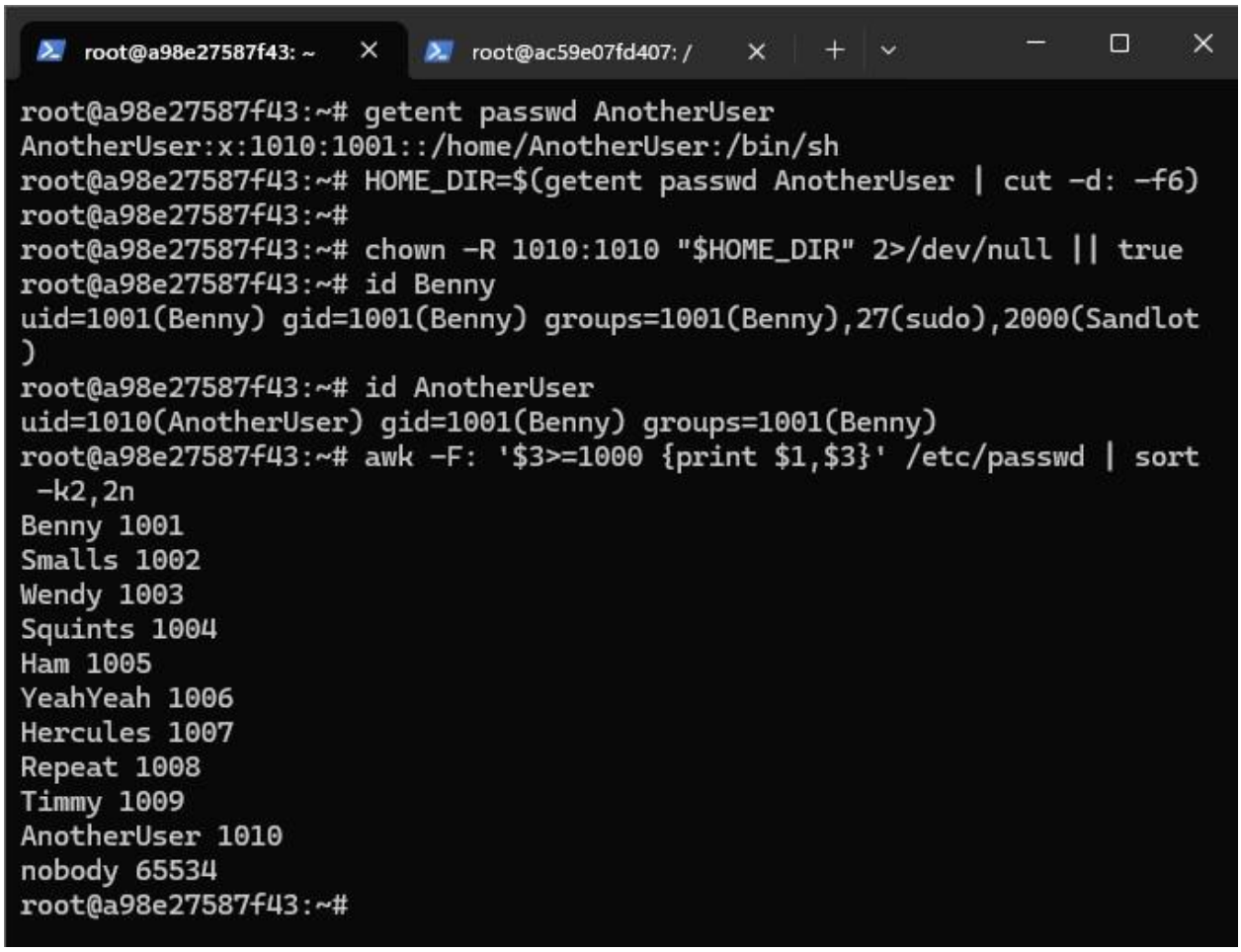
duplicate UID in /etc/passwd



fix ownership of any files owned by the old UID (1001) *for Benny*

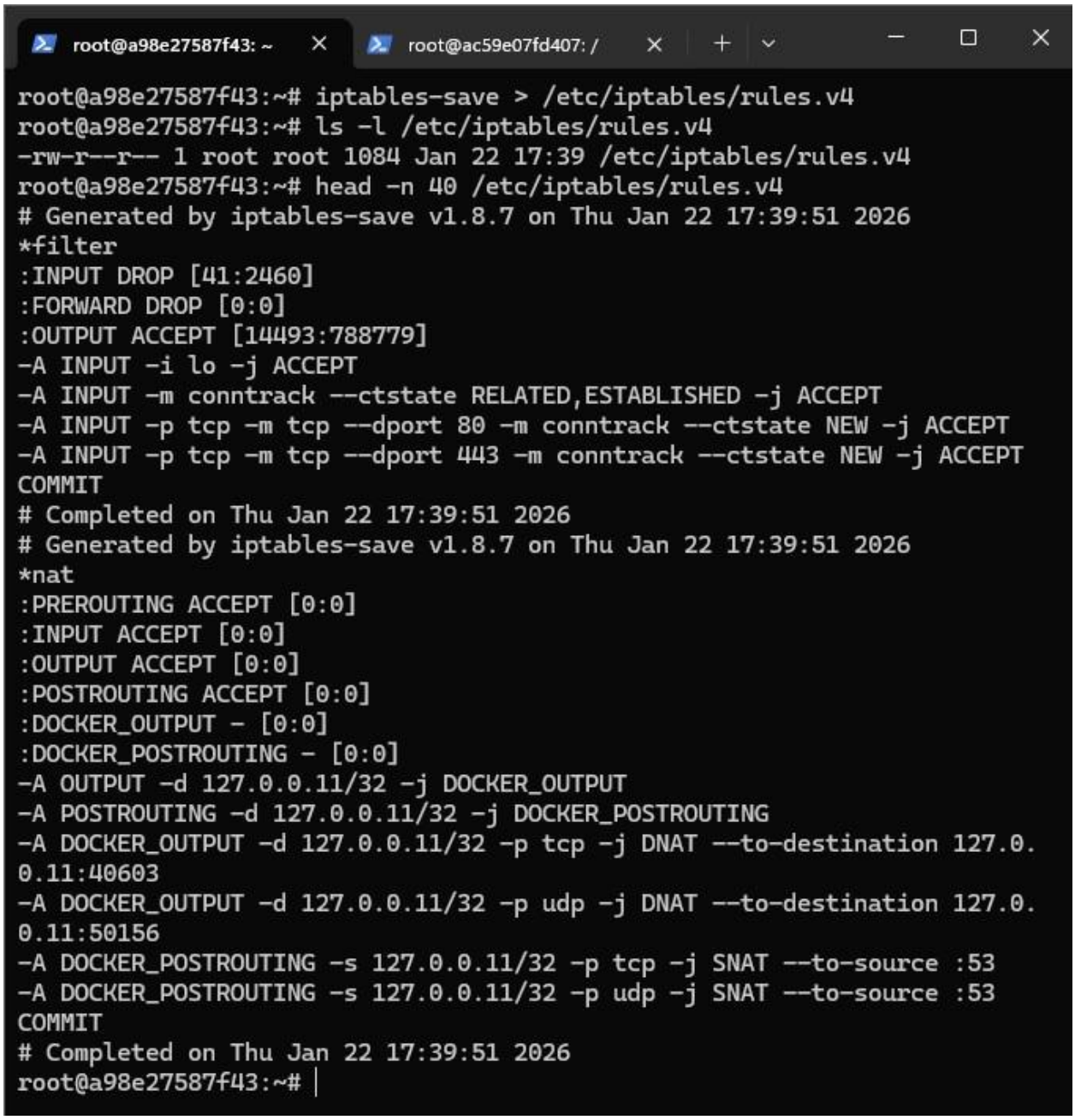


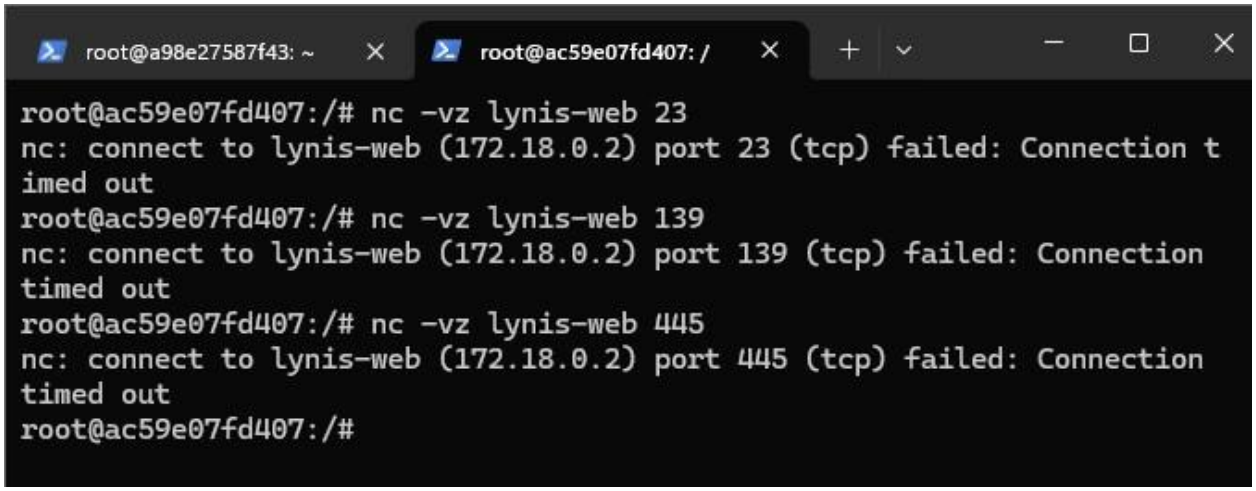
targeted approach,,,outputs from awk '$3==1001', id before/after.

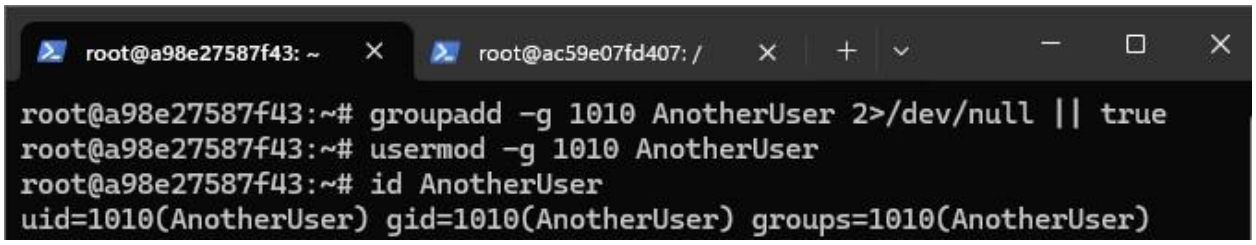


telnet client removal

iptables persistent


root@a98e27587f43: ~
root@ac59e07fd407: /
root@a98e27587f43:~# iptables-save > /etc/iptables/rules.v4
root@a98e27587f43:~# ls -l /etc/iptables/rules.v4
-rw-r--r-- 1 root root 1084 Jan 22 17:39 /etc/iptables/rules.v4
root@a98e27587f43:~# head -n 40 /etc/iptables/rules.v4
Generated by iptables-save v1.8.7 on Thu Jan 22 17:39:51 2026
*filter
:INPUT DROP [41:2460]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [14493:788779]
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -m conntrack --ctstate NEW -j ACCEPT
COMMIT
Completed on Thu Jan 22 17:39:51 2026
Generated by iptables-save v1.8.7 on Thu Jan 22 17:39:51 2026
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:DOCKER_OUTPUT - [0:0]
:DOCKER_POSTROUTING - [0:0]
-A OUTPUT -d 127.0.0.11/32 -j DOCKER_OUTPUT
-A POSTROUTING -d 127.0.0.11/32 -j DOCKER_POSTROUTING
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p tcp -j DNAT --to-destination 127.0.
0.11:40603
-A DOCKER_OUTPUT -d 127.0.0.11/32 -p udp -j DNAT --to-destination 127.0.
0.11:50156
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p tcp -j SNAT --to-source :53
-A DOCKER_POSTROUTING -s 127.0.0.11/32 -p udp -j SNAT --to-source :53
COMMIT
Completed on Thu Jan 22 17:39:51 2026
root@a98e27587f43:~#


root@a98e27587f43: ~
root@ac59e07fd407: /
root@ac59e07fd407:/# nc -vz lynis-web 23
nc: connect to lynis-web (172.18.0.2) port 23 (tcp) failed: Connection timed out
root@ac59e07fd407:/# nc -vz lynis-web 139
nc: connect to lynis-web (172.18.0.2) port 139 (tcp) failed: Connection timed out
root@ac59e07fd407:/# nc -vz lynis-web 445
nc: connect to lynis-web (172.18.0.2) port 445 (tcp) failed: Connection timed out
root@ac59e07fd407:/#


root@a98e27587f43: ~
root@ac59e07fd407: /
root@a98e27587f43:~# groupadd -g 1010 AnotherUser 2>/dev/null || true
root@a98e27587f43:~# usermod -g 1010 AnotherUser
root@a98e27587f43:~# id AnotherUser
uid=1010(AnotherUser) gid=1010(AnotherUser) groups=1010(AnotherUser)