Cybersecurity Home Lab Documentation

Internal Runbook / Engineering Playbook

Platform: Proxmox VE
Primary VM: Ubuntu Server 24.04 LTS
Security Stack: Wazuh SIEM (Manager, Indexer, Dashboard)
Author: Daniel Lopez
Purpose: Internal operational documentation and reproducibility

---

1. Purpose of This Document

This runbook documents the **design, deployment, configuration, troubleshooting, and validation** of a personal cybersecurity home lab built to simulate a **small-scale Security Operations Center (SOC)** environment.

The intent of this document is to:

- Capture **what actions were taken**, **why they were taken**, and **how issues were resolved**
- Serve as an **internal operational reference** rather than a tutorial
- Enable reproducibility by another engineer with similar infrastructure
- Demonstrate applied skills in **Linux administration, virtualization, networking, and SIEM operations**

This document intentionally includes:

- Design rationale
- Failure scenarios
- Validation steps
- Operational commands
- Lessons learned

---

2. Lab Objectives

The lab was built to meet the following objectives:

- Deploy and manage a virtualized Linux server using an enterprise-style hypervisor
- Configure stable networking suitable for security monitoring
- Install and operate a SIEM stack capable of log ingestion and alerting

- Practice authentication troubleshooting and service recovery
- Validate system functionality using real command-line verification
- Document the environment in a professional, SOC-aligned format

3. High-Level Architecture Overview

Architecture Summary

- **Hypervisor:** Proxmox VE running on a local host
- **Virtual Machines:**
  - Ubuntu Server VM acting as a centralized SIEM server
- **Security Stack:**
  - Wazuh Manager
  - Wazuh Indexer
  - Wazuh Dashboard
- **Networking:**
  - Bridged networking using `vmbr0`
  - Static IP assigned to SIEM VM

Architecture Diagram (Placeholder)

`[ Proxmox Host ] | vmbr0 | [ Ubuntu Server VM ] | [ Wazuh Manager / Indexer / Dashboard ]`

> **Placeholder:** Insert architecture diagram screenshot or exported diagram
> **File:** `architecture_overview.png`

4. Chronological Build Timeline

Phase 1 — Proxmox Access & Baseline Validation

Timeframe: Early setup phase
Goals:

- Gain access to Proxmox web interface
- Resolve login warnings and kernel update messages
- Establish a stable hypervisor baseline before VM creation
  Key Observations:
- Proxmox displayed a non-subscription warning
- Kernel update notification required acknowledgment

- These warnings were informational and did not block functionality

Why This Matters:
SOC infrastructure often runs without paid support. Knowing which warnings are critical vs informational is essential for operational judgment.

Phase 2 — Network Design & Bridging Decisions

Goals:

- Ensure VMs could communicate with the LAN
- Avoid NAT-based isolation
- Prepare for SIEM log access and future agent connectivity
  Actions Taken:
- Selected `vmbr0` as the primary bridge
- Verified bridge configuration via CLI
  Verification Example:
  `ip a cat /etc/network/interfaces`
  Why Bridged Networking Was Chosen:
- Enables direct LAN visibility
- Mimics real SOC server placement
- Simplifies static IP assignment
- Supports future expansion (agents, sensors)

Phase 3 — Ubuntu Server VM Creation

Goals:

- Create a stable, performance-appropriate SIEM VM
- Avoid overcommitment and misconfiguration
  Key Configuration Decisions:

Component	Decision
BIOS	OVMF (UEFI)
Machine Type	q35
CPU Type	host
Sockets	1
Cores	6
Memory	10 GB
Ballooning	Disabled

Component	Decision
Disk	SCSI (VirtIO), 100 GB
Network	VirtIO on vmbr0
QEMU Agent	Enabled

Why These Choices Matter:

- SIEM workloads are memory-intensive
- Ballooning can destabilize log indexing
- VirtIO improves disk and network performance
- QEMU agent enables better VM visibility and control

Phase 4 — Ubuntu Server Installation

Goals:

- Deploy a minimal, secure Linux environment
- Avoid unnecessary packages
- Retain full control over installed services
  Decision:
- Chose **Ubuntu Server (not minimized)** to retain flexibility during troubleshooting
  Why This Matters:
  Minimized installs reduce footprint but can slow down debugging when tools are missing. For a learning-focused SOC lab, transparency was prioritized over minimalism.

---

5. Initial Post-Install Validation

Immediate Checks Performed:
`ip a lsb_release -a systemctl status ssh`
Goals:

- Confirm network connectivity
- Confirm OS version
- Verify remote access capability

---

6. Static IP Configuration & Network Stability

Problem Identified

- Ubuntu VM initially received a **dynamic IP**

- SIEM services require stable addressing
- Dashboard and API endpoints broke across reboots

Resolution

- Static IP configured using Netplan
- Gateway and DNS explicitly defined
  Verification Example:
  `ip a ip route`
  Why This Matters in SOC Environments:
  SIEM components rely on consistent endpoints. Dynamic addressing introduces avoidable operational risk.

---

7. Snapshot Strategy (Stability Milestone)

Once:

- Networking stabilized
- SSH access confirmed
- Base OS validated
  A **Proxmox snapshot** was taken.
  Purpose of Snapshot:
- Create a rollback point before SIEM installation
- Preserve a known-good infrastructure state
- Enable rapid recovery during experimentation

8. Wazuh SIEM Deployment

Objective

Deploy a fully functional **Wazuh SIEM stack** on the Ubuntu Server VM, consisting of:

- Wazuh Manager
- Wazuh Indexer
- Wazuh Dashboard
  The goal was to establish centralized log ingestion, indexing, and visualization within a single-node SOC-style deployment.

Installation Approach

Deployment Model:

- Single-node Wazuh stack (Manager + Indexer + Dashboard co-located)
  Reasoning:
- Reduces operational complexity during initial learning
- Common pattern for small SOCs and PoC environments
- Easier troubleshooting compared to distributed deployments

> **Note:** This mirrors how engineers often prototype before scaling to multi-node architectures.

---

9. Service Initialization & Early Failures

Initial Symptoms Observed

After installation:

- Wazuh Manager failed to restart properly
- Authentication attempts against the Wazuh API returned `Unauthorized`
- Dashboard access was inconsistent
  Observed Error Example:
  `{"title":"Unauthorized","detail":"Invalid credentials"}`

Services Involved

Service	Purpose
`wazuh-manager`	Log processing and rule engine
`wazuh-indexer`	Data indexing and search
`wazuh-dashboard`	Visualization and UI
Wazuh API	Authentication and management

---

10. Troubleshooting Methodology

Rather than reinstalling blindly, a structured troubleshooting approach was used.

Step 1 — Service Status Inspection

```
systemctl status wazuh-manager journalctl -xeu wazuh-manager
```
Why This Matters:
SOC engineers rely on systemd logs to diagnose startup failures rather than guessing root causes.

Step 2 — Configuration File Validation

```
sudo cat /usr/share/wazuh-dashboard/data/wazuh/config/wazuh.yml
```

Goal:

- Confirm authentication configuration
- Validate user and API credential references
- Ensure file integrity post-install

Step 3 — Credential Verification via API

```
curl -k -u 'wazuh-wui:<password>' \
https://127.0.0.1:55000/security/user/authenticate?raw=true
```

Observed Result:

- Authentication failed initially
- Confirmed issue was **credential-related**, not service availability

---

11. Authentication & Indexer Role Fix

Root Cause Identified

- The Wazuh Indexer role configuration was incomplete
- Required permissions for log ingestion and access were missing

---

Corrective Action Taken

Updated the `logstash` role via the Indexer Security API.

```
curl -k -u "admin:<password>" -X PUT \
"https://127.0.0.1:9200/_plugins/_security/api/roles/logstash" \ -H 'Content-Type:
application/json'
```

Why This Worked:

- Restored proper role-based access
- Allowed index creation and data flow
- Unblocked SIEM ingestion pipeline

---

12. Validation of SIEM Functionality

Index Verification

`curl -k -u "admin:<password>" \ "https://127.0.0.1:9200/_cat/indices/wazuh-alerts-4.x-*?v"`

Successful Output Indicators:

- Index status: `green`
- Documents present
- Storage size increasing

Example (summarized):
`green open wazuh-alerts-4.x-YYYY.MM.DD`

Why This Step Is Critical

SOC engineers do not trust dashboards alone.
Indexer verification confirms:

- Logs are ingested
- Rules are firing
- Data persistence is functioning

---

13. Operational Stability Check

After fixing authentication and index roles:

- `wazuh-manager` restarted successfully
- Indexer maintained green health
- Dashboard access stabilized
- No recurring authentication failures observed
  Stability Verification Commands:
  `systemctl status wazuh-manager systemctl status wazuh-indexer systemctl status wazuh-dashboard`

---

14. Post-SIEM Snapshot (Critical Milestone)

Once:

- Services were stable
- Indexing was confirmed
- Authentication issues resolved
  A **Proxmox snapshot** was taken.

Purpose of This Snapshot

- Preserve a **known-good SIEM state**
- Enable safe experimentation with rules and agents
- Provide rapid rollback if future misconfigurations occur

> **SOC Best Practice:** Always snapshot after major service milestones.

---

15. Known Failure Scenarios (Documented)

Scenario	Impact	Resolution
Dynamic IP changes	Dashboard/API breakage	Static IP via Netplan
Invalid Wazuh credentials	Unauthorized API access	Credential verification
Missing indexer roles	No log ingestion	Role API update
Service restart loops	SIEM unavailable	systemd + journalctl analysis

---

16. Lessons Learned (Engineering Takeaways)

- Authentication issues are often **permission-based**, not service-based
- API testing is faster than UI-based debugging
- Static networking is mandatory for infrastructure services
- Snapshots save hours of recovery time
- Reading logs beats reinstalling software

---

17. Future Improvements (Planned Enhancements)

- Add Wazuh agents on additional VMs or hosts
- Enable file integrity monitoring (FIM)
- Configure alert forwarding
- Implement role separation (Manager vs Dashboard)
- Add dashboards tailored to SOC workflows

---

18. Day-to-Day Operations Reference

This section serves as a **quick-access operational guide**

18.1 Proxmox VM Operations

List VMs
`qm list`

Start / Stop VM

`qm start <VMID> qm shutdown <VMID> qm stop <VMID>`

Snapshot Management

`qm snapshot <VMID> <snapshot-name> qm rollback <VMID> <snapshot-name>`

Operational Notes:

- Snapshots taken only after verified stable states
- Avoid snapshot sprawl to reduce storage pressure

18.2 Ubuntu Server Core Checks

System Info

`lsb_release -a uname -r uptime`

Network Verification

`ip a ip route`

SSH Availability

`systemctl status ssh`

18.3 Wazuh Service Management

Service Status

`systemctl status wazuh-manager systemctl status wazuh-indexer systemctl status wazuh-dashboard`

Restart Services

`sudo systemctl restart wazuh-manager sudo systemctl restart wazuh-indexer sudo systemctl restart wazuh-dashboard`

18.4 Log & API Validation

Manager Logs

`journalctl -xeu wazuh-manager`

API Authentication Test

```
curl -k -u '<user>:<password>' \
https://127.0.0.1:55000/security/user/authenticate?raw=true
```

Indexer Health

`curl -k -u admin:<password> \ https://127.0.0.1:9200/_cat/indices/wazuh-alerts-4.x-*?v`

---

19. SOC-Style Verification Checklist

This checklist is used after:

- Reboots
- Configuration changes

- Service restarts
- Snapshot restores

Infrastructure

- Proxmox reachable
- VM running
- Snapshot integrity verified

OS

- Static IP preserved
- Gateway reachable
- SSH accessible

SIEM

- Wazuh Manager active
- Indexer health green
- Dashboard reachable
- Alerts index present

> **SOC Mindset:** If it's not verified, it's not trusted.

---

20. Security Rationale Behind Design Choices

Why Proxmox?

- Type-1 hypervisor
- Strong VM isolation
- Snapshot control
- Widely used in homelab and enterprise environments

Why Ubuntu Server?

- Stability and LTS support
- Extensive documentation
- Common in SOC and cloud deployments

Why Single-Node Wazuh?

- Simplified troubleshooting
- Lower resource requirements
- Ideal for learning pipelines before scaling

---

21. Documentation Philosophy Used

This runbook follows **professional security documentation principles**:

- **What** was done (action)
- **Why** it was done (reasoning)
- **How** it was validated (proof)
- **What broke** (failure cases)
- **How it was fixed** (resolution)
  This mirrors:
- SOC playbooks
- Incident response runbooks
- Platform engineering documentation

---

22. Interview-Ready Explanation (Use This Verbatim)

> "I built and documented a SOC-style cybersecurity homelab using Proxmox and Ubuntu Server. I deployed a Wazuh SIEM stack, configured stable networking, and validated log ingestion through the indexer API rather than relying solely on dashboards. I intentionally documented failures like authentication errors and role misconfigurations, then fixed them using systemd logs and API calls. I treat the lab like production infrastructure and maintain snapshots, verification checklists, and operational runbooks."

---

23. What I Would Improve Next (Growth Signal)

- Multi-node Wazuh deployment
- Agent rollout on additional hosts
- Custom alert rule creation
- Alert severity tuning
- Integration with ticketing or notification systems
- Network segmentation for SIEM traffic