Configuration Manual

# Introduction

Zero Trust Architecture (ZTA) is a cybersecurity paradigm that eliminates implicit trust within network boundaries and replaces it with a never-ending verification of all resource access, whether that is by users or by devices. According to NIST, Zero Trust can be defined as a direction in which there is no explicit trust directly placed on assets or user accounts based only their physical geographical location or network location. ZTA in practice focuses on strict authentications, excessive access right with least-privilege, and continuous observation of system behaviours. Ransomware defence is one such crucial application as ransomware usually leaves numerous copies of the file encrypted with the deletion of the original, a comprehensive file and process monitoring should be able to pick up on such behaviour pattern. File integrity monitoring in Wazuh, as an example, can highlight such an excessive creation/deletion of files that defines the characteristics of a ransomware attack. The configuration manual describes an experimental Zero Trust testbed based on VirtualBox, Ubuntu 22.04 LTS (victim, Linux distribution), Kali Linux (attacker, Linux distribution) and a set of security tools (Wazuh, auditd, UFW, OpenSSH, etc) to implement monitoring, tight access control, and ransomware containment. The aim is to bring out a step by step guidable that can be documented academically on such a set-up.

# Configurations

|  |  |  |
| --- | --- | --- |
| **Component** | **Purpose in Setup** | **Configuration Steps** |
| Oracle VirtualBox 7.1.12 | Hypervisor to generate and operate isolated virtual machines (VMs). | 1. Available at the official site of Oracle.  2. Installer Win - use a .exe Installer.  3. Add user to vboxusers group for USB/VM access. |
| Ubuntu 22.04 LTS (Victim VM) | Wazuh Manager, auditd and UFW is deployed in the following monitored system. | 1. Establish VM in VirtualBox (2–4 GB RAM, 2 CPUs, 20+ GB storage).  2. Install OS, by attaching Ubuntu ISO.  3. Update packages. |
| Kali Linux (Attacker VM) | Kali Linux allows creating an environment of an attacker to test ransomware. | 1. Establish VM in VirtualBox (2 Gb RAM 2 CPUs).  2. Install OS, by attaching Kali ISO.  3. Configure Host-only/Internal network to connect to Ubuntu VM. |
| Static IP Configuration | High risk of IP conflicting amongst VMs. Static IP Configuration Provides consistent network addresses between VMs. | Set each VM with fixed IP on network setting (Host-only/Internal network). |
| Wazuh Manager + Dashboard | Wazuh Manager + Dashboard SIEM/XDR platform to track audit logs and find anomalies. | 1. Add Wazuh repo & GPG key.  2. Install manager & dashboard.  3. Enable & start services. |
| auditd | The kernel-level logging of commands, accesses to files, system calls. | 1. Install auditd.  2. Install additional Audit Policies: propel Audit Policies: command exec, sudoers file, and sensitive dirs.  3. Load rules. |
| Wazuh–auditd Integration | Wazuh-auditd Integration Enables Wazuh to be able to parse audit logs in real time in order to provide alerts. | 1. Go to the editing /var/ossec/etc/ossec.conf with audit log.  2. Restart Wazuh agent/manager. |
| UFW (Uncomplicated Firewall) | UFW (Uncomplicated Firewall) Performs micro-segmentation and limits access to networks. | 1. Allow outgoing, set default deny of incoming.  2. Only Kali should be permitted to SSH.  3. Enable firewall. |
| OpenSSH Server | OpenSSH Server Allows remote secure management across VMs. | 1. Install the OpenSSH client and server.  2. Enabling SSH service. |
| Fake Ransomware (Python) | Fake Ransomware (Python) mimics the ransomware encryption processes used to test detection. | 1. Write the script to change/encrypt files.  2. Install in Kali to Ubuntu using SCP / SSH.  3. Perform alert operation and supervise. |
| htop | This monitors system performance as it is under attack. | Install and use htop when testing attacks. |

# Implementation

Implementation process is sequential and structured to establish, configure and test the Zero Trust Architecture (ZTA) laboratory environment. This gives a replicable, understandable, way to check every step with the visual and technical evidence. Image captures and screenshots are needed to record configuration state at every point and are useful in the later assessment.

**STEP 1: Set Up the Virtual Environment**

**1.1 Install VirtualBox (or VMware) [1]**

* Install **VirtualBox** here <https://www.virtualbox.org/>
* Download ISOs:
  + Ubuntu 22.04 LTS [2]
  + Kali Linux (latest) [3]

**1.2 Create 2 VMs**

* VM1: Ubuntu (Victim)
* VM2: Kali (Attacker)

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1. VirtualBox with both VM installed

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 2. VM settings for Ubuntu

A screenshot of a computer

AI-generated content may be incorrect.

Figure 3. VM settings for Kali

* Set **Internal Network** on both VMs so they are isolated but can talk to each other:
  + Go to Settings > Network > Adapter 1 > Attached to: Internal Network
  + Name: ZTA-Net

**1.3 Assign Static IPs**

**On VM1:**

sudo nano /etc/netplan/01-netcfg.yaml

network:

version: 2

ethernets:

enp0s3:

dhcp4: no

addresses: [192.168.56.101/24]

Apply:

sudo netplan apply

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 4. Static Ip assigned to Ubuntu

**Do the same on VM2, with IP 192.168.56.102**

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 5. Static Ip assigned to Kali

**STEP 2: Set Up Micro-Segmentation**

**2.1 On Ubuntu (VM1):**

Install firewall: [6]

sudo apt update

sudo apt install ufw

**2.2 Deny all incoming traffic:**

sudo ufw default deny incoming

sudo ufw allow from 192.168.56.102 to any port 22 (Allow only attacker access via SSH)

sudo ufw allow out

sudo ufw enable

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 6. ufw status verbose

**STEP 3: Set Up Strict Verification**

**3.1 Create New User with Limited Privileges:**

sudo adduser testuser

sudo usermod -L testuser # Lock the password

sudo usermod -s /usr/sbin/nologin testuser # Disable shell access

**3.2 Optional: Limit file access**

sudo chmod 700 /home/testuser

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

Figure 7. Testuser added

**STEP 4: Install and Configure Wazuh [4]**

**4.1 On VM1 (Ubuntu - Victim):**

Install Wazuh agent:

curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --dearmor -o /usr/share/keyrings/wazuh.gpg

echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee /etc/apt/sources.list.d/wazuh.list

sudo apt update

sudo apt install wazuh-manager -y

A screenshot of a computer

AI-generated content may be incorrect.

Figure 8. Wazuh status

**4.2 Start Wazuh:**

sudo systemctl enable wazuh-manager

sudo systemctl start wazuh-manager

A computer screen with text

AI-generated content may be incorrect.

Figure 9. Wazuh log

A screenshot of a computer

AI-generated content may be incorrect.

Figure 10. Wazuh login page

**STEP 5: Simulate Ransomware Behaviour**

**5.1 On Kali (VM2):**

Install OpenSSH if you want to connect: [7]

sudo apt install openssh-client

ssh [username@192.168.56.101](mailto:username@192.168.56.101)

A computer screen shot of a computer

AI-generated content may be incorrect.

Figure 11. SSH connection to victim machine

**5.2 Create Fake Ransomware Script on VM1:**

echo "Top secret" > secret1.txt

echo "Don't share this" > secret2.txt

A screenshot of a computer

AI-generated content may be incorrect.

Figure 12. Before creation of files (victim files)

nano fake\_ransom.py

Paste:

import os, glob

for file in glob.glob("\*.txt"):

os.rename(file, file + ".locked")

Run:

python3 fake\_ransom.py

A screenshot of a computer

AI-generated content may be incorrect.

Figure 13. Fake ransomware script added

A screenshot of a computer

AI-generated content may be incorrect.

Figure 14. Locked files due to ransomware script

**STEP 6: Enable Audit Monitoring [5]**

**6.1 On VM1:**

sudo apt install auditd

sudo auditctl -w /home/ -p war -k ransomware\_test

Run ransomware again, then check:

sudo ausearch -k ransomware\_test

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 15. ausearch logs after the fake ransomware python script was run

**STEP 7: Collect Evaluation Metrics**

A screenshot of a computer

AI-generated content may be incorrect.

Figure 16. Wazuh dashboard referring to threat severity

The **commands listed in below table** (also shown in the image) should be run inside the **Victim VM (Ubuntu)** — because that is where:

* The **ransomware simulation** happens
* The **Wazuh agent** or auditd is logging the events
* The **system metrics** like CPU/RAM usage and file encryption activity occur

| **Metric** | **Run on** | **Detailed Instruction** |
| --- | --- | --- |
| **Propagation Success** | **Victim VM (Ubuntu)** | After running the fake ransomware script:ls \*.locked | wc -l |
| **Detection Latency** | **Victim VM (Ubuntu)** | Run:sudo ausearch -k ransomware\_testNote timestamps of first file change |
| **Containment Time** | **Victim VM (Ubuntu)** | Manually note: 1. When alert is triggered2. When ufw rules are applied |
| **CPU/RAM Usage** | **Victim VM (Ubuntu)** | Install and run:sudo apt install htop -y && htop [8] |
| **False Positives** | **Victim VM (Ubuntu)** | Monitor alerts:sudo tail -f /var/ossec/logs/ossec.logLook for alerts unrelated to attack |

****

Figure 17. Number of locked files (Propagation success)

**Workflow:**

1. Log into **Victim VM**
2. Run fake ransomware:

python3 fake\_ransom.py

1. Check how many .locked files:

ls \*.locked | wc -l

1. View alert time:

ausearch -k ransomware\_test

1. Start htop to capture CPU usage:

htop

1. Monitor Wazuh logs:

tail -f /var/ossec/logs/ossec.log

**STEP 8: Document Your Findings**

* Architecture diagram
* Description of each step
* Screenshots
* Metric table

Ethical note: simulation only, no real malware.

**CLEAR PLAN TO RUN NEW EXPERIMENTS IN THE FUTURE:**

Use existing **Algorithm (Phases 1–6)** as the backbone, below I list each add-on with concrete steps so you can plug them into the same workflow.

**Exp A: Establish a clean baseline (No ZTA vs ZTA)**

**Goal:** Causal improvement deltas (Δ latency, Δ files encrypted, Δ CPU).

**Steps (one testbed):**

1. **Baseline (No-ZTA):** Disable auditd rules, stop Wazuh agent, set UFW to allow all. Snapshot.
2. Run the same Python simulator; record: detection latency (expected “none”), files encrypted, CPU/RAM.
3. **ZTA condition:** Restore snapshot, enable auditd + Wazuh + UFW (current config).
4. Repeat runs, collect metrics. Target **N≥30** per condition; report mean, SD, 95% CI, and Δ between conditions. (Attach one master table.)

**Exp B: Separate the Wazuh manager (resilience)**

**Goal:** Reflect realistic SOC placement.

**Steps (add one VM):**

1. Add a **Wazuh Manager VM**; leave **Agent on Ubuntu endpoint** only. Update agent to point to new manager IP.
2. Rerun Exp A’s ZTA condition, confirm alerts flow and measure same metrics. (Note in methods that only topology changed.)

**Exp C: Lateral movement containment**

**Goal:** Validate micro-segmentation beyond a single host.

**Steps (add one VM):**

1. Add a third Linux VM (“Neighbor”).
2. UFW on Endpoint: default-deny, allow only necessary management flows; **explicitly block SMB 445/139 and RDP 3389**, log deny.
3. From Attacker, attempt scans and connections (e.g., nmap, smbclient, ssh) to Neighbor, verify blocks in UFW logs and record any successful sessions.
4. Report success rate of blocks (%), with example log lines.

**Exp D: Identity flavored hardening (minimal)**

**Goal:** Keep it lightweight but closer to ZTA’s “verify explicitly.”

**Steps (same VMs):**

1. Keep the **non-interactive user** (/usr/sbin/nologin) and restrictive permissions, add simple **PAM** lockouts (e.g., faillock) and sudo policy to require TTY + reason codes.
2. Attempt credential abuse (password guess, su, sudo policy violations); log and report alert mapping.

**Exp E: Statistical hygiene**

**Goal:** Make results publishable.

**Steps:**

1. Pre-declare metrics: detection latency (s), files encrypted (%), CPU/RAM (%), alert FP/TP counts under a defined benign workload.
2. Use **N≥30** per condition; compute mean, SD, 95% CI; if distributions look normal, two-sample t-tests ZTA vs No-ZTA; otherwise non-parametric (Mann-Whitney).
3. Update Abstract and Table 2 from the master results table only.

**C) One-page “master table” you’ll produce after new runs**

(Template fill with numbers from Exp A/B/C/E)

| **Metric** | **No-ZTA (mean±SD, N)** | **ZTA-same-host (mean±SD, N)** | **ZTA-manager-off-host (mean±SD, N)** |
| --- | --- | --- | --- |
| Detection latency (s) | – | – | – |
| Files encrypted (%) | – | – | – |
| CPU during attack (%) | – | – | – |
| FP rate (benign) | – | – | – |
| Lateral-move blocks (%) | – | – | – |

# References

[1] Oracle (2023). *Oracle VM VirtualBox*. [online] VirtualBox. Available at: <https://www.virtualbox.org/>.

[2] Ubuntu 22.04.2 LTS *(Jammy Jellyfish)*. [online] Available at: <https://www.releases.ubuntu.com/22.04/>.

[3] Kali Linux. (n.d.). *Get Kali*. [online] Available at: <https://www.kali.org/get-kali/#kali-platforms>.

[4] Wazuh (2018). *Wazuh*. [online] Wazuh. Available at: <https://wazuh.com/>.

[5] GeeksforGeeks (2022). *Auditd Tool for Security Auditing on Linux Server*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/linux-unix/auditd-tool-for-security-auditing-on-linux-server/>.

[6] Ubuntu (2023). *UFW - Community Help Wiki*. [online] help.ubuntu.com. Available at: <https://help.ubuntu.com/community/UFW>.

[7] Terpollari, O. (n.d.). *How to Install and Configure OpenSSH Server In Linux*. [online] www.tecmint.com. Available at: <https://www.tecmint.com/install-openssh-server-in-linux/>.

[8] GeeksforGeeks (2019). *htop command in Linux with examples*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/linux-unix/htop-command-in-linux-with-examples/>.