

Mini SOC Lab Documentation

Project Title: Building a Mini SOC Lab using Wazuh SIEM

Platform: Oracle VirtualBox

Author: *Rishav Kumar Thapa*

Date: 6/11/2025

1. Project Overview

This project demonstrates the setup of a Mini Security Operations Center (SOC) within a controlled virtualized lab environment. The main objective is to collect, monitor, and analyze endpoint logs to detect malicious activity and better understand incident detection workflows.

Wazuh, an open-source Security Information and Event Management (SIEM) tool, is deployed on an Ubuntu virtual machine to act as the SOC server. A Windows 10 machine is configured as a monitored endpoint. This environment replicates a real-world SOC scenario at a foundational level, enabling hands-on experience with threat detection and response.

2.Lab Components

VM Name

- ubuntu-siem
- win10-client

Operating System

- Ubuntu Desktop 22.04
- Windows 10 Pro

Purpose

- Wazuh SIEM server (manager + dashboard)
- Endpoint with Wazuh agent and Sysmon installed

3. Key Features Implemented

- Installation and configuration of Wazuh Manager on Ubuntu
- Deployment of Wazuh Dashboard for centralized monitoring
- Installation of Wazuh Agent on Windows 10
- Deployment of Sysmon on Windows with a custom ruleset
- Secure agent-server integration
- Real-time event monitoring and alert visibility
- Display of logs related to system, security, and application events

4. Setup Summary

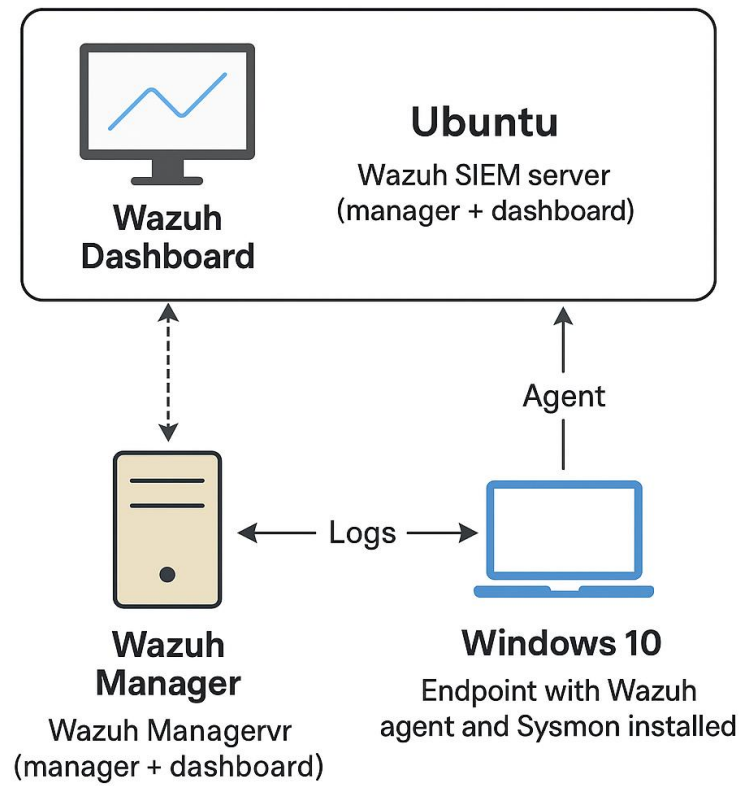
Ubuntu Wazuh Server:

- Installed Wazuh Manager using official installation script
- Deployed Wazuh Dashboard (web interface)
- Configured Wazuh to accept agents

Windows 10 Client:

- Installed Wazuh Agent
- Connected agent to server (verified via dashboard)
- Installed Sysmon and applied a public ruleset (e.g., SwiftOnSecurity)
- Confirmed Sysmon logs were forwarded to the server

5. Architecture Diagram

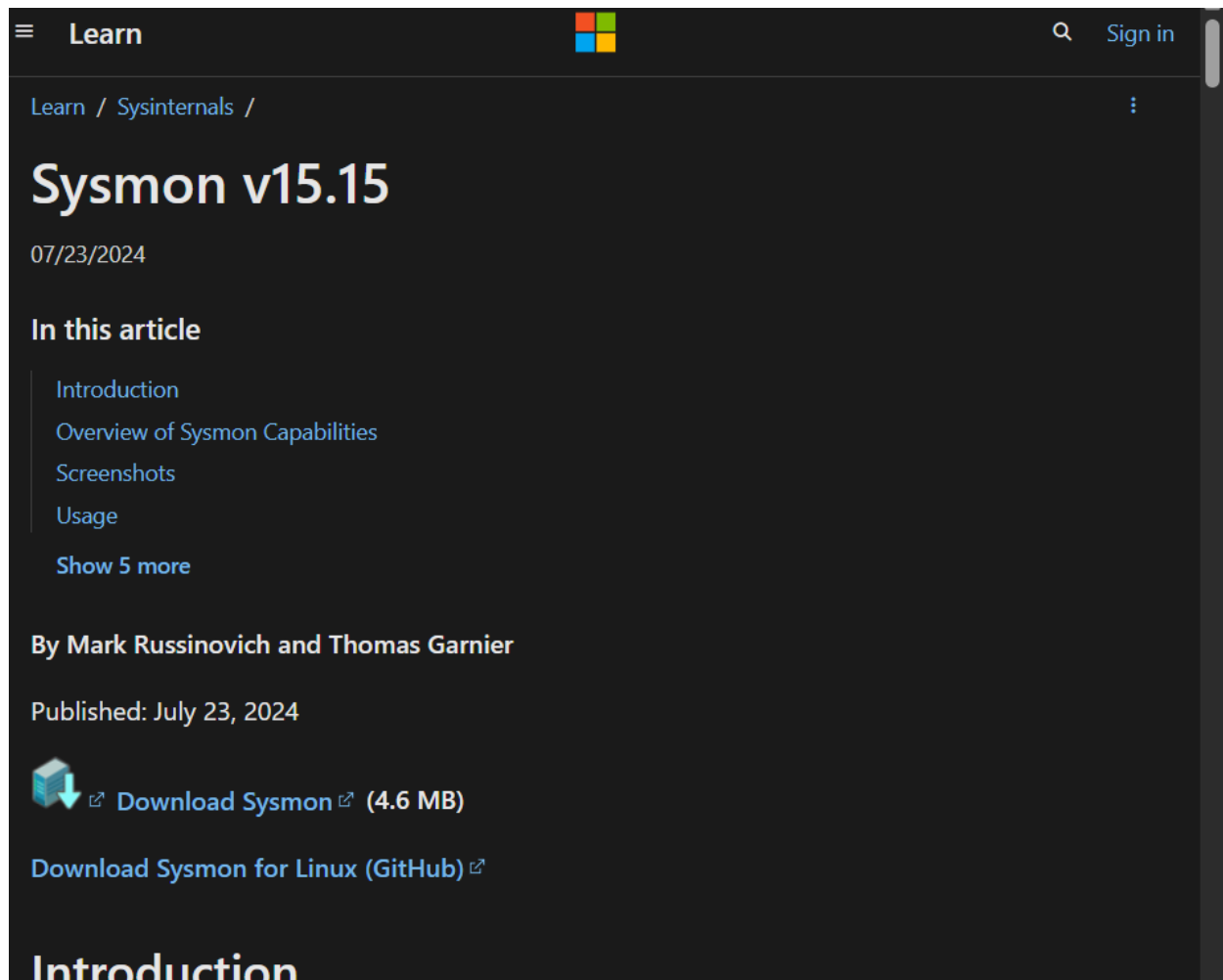


6. Setup Implementation Overview

This section outlines the practical steps taken to successfully install and configure the Mini SOC Lab environment using Wazuh and Sysmon. Although active detection and response scenarios were not simulated, the full system setup was completed, allowing for future testing and monitoring.

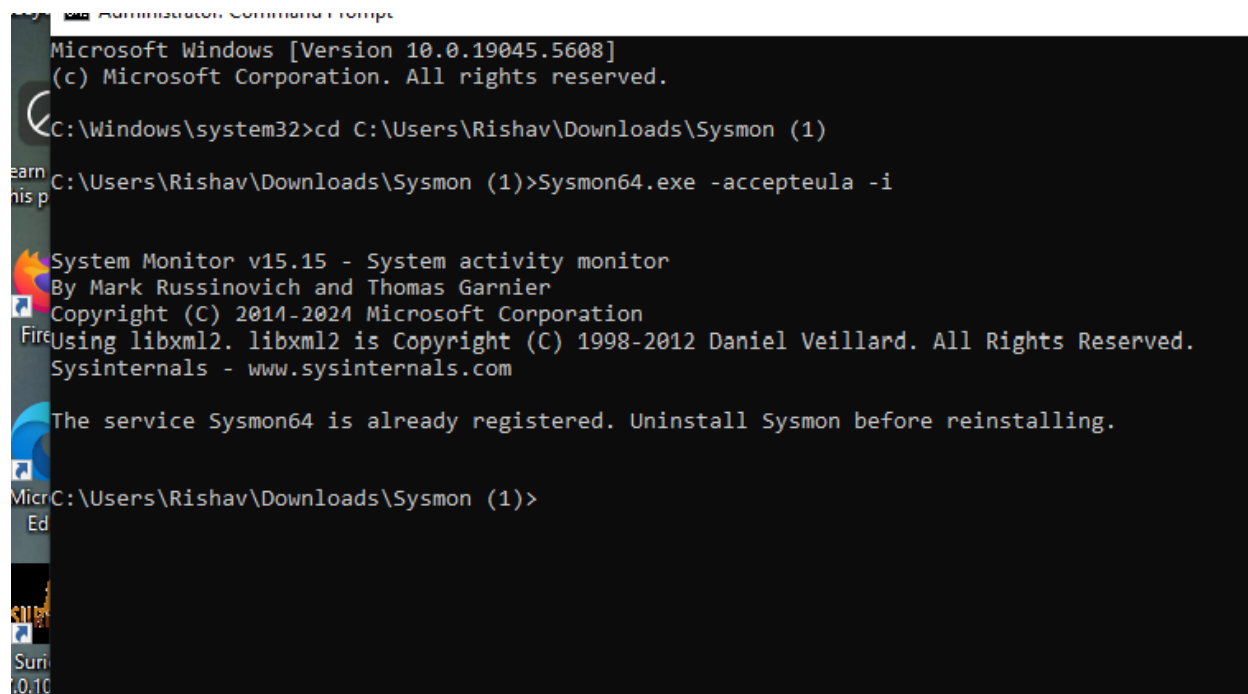
Step 1: Installing Sysmon on Windows 10

To begin the demonstration, I am downloading Sysmon, a system monitoring tool from Microsoft's Sysinternals suite. Sysmon provides detailed logging of system activity, which is crucial for detecting and investigating security events. I'm obtaining the latest version directly from the official Microsoft documentation page to ensure the tool is authentic and up to date.



Step 2: Installing and Registering Sysmon Service

Next, I run the Sysmon executable with administrative privileges to install and register the Sysmon service on the Windows 10 system. The command includes the '-accepteula' flag to automatically accept the license agreement, and '-i' to install the service. The system responds that Sysmon is already registered, indicating the service is currently active. If needed, Sysmon would have to be uninstalled before reinstalling to avoid conflicts.



```
Microsoft Windows [Version 10.0.19045.5608]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\Users\Rishav\Downloads\Sysmon (1)

C:\Users\Rishav\Downloads\Sysmon (1)>Sysmon64.exe -accepteula -i

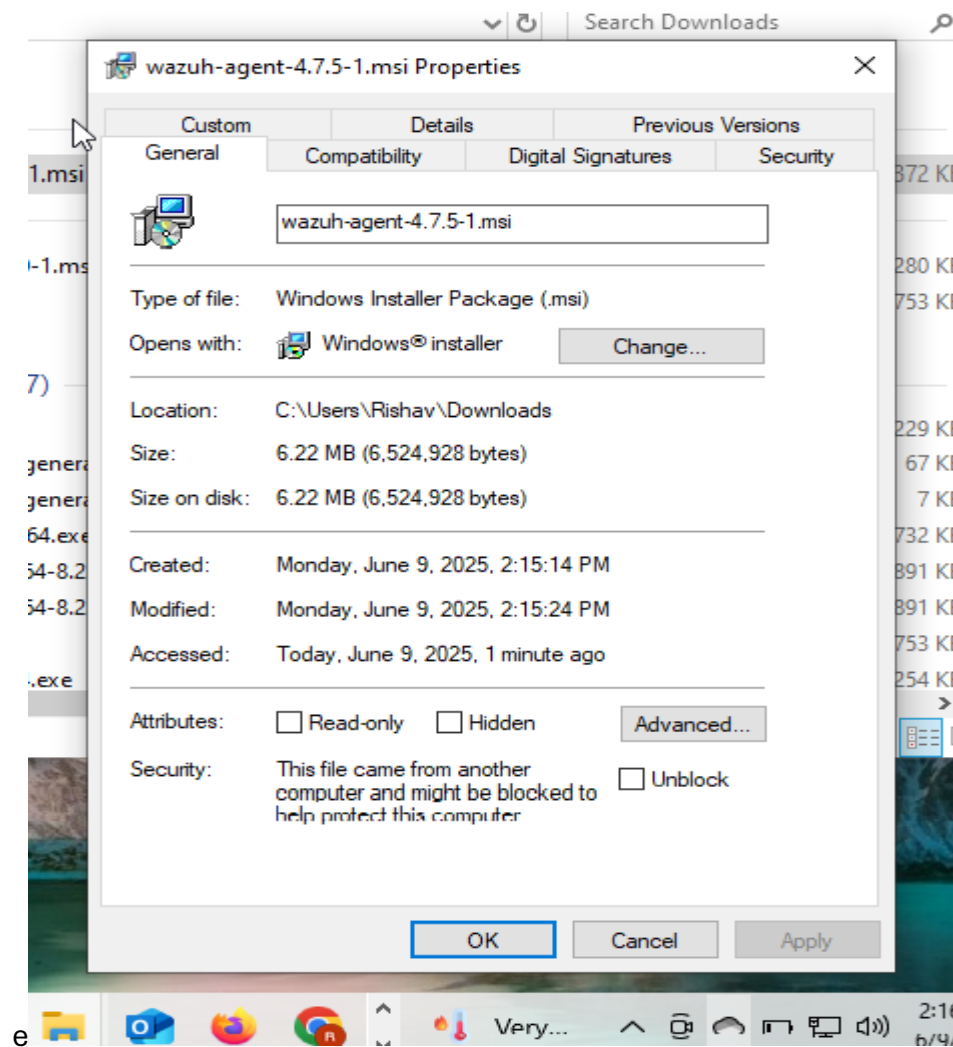
System Monitor v15.15 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2024 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights Reserved.
Sysinternals - www.sysinternals.com

The service Sysmon64 is already registered. Uninstall Sysmon before reinstalling.

C:\Users\Rishav\Downloads\Sysmon (1)>
```

Step 3: Downloading and Installing Wazuh Agent on Windows 10

In this step, I am downloading the Wazuh agent version 4.7.5 on the Windows 10 machine. The Wazuh agent is essential for collecting and forwarding security event data from the endpoint to the centralized Wazuh server running on Ubuntu. Ensuring the agent version matches the server version helps maintain compatibility and reliable communication between the client and server.

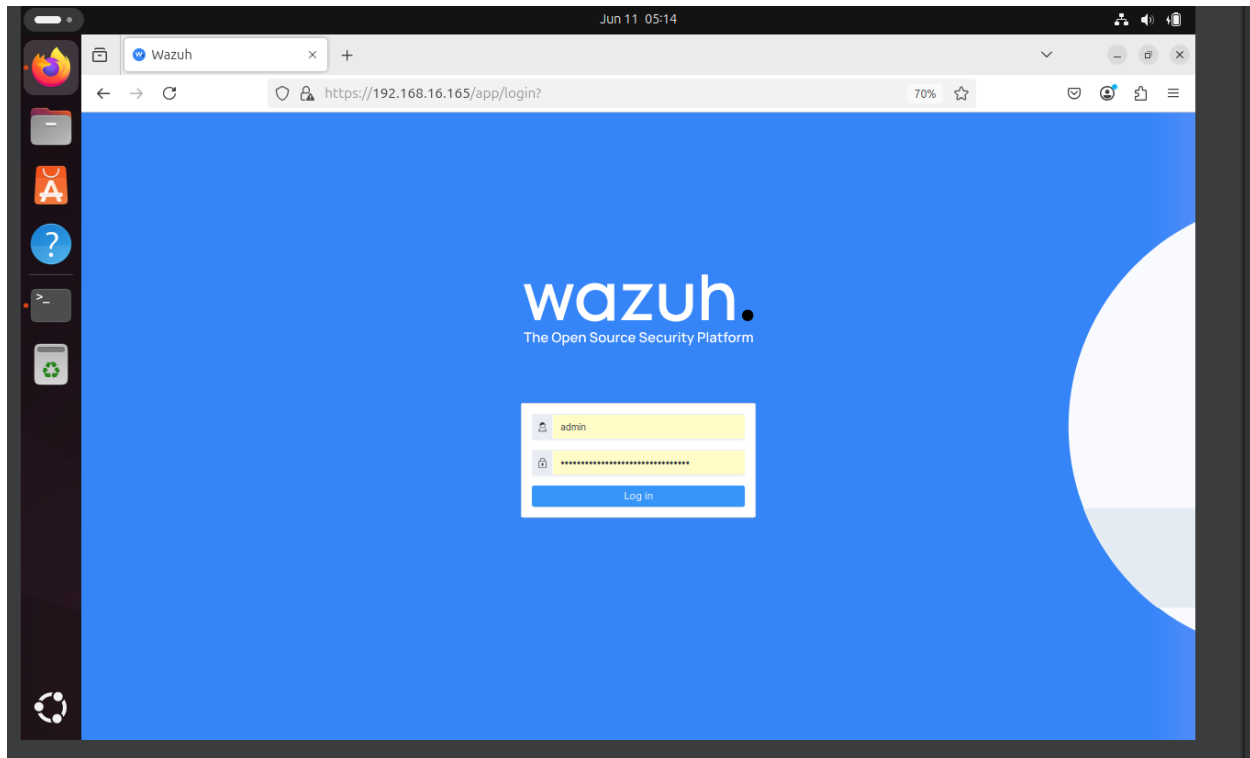


Step 4: Installing Wazuh Server on Ubuntu

On the Ubuntu server, I download the Wazuh installation script using curl to fetch it directly from the official Wazuh repository. I then execute the script with elevated privileges (sudo) to install the Wazuh server components. During installation, I set the administrative password (wazuhadmin) to secure access to the Wazuh web interface. After the installation completes, I receive a unique authentication key for enrolling agents. Finally, I verify the Wazuh server is running and accessible by logging into the web dashboard using the username admin and the password configured during installation, via the server's IP address in a web browser.

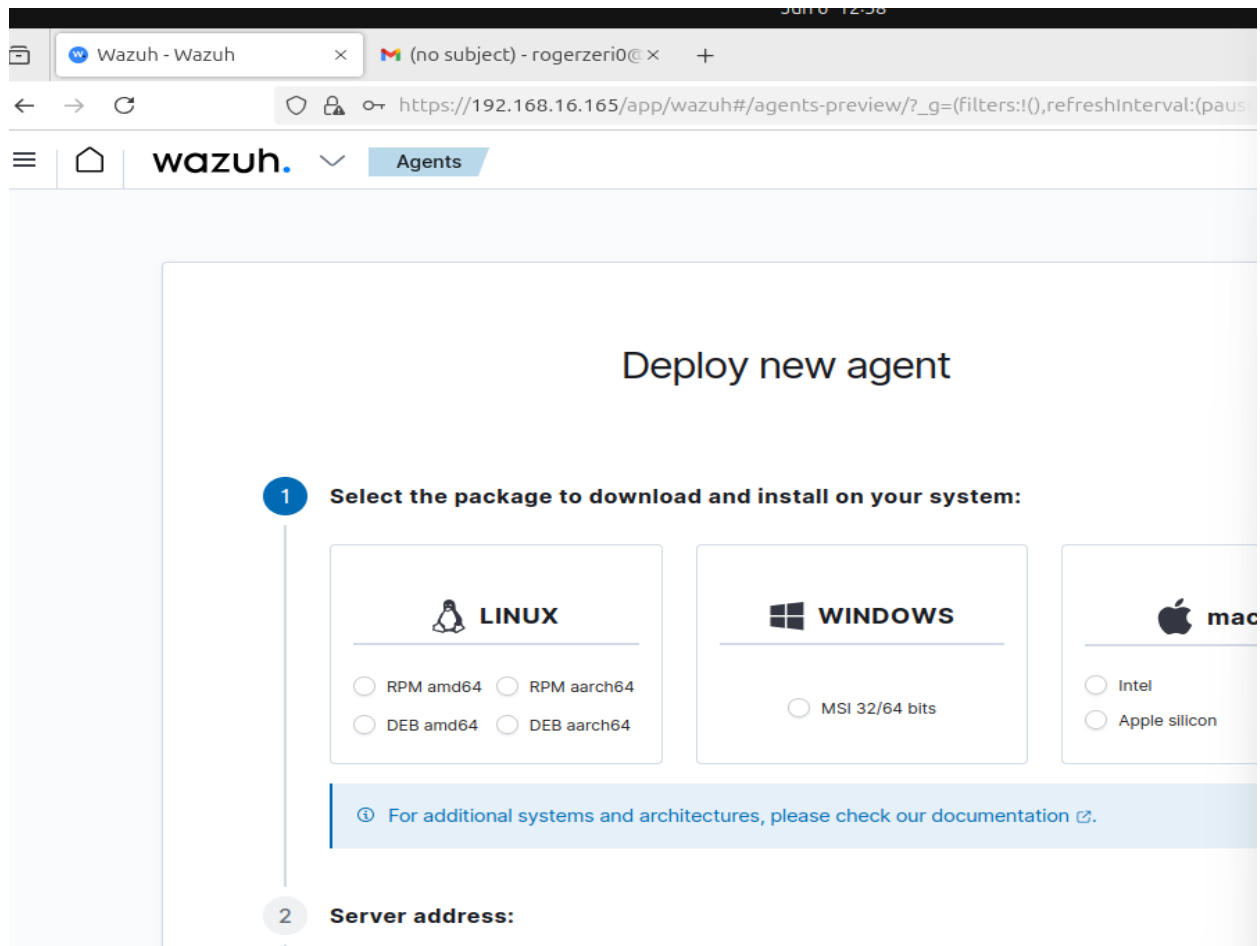
Cmd

- `curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh`
- `sudo bash wazuh-install.sh --wazuh-password wazuhadmin`



Step 5: Deploying New Agents via Wazuh Dashboard or Command Line

“Once the Wazuh server is up and running, new agents can be deployed and managed efficiently through two main methods. The first is via the Wazuh dashboard’s ‘Agents’ section, which provides a user-friendly interface to register and monitor agents. Alternatively, deployment and management can be performed using command-line tools on the Ubuntu server, allowing for scripting and automation of agent registration and configuration, which is especially useful in larger or dynamic environments.”



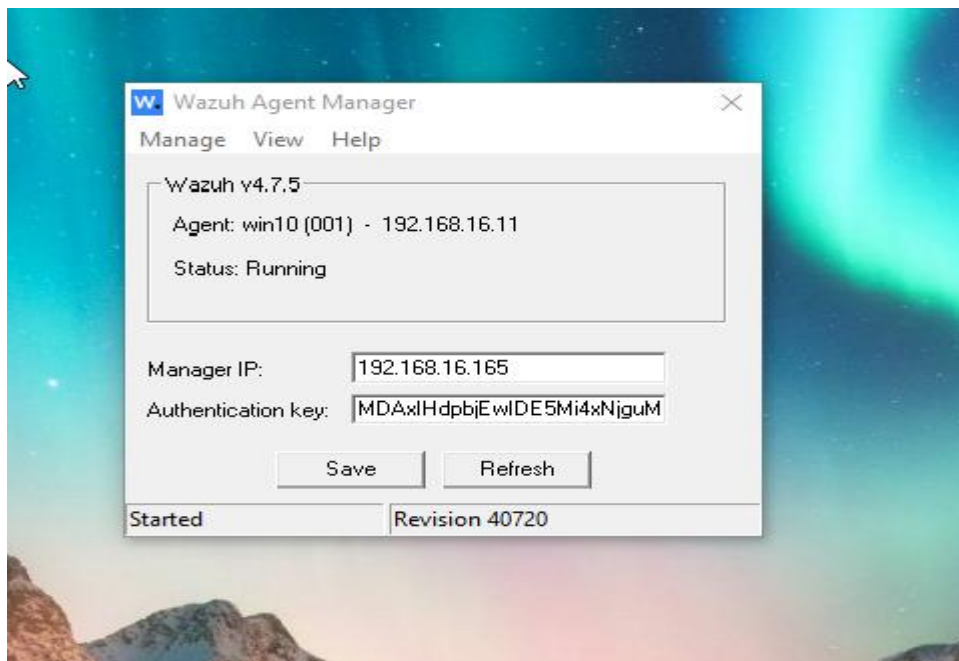
On Ubuntu server, I use the `manage_agents` utility to handle Wazuh agent operations from the command line. This tool provides options to add new agents, extract their authentication keys, list existing agents, or remove them as needed. Using this interactive menu, administrators can efficiently control agent enrollment and maintenance without relying solely on the web dashboard, which is beneficial for automation or remote management.

```
vboxuser@UBUNTU:~$ sudo /var/ossec/bin/manage_agents
[sudo] password for vboxuser:

*****
* Wazuh v4.7.5 Agent manager.          *
* The following options are available: *
*****
  (A)dd an agent (A).
  (E)xtract key for an agent (E).
  (L)ist already added agents (L).
  (R)emove an agent (R).
  (Q)uit.
Choose your action: A,E,L,R or Q:
```

Step 6: Verifying Wazuh Agent Status on Windows 10

On the Windows 10 client, I open the Wazuh Agent Manager application to verify the agent's connection status with the Wazuh server. The interface displays the agent's name, unique ID, IP address, and confirms that the agent service is actively running. This verification step ensures that the agent is properly communicating with the server and ready to send security event data for monitoring and analysis.



Step 7: Listing Active Wazuh Agents on the Server

“Using the `agent_control` command on the Ubuntu server, I retrieve a list of all registered Wazuh agents. This command shows each agent’s ID, name, IP address, and status. In this case, both the local Ubuntu server and the Windows 10 agent are listed as active, confirming successful communication and registration within the Wazuh environment.”

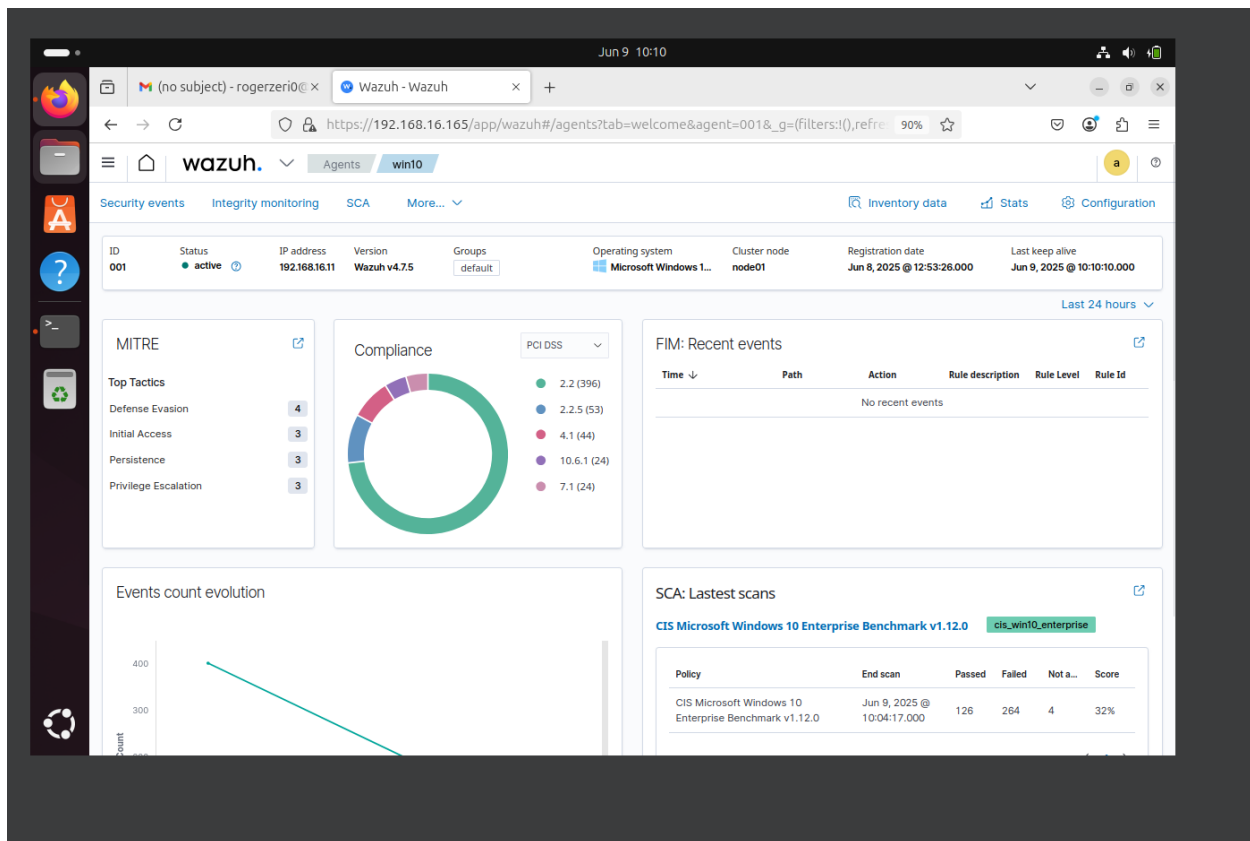
```
vboxuser@UBUNTU: ~  
ddr      valid_lft 258982sec preferred_lft 172582sec  
         inet6 2407:1400:aa71:48e0:6d09:cfe7:3c75:af8c/64 scope global temporary dyna  
mic      valid_lft 258789sec preferred_lft 84423sec  
         inet6 2407:1400:aa71:48e0:a00:27ff:fe36:77cb/64 scope global dynamic mngtmpa  
ddr      valid_lft 258789sec preferred_lft 172389sec  
         inet6 fdef:2024:ef2f:0:22c:11fe:472d:1c1c/64 scope global temporary dynamic  
         valid_lft 603124sec preferred_lft 84423sec  
         inet6 fdef:2024:ef2f:0:a00:27ff:fe36:77cb/64 scope global dynamic mngtmpaddr  
  
         valid_lft forever preferred_lft forever  
         inet6 fe80::a00:27ff:fe36:77cb/64 scope link  
         valid_lft forever preferred_lft forever  
vboxuser@UBUNTU:~$ sudo /var/ossec/bin/agent_control -l  
  
Wazuh agent_control. List of available agents:  
  ID: 000, Name: UBUNTU (server), IP: 127.0.0.1, Active/Local  
  ID: 001, Name: win10, IP: 192.168.16.11, Active  
  
List of agentless devices:  
  
vboxuser@UBUNTU:~$
```

Step 8: Monitoring Security Events via the Wazuh Dashboard

The Wazuh Dashboard serves as the central interface for security monitoring. Key features include:

- **Security Events:** Live feed of system and application alerts.
- **Security Configuration Assessment (SCA):** Policy compliance checks.
- **Inventory Management:** System and software inventory.
- **MITRE & CIS Integration:** Helps identify threats using known security frameworks.

The dashboard provides actionable insights into the endpoint's security posture, supporting continuous monitoring and threat detection.



7. System Specifications

Ubuntu Desktop (ubuntu-siem)

RAM: 4–6 GB

Storage: 40 GB

Username: admin

Password: system provide

To update system:

- `sudo apt update && sudo apt upgrade -y`

Install Wazuh All-In-One

- `curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh`
- `sudo bash wazuh-install.sh --wazuh-password wazuhadmin`

8. References

- [Microsoft Sysmon](#)
- [SwiftOnSecurity Sysmon Config](#)

9. Challenges and Issues Faced

- **Ubuntu Crashes or Freezes:** During prolonged operation, the Ubuntu VM occasionally experienced performance degradation or system freezes. This was mitigated by increasing allocated memory and ensuring all updates were applied.
- **Agent Not Connecting:** At times, the Wazuh Agent on Windows did not reflect on the server dashboard. This was resolved by verifying IP addresses, ensuring the firewall was configured correctly, and rechecking agent authentication keys.
- **Sysmon Rules Not Triggering Alerts:** Initial ruleset misconfiguration led to missing alerts. Replacing the config with the SwiftOnSecurity Sysmon rules resolved this.

10. Future Enhancements

- Integration with ELK Stack for extended data visualization
- Automated alerting via email or messaging platforms (e.g., Slack)
- Use of additional endpoints (Linux/macOS) for cross-platform monitoring
- Integration with threat intelligence feeds for advanced detection
- Regular rule updates and tuning for improved detection accuracy
- Define custom rules for network scanning tools such as Nmap; e.g., detect scan intensity levels like Level 10 or 12, and generate alerts when thresholds are breached
- Schedule simulated attack scenarios using Metasploit for response training
- Develop automatic remediation scripts triggered by specific high-severity alerts

11. Conclusion

This Mini SOC Lab demonstrates the effective deployment of a security monitoring solution using Wazuh. By integrating Sysmon with Wazuh Agent and server infrastructure, it provides visibility into endpoint behavior, making it possible to detect and investigate suspicious events in real time.

Additionally, the documentation reflects both the technical process and the human experience acknowledging system failures and troubleshooting steps. With future enhancements, this lab can evolve into a highly functional and scalable security monitoring platform.

12. Future Enhancements

- Integration with ELK Stack for extended data visualization
- Automated alerting via email or messaging platforms (e.g., Slack)
- Use of additional endpoints (Linux/macOS) for cross-platform monitoring
- Integration with threat intelligence feeds for advanced detection
- Regular rule updates and tuning for improved detection accuracy

13. Appendix

Custom Wazuh Detection Rules

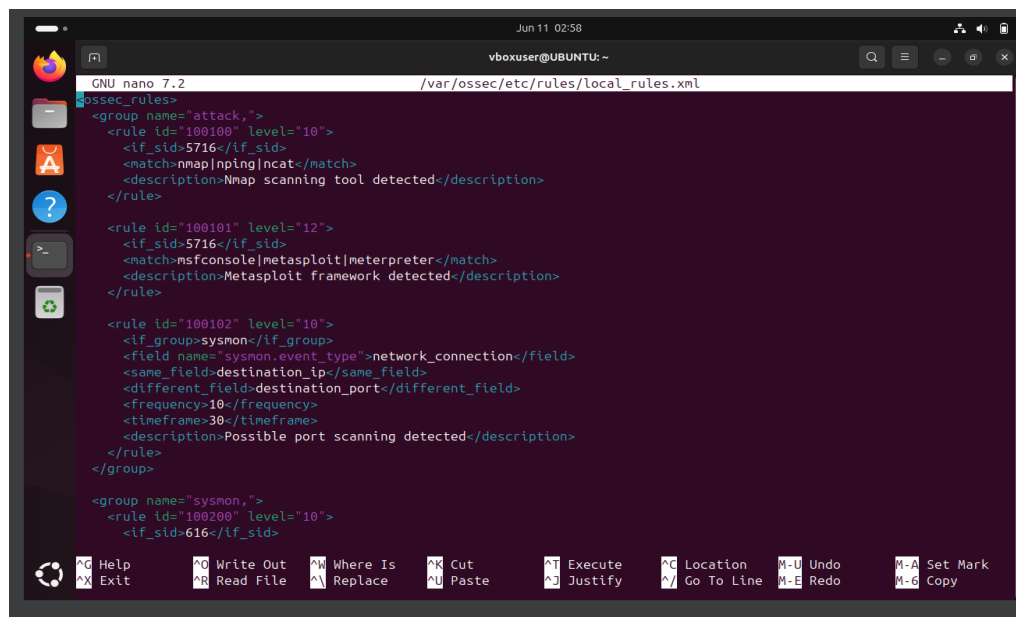
File Path: `sudo nano /var/ossec/etc/rules/local_rules.xml`

This screenshot displays the manually created custom detection rules added to Wazuh. These rules help enhance the default detection capabilities by identifying specific tools or behaviors that are commonly used during attack simulations.

- **Rule ID 100100** – Detects Nmap, Ncat, or Nping usage.
- **Rule ID 100101** – Detects use of Metasploit-related tools (e.g., msfconsole, meterpreter).
- **Rule ID 100102** – Detects rapid port scanning behavior by analyzing multiple connections to the same destination IP but different ports in a short time window, using Sysmon data.

These rules help simulate and detect:

- Reconnaissance activities (e.g., scanning with Nmap)
- Exploitation phase using Metasploit
- Suspicious behavior patterns flagged by Sysmon



```
GNU nano 7.2 /var/ossec/etc/rules/local_rules.xml
<ossec_rules>
  <group name="attack,">
    <rule id="100100" level="10">
      <if_sid>5716</if_sid>
      <match>nmap|nping|ncat</match>
      <description>Nmap scanning tool detected</description>
    </rule>

    <rule id="100101" level="12">
      <if_sid>5716</if_sid>
      <match>msfconsole|metasploit|meterpreter</match>
      <description>Metasploit framework detected</description>
    </rule>

    <rule id="100102" level="10">
      <if_group>sysmon</if_group>
      <field name="sysmon.event_type">network_connection</field>
      <same_field>destination_ip</same_field>
      <different_field>destination_port</different_field>
      <frequency>10</frequency>
      <timeframe>30</timeframe>
      <description>Possible port scanning detected</description>
    </rule>
  </group>

  <group name="sysmon,">
    <rule id="100200" level="10">
      <if_sid>616</if_sid>
```

14. Planned Attack Simulation Scenarios

To evaluate the effectiveness of the Mini SOC Lab, a series of simulated cyberattacks are planned for future testing. These simulations will help validate the detection capabilities of the Wazuh-based monitoring system, including the performance of custom rules and Sysmon configurations. Although these tests have not yet been executed, the lab is fully prepared to support and analyze them in upcoming validation phases.

Attack Simulation	Tool/Method	Expected Detection	Wazuh Rule ID
Port Scanning	nmap -sS, nmap -A	Detection of multiple port probes and scanning patterns	100100 / 100102
Exploitation Attempts	msfconsole, Metasploit	Alert on shell access, meterpreter activity, exploit use	100101
Brute-Force Login	Manual or scripted login	Multiple failed login attempts triggering security logs	Custom + Sysmon SIDs
File Tampering	Dropped or modified .exe	Unexpected file creation in sensitive paths	Sysmon Logs
Command Line Abuse	Encoded PowerShell, wmi	Suspicious command-line usage, process tree anomalies	Sysmon + MITRE Maps

Objective

These simulations are designed to:

- Validate the proper functioning of Wazuh detection mechanisms.
- Test the visibility of malicious behaviors via Sysmon.
- Ensure that custom rules (e.g., Rule ID 100100, 100101, 100102 — see Appendix) are correctly identifying common attack techniques.
- Demonstrate the lab's readiness for use as a cybersecurity research and training platform.