TUNISIAN REPUBLIC
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY OF CARTHAGE
HIGHER SCHOOL OF COMMUNICATIONS OF TUNIS (SUP'COM)

Engineering Internship Report

Prepared by
Habib Chelbi

Supervised by
Haythem Benna

Internship Period
23/06/2025 – 30/08/2025

Host Company
DOT'COM

Academic Year : 2024 – 2025

Acknowledgements

I would like to express my sincere gratitude to DOTCOM for giving me the opportunity to carry out my internship within its cybersecurity team. My deepest appreciation goes to my supervisor, Mr. Haythem Benna, for his valuable guidance, constructive feedback, and constant support throughout the course of this internship. I am also thankful to all the team members for their collaboration, encouragement, and for creating a professional and welcoming environment that enriched my learning experience.

General Introduction

In today's digital age, the security of information systems has become a central concern for organizations of all sizes. Protecting infrastructures against vulnerabilities, ensuring service availability, and safeguarding sensitive data are now essential objectives in an increasingly connected world.

My internship at DOT'COM offered me the opportunity to engage with these challenges by contributing to the reinforcement of the company's secure infrastructure. This experience allowed me to combine theoretical knowledge with practical applications in the field of cybersecurity, while gaining valuable insight into real-world organizational needs and constraints.

The purpose of this report is to present the work accomplished during the internship. It begins with an overview of DOT'COM and the context of the project, followed by a description of the theoretical foundations and technical environment that guided the work. The final part details the tasks completed, the methodologies applied, and the results obtained, concluding with reflections on the lessons learned and perspectives for improvement.

TABLE OF CONTENT

LIST OF FIGURES

List of Acronyms

- **HIDS** : Host-based Intrusion Detection System – Monitors activities on individual hosts for security threats.
- **SIEM** : Security Information and Event Management – Centralized system for collecting, analyzing, and alerting on security events.
- **SSH** : Secure Shell – Protocol for secure remote login and command execution.
- **DoS** : Denial of Service – Attack that aims to make a system or service unavailable.
- **DDoS** : Distributed Denial of Service – DoS attack using multiple sources to overwhelm a system.
- **API** : Application Programming Interface – Interface for software components to communicate ; used with VirusTotal API.
- **VM** : Virtual Machine – Virtualized environment simulating a physical computer.
- **IDS** : Intrusion Detection System – Monitors network or systems for malicious activity.
- **HTTP** : Hypertext Transfer Protocol – Protocol for web communication.
- **TCP** : Transmission Control Protocol – Reliable transport protocol used in networking.
- **UDP** : User Datagram Protocol – Connectionless transport protocol used in networking.
- **CVE** : Common Vulnerabilities and Exposures – Publicly disclosed security vulnerabilities (used in vulnerability scanning).
- **MD5** : Message Digest Algorithm 5 – Cryptographic hash function (may appear in malware analysis).

Chapter

1 Presentation

Introduction

This chapter provides an overview of the host organization, DOT'COM, and the project carried out during the internship. The aim is to present the professional environment in which the internship took place and to outline the objectives and scope of the assigned project.

1.1 Host Organism Presentation

DOTCOM, is a digital agency specialising in IT services and solutions. They help businesses improve their online presence through web development, server security and infrastructure setup. With a focus on digital transformation, DOTCOM uses its expertise to provide cutting-edge services and support growth by increasing visibility and sales online.

FIGURE 1.1 – logo of DOTCOM

1.2 Company's structure

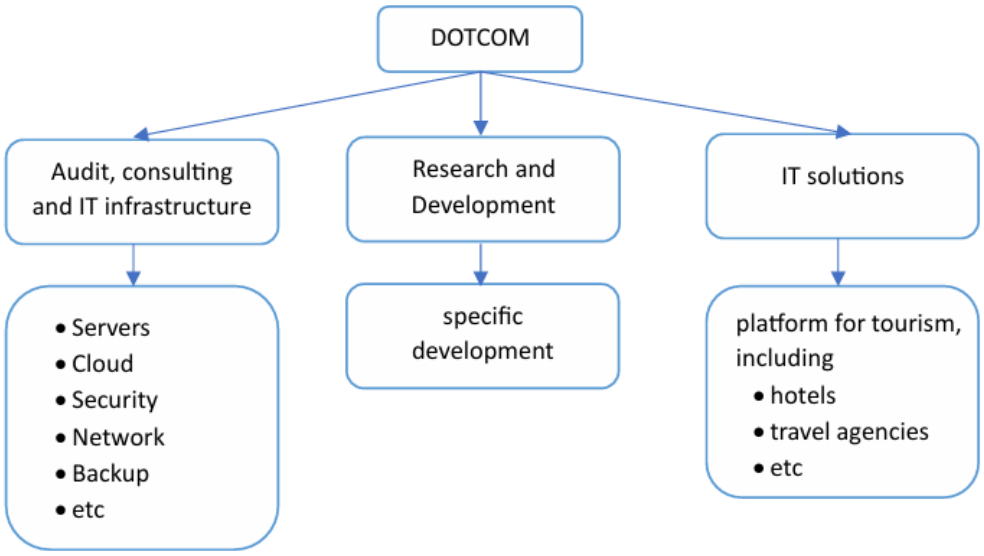


FIGURE 1.2 – DOTCOM Structure

My internship was in the Audit, Consulting and IT Infrastructure department.

1.3 Internship Project

The assigned project, "Hardening Infrastructure Security through Monitoring and Incident Detection", aimed to strengthen the resilience of IT systems and improve visibility over operations.

Objectives

— Deploy monitoring tools for continuous system and service supervision.

— Implement security solutions to detect vulnerabilities and intrusions.

— Provide actionable insights through dashboards and alerts.

Scope of Work

— Deployment and configuration of Zabbix for infrastructure monitoring.

— Integration of Wazuh as a SIEM solution for security event management.

— Automation of Wazuh agent installation across multiple hosts.

— Visualization of system and security metrics through integrated dashboards.

Conclusion

This chapter has provided an overview of DOT'COM, emphasizing its role as a digital agency and detailing the department in which the internship took place. By presenting the internship project, its objectives, and its scope, we have established the context for the practical work conducted. The following chapter, *Theoretical and Technical Foundations*, will delve into the key concepts, tools, and technologies that underpin the project, providing the necessary background to understand the implementation and results.

Chapter

2 Theoretical and Technical Foundations

Introduction

This chapter presents the theoretical and technical foundations necessary to understand the internship project. It covers key cybersecurity principles, monitoring concepts, and Security Information and Event Management (SIEM), as well as the main tools used, including Zabbix and Wazuh, which supported the implementation of infrastructure monitoring and incident detection.

2.1 Theoretical Foundations

Understanding the cybersecurity concepts underlying my internship project requires knowledge of several fundamental areas. These foundations provide the necessary context to comprehend attacks, monitoring, and defense mechanisms implemented during the project.

2.1.1 Cybersecurity Fundamentals

Cybersecurity aims to protect systems, networks, and data from unauthorized access or damage, focusing on the core principles of confidentiality, integrity, and availability (CIA triad). Threats can originate internally or externally, and may be targeted or opportunistic. Understanding risk management concepts, including vulnerabilities, threats, and potential impact, is essential for identifying and mitigating risks effectively.

2.1.2 Network Security Basics

Network security ensures the safe transmission of data across networks by protecting against unauthorized access and attacks. It is essential to understand the OSI model, as attacks can target different network layers. Familiarity with common network protocols such as TCP, UDP, HTTP, and SSH, as well as their potential vulnerabilities, forms the basis for network protection. Firewalls, intrusion detection/prevention systems (IDS/IPS), and network segmentation are key components of a robust network security strategy.

2.1.3 Host-Based Security and Monitoring

Host-based security involves protecting individual endpoints and monitoring their activities. Log collection and analysis allow detection of unusual behavior or security breaches. Tools such as Wazuh provide host-based intrusion detection (HIDS), enabling administrators to monitor system events, analyze threats, and respond to incidents effectively.

2.1.4 Authentication and Access Control

Authentication and access control are critical for ensuring that only authorized users can access system resources. Common authentication methods include passwords, SSH keys, and multi-factor authentication. Understanding attacks such as brute-force, dictionary attacks, and credential stuffing helps in implementing effective defenses. The principles of least privilege and proper account management further enhance system security.

2.1.5 Malware and Threat Detection

Malware refers to malicious software designed to disrupt, damage, or gain unauthorized access to systems. Common types include viruses, trojans, worms, and ransomware. Knowledge of how malware spreads and affects systems is essential for detection and prevention. Threat

intelligence platforms, such as VirusTotal, provide valuable insights by analyzing files and URLs to identify potential malware.

2.1.6 Vulnerability Assessment

Vulnerability assessment involves identifying weaknesses in systems that could be exploited by attackers. Tools and techniques for vulnerability scanning help uncover these weaknesses. Regular patch management and proactive remediation are essential to reduce risk and maintain system security.

2.1.7 Security Incident Detection and Response

Security incident detection and response is the process of identifying, analyzing, and mitigating security breaches. Real-time alerts and dashboards facilitate rapid response to incidents. Effective incident response follows a structured approach : detection, analysis, containment, eradication, and recovery, ensuring minimal impact on systems and data.

2.1.8 Security Logging and Event Correlation

Logging provides a record of system, application, and network events. Correlating these events helps detect anomalies that may indicate security incidents. Security Information and Event Management (SIEM) solutions centralize log collection, analyze data, and generate alerts to enhance overall situational awareness and response capabilities.

2.1.9 Types of Cyberattacks Relevant to the Internship

During my internship, several types of cyberattacks were studied and monitored using Wazuh, vulnerability scanning, and malware detection. Understanding these attacks is essential to comprehend the alerts and defenses implemented in the project.

2.1.9.1 Brute-Force Attacks

Brute-force attacks involve systematically trying all possible password combinations to gain unauthorized access to a system. In this project, controlled SSH brute-force attacks were performed on monitored hosts, and Wazuh successfully detected and alerted on these attempts, highlighting the importance of monitoring login activities.

2.1.9.2 Credential Stuffing and Password Attacks

Credential stuffing uses leaked or stolen credentials to access user accounts. Monitoring repeated failed login attempts and authentication anomalies allows early detection of such attacks, a functionality effectively demonstrated through the Wazuh alerts during the internship.

2.1.9.3 Malware Attacks

Malware includes viruses, trojans, worms, ransomware, and spyware designed to disrupt or gain unauthorized access to systems. Integration with VirusTotal enabled automated scanning of files to detect malicious content, reinforcing proactive threat detection.

2.1.9.4 Exploitation of Vulnerabilities

Attackers often exploit software or configuration weaknesses to gain unauthorized access or escalate privileges. Vulnerability scanning performed in the internship identified potential weaknesses in monitored systems, emphasizing the need for continuous assessment and remediation.

2.1.9.5 Denial of Service (DoS) and Distributed Denial of Service (DDoS)

DoS and DDoS attacks aim to overwhelm system resources, causing service disruption. Monitoring system and network activity with Wazuh allows detection of unusual patterns that may indicate such attacks.

2.1.9.6 Unauthorized Access and Privilege Escalation

Unauthorized access occurs when attackers gain access beyond their allowed permissions, often escalating privileges to compromise critical systems. Authentication anomalies and detailed log monitoring in Wazuh help detect such malicious activity.

2.1.9.7 Log Tampering and Data Manipulation

Attackers may attempt to hide traces of their activity or alter system logs to cover malicious actions. Wazuh continuously monitors system and application logs, ensuring integrity and detecting suspicious modifications.

2.1.9.8 Reconnaissance and Information Gathering

Reconnaissance involves scanning systems or networks to identify open ports, services, and vulnerabilities. Vulnerability scanning in the project simulates this activity, helping identify security gaps before attackers can exploit them.

Understanding these threats is essential for designing effective security measures, including monitoring, hardening, and incident response strategies.

2.2 Monitoring and Event Management

Monitoring and event management are essential components of cybersecurity, ensuring that systems remain available, performant, and secure.

2.2.1 Infrastructure Monitoring

Infrastructure monitoring involves continuously observing system resources and services to detect performance issues or failures. Key metrics include CPU, memory, disk usage, network traffic, and logs. Beyond traditional monitoring for uptime and performance, security monitoring also identifies anomalies and potential threats in real time.

2.2.2 Security Information and Event Management (SIEM)

Security Information and Event Management (SIEM) centralizes the collection, normalization, and analysis of logs from multiple sources to detect and respond to security incidents. SIEM improves visibility across the infrastructure, enables correlation of events, and supports timely incident response. Challenges include handling false positives and maintaining scalability in large environments.

2.3 Theoretical Background of Tools Used

2.3.1 Zabbix (Monitoring System)

Zabbix is an open-source monitoring solution that collects and analyzes data from IT infrastructure to ensure system health and performance. Its architecture includes :

- **Zabbix Server :** The central component that receives data from monitored hosts, stores it in a database, and evaluates triggers to generate alerts.
- **Zabbix Agents :** Installed on monitored hosts, they collect metrics such as CPU, memory, disk usage, and application status.
- **Zabbix Proxy (optional) :** Acts as an intermediary to collect data from remote locations and forward it to the server, reducing network load.
- **Dashboards/Frontend :** Web interface that visualizes metrics, alerts, and trends for administrators.

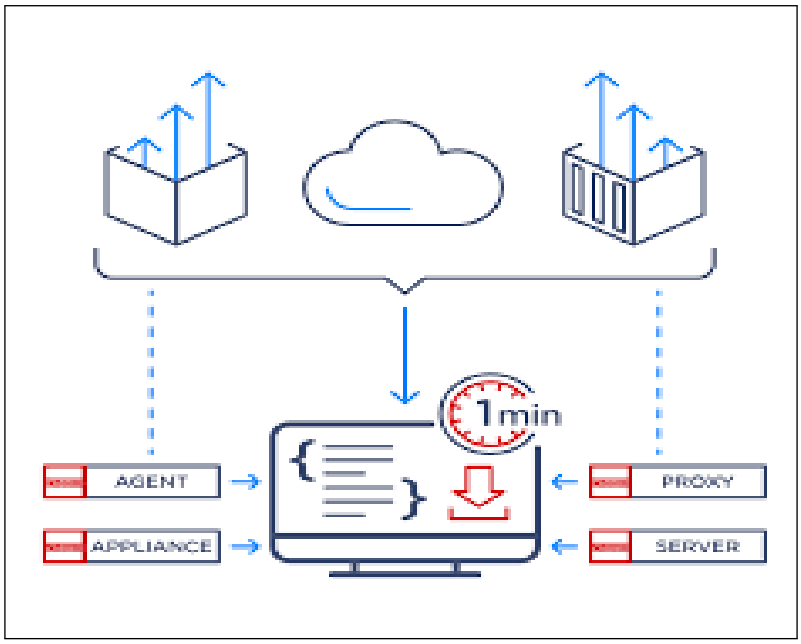


FIGURE 2.1 – zabbix Architecture

The system works by agents (or agentless checks) sending data to the server, which processes it, triggers alerts if thresholds are exceeded, and displays results on dashboards.

2.3.2 Wazuh (SIEM Solution)

Wazuh is an open-source Security Information and Event Management (SIEM) and security monitoring platform that provides organizations with comprehensive visibility into their systems and networks. It integrates log analysis, intrusion detection, vulnerability assessment, file integrity monitoring, and compliance management into a unified solution. By combining these capabilities, Wazuh enables both proactive and reactive approaches to cybersecurity.

Its architecture is composed of :

— **Wazuh Manager :** The central server responsible for receiving, processing, and analyzing data from agents. It handles rule evaluation, alert generation, and security event correlation.

— **Wazuh Agents :** Lightweight agents installed on monitored hosts to collect logs, monitor file integrity, detect vulnerabilities, and check compliance with defined policies.

— **Indexer (Elasticsearch) :** Stores and indexes all collected events, enabling fast queries and advanced analytics over large volumes of data.

— **Wazuh Dashboard (Kibana) :** A web-based interface that allows administrators to explore and visualize data, build custom dashboards, and investigate incidents in detail.

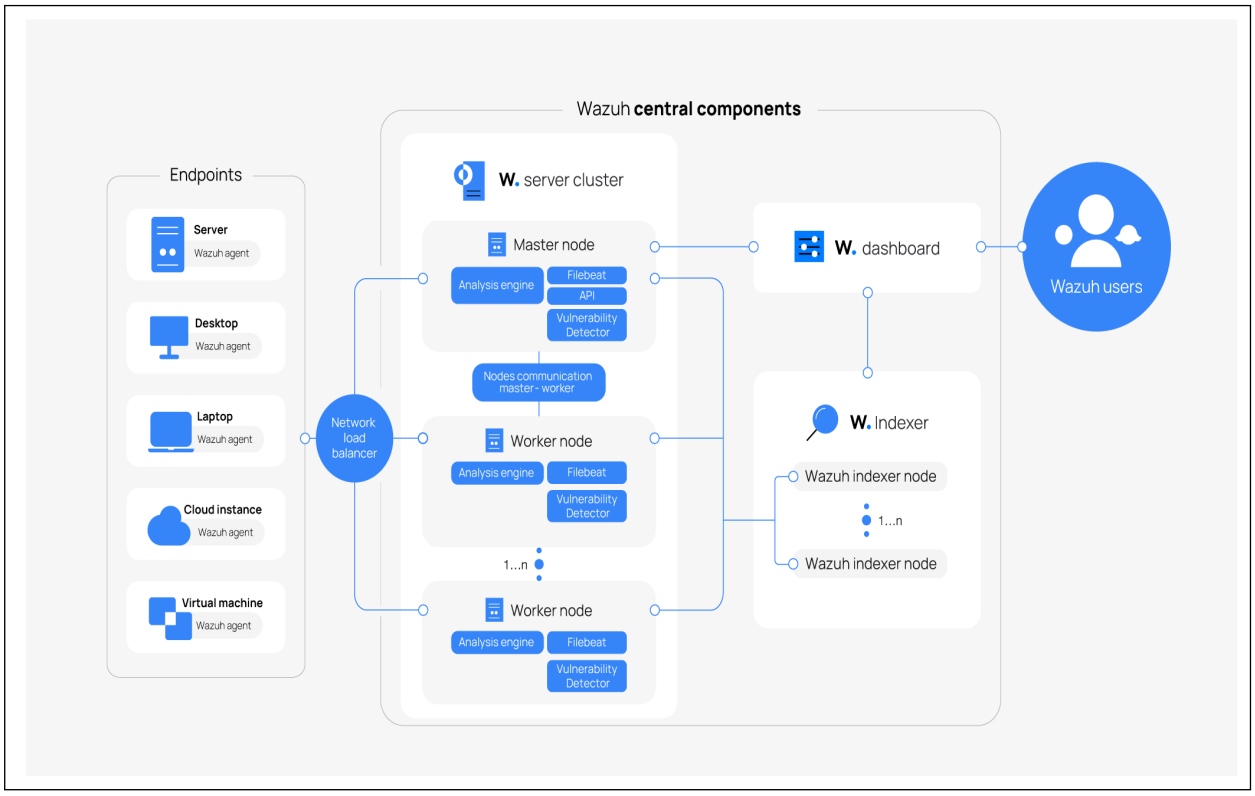


FIGURE 2.2 – Wazuh Architecture

Data flows from agents to the manager, where events are analyzed and sent to the indexer for storage. Administrators then use Kibana to view alerts and monitor the security status of the infrastructure.

Beyond its modular architecture, Wazuh stands out as a robust solution for several reasons :

— **Open-source and cost-effective :** Unlike many commercial SIEMs, Wazuh is free to use, making it accessible to organizations of different sizes while still offering enterprise-grade features.

— **Scalability :** It can handle large environments with thousands of agents, making it suitable for both small deployments and enterprise infrastructures.

— **Flexibility and integration :** Wazuh integrates seamlessly with other security and monitoring tools (such as Zabbix, Suricata, or cloud services) and supports multiple operating systems and platforms.

— **Regulatory compliance :** It includes predefined rules and reports that help organizations meet requirements for standards such as PCI DSS, HIPAA, GDPR, and ISO 27001.

— **Community and ecosystem :** Being open-source, Wazuh benefits from a large and active community that contributes improvements, updates, and integrations, ensuring the solution evolves rapidly with the cybersecurity landscape.

In summary, Wazuh is not just a SIEM but a complete security monitoring platform that combines detection, analysis, and compliance capabilities. Its balance of advanced features, scalability, and cost-effectiveness makes it a strong candidate for organizations seeking to strengthen their security posture without the high costs of proprietary solutions.

2.3.3 Ansible (Automation Tool)

Ansible is an open-source automation platform that simplifies configuration management, application deployment, and IT orchestration. Its architecture, as shown in Figure 3.1, is built around the **Ansible Orchestration Engine**, which serves as the central component. The engine interacts with various inputs and produces outputs across multiple environments.

— **Inputs :**

  — **Public/Private Cloud :** Infrastructure resources managed through Ansible.

  — **Users :** Administrators or operators who define automation tasks.

  — **Playbooks :** YAML-based instructions that describe desired configurations and workflows.

— **Core Components of the Orchestration Engine :**

  — **Inventory :** A list of managed hosts and groups.

  — **API :** Interfaces that enable integration with external tools and services.

  — **Modules :** Predefined units of code that perform specific tasks (e.g., install packages, configure services).

  — **Plugins :** Extensions that add functionality such as logging, connection handling, or caching.

— **Outputs :**

— **Linux, Windows, iOS :** Operating systems that can be configured and automated by Ansible.

— **Networking :** Network devices and services that can be managed through automation.

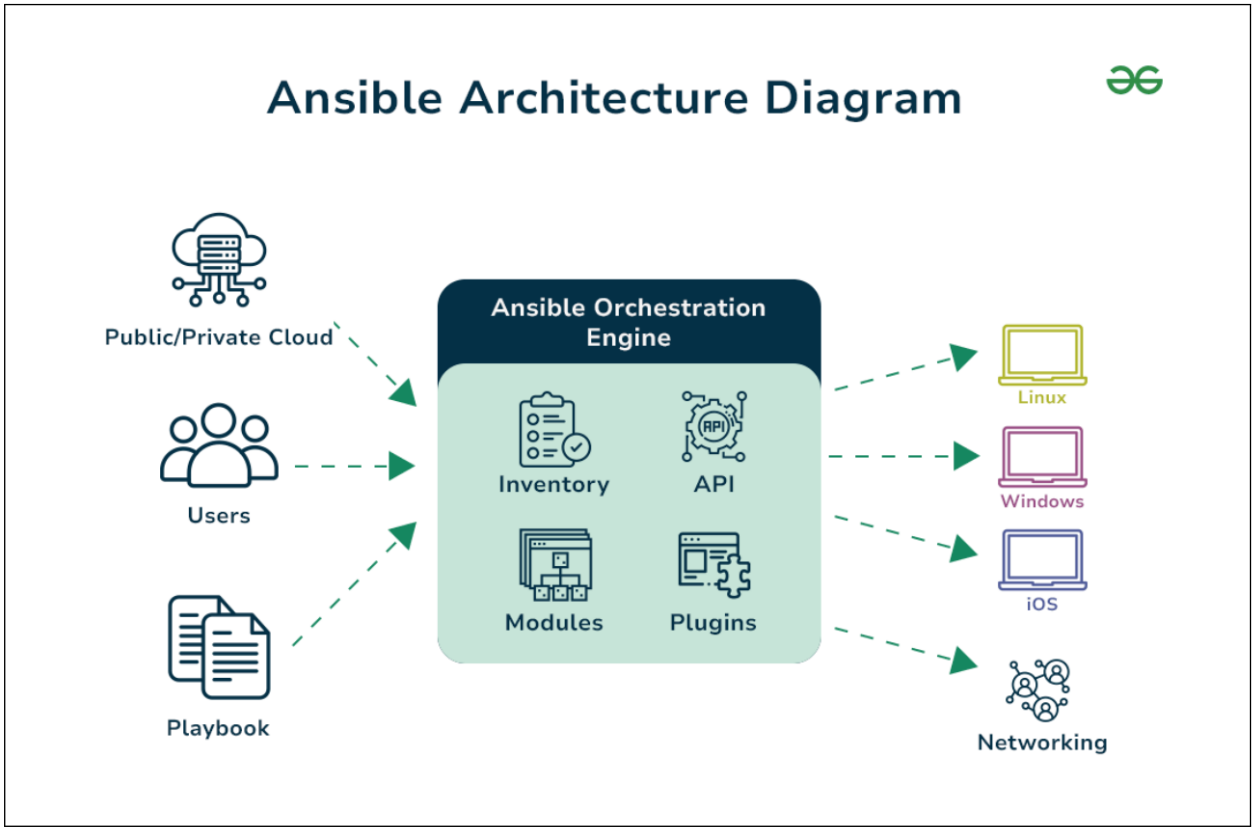


FIGURE 2.3 – Ansible Architecture Diagram

Ansible operates by processing playbooks and inputs through its orchestration engine, which uses modules and plugins to interact with target systems. This design ensures a flexible, agentless approach to automation across heterogeneous environments.

2.3.4 Integration of Monitoring, Security, and Automation

Combining monitoring, SIEM, and automation offers a comprehensive approach to infrastructure management. Monitoring ensures system availability and performance, SIEM provides visibility into security events and threats, and automation tools like Ansible streamline the deployment and management of monitoring and security agents. Together, these components significantly enhance reliability, security, and operational efficiency.

2.4 Technical Environment of the Internship

2.4.1 Cloned Development Infrastructure

The internship was carried out in a cloned version of the Aurora development environment(Aurora is the main SaaS application DOT'COM is developing). This clone reproduced the architecture of the production system while ensuring isolation and confidentiality. The environment consisted of several layers, including :

— A reverse proxy and load balancer based on Nginx.

— Web servers responsible for serving static and dynamic content.

— An API gateway used for routing client requests toward backend services.

— A PostgreSQL database cluster accessed through load balancers and administered using *pgAdmin*.

To respect confidentiality, only the components involved in the monitoring setup are detailed in this report.

2.4.2 Monitored Components

The monitoring activities focused on two main parts of the cloned environment :

1. **Database Layer :** PostgreSQL instances were monitored to ensure service availability and to track resource utilization (CPU, memory, disk usage) as well as performance metrics.
2. **Application Layer :** Nginx web servers were monitored to verify their availability, process status, and performance indicators.

2.4.3 Deployment of Monitoring Tools

A dedicated **Zabbix server** was installed on a virtual machine (VM) within the same LAN as the cloned infrastructure. Zabbix agents were then deployed on the selected VMs (PostgreSQL

and Nginx servers), enabling the collection of performance data and service availability metrics. The collected information was visualized in real time through the Zabbix dashboard.

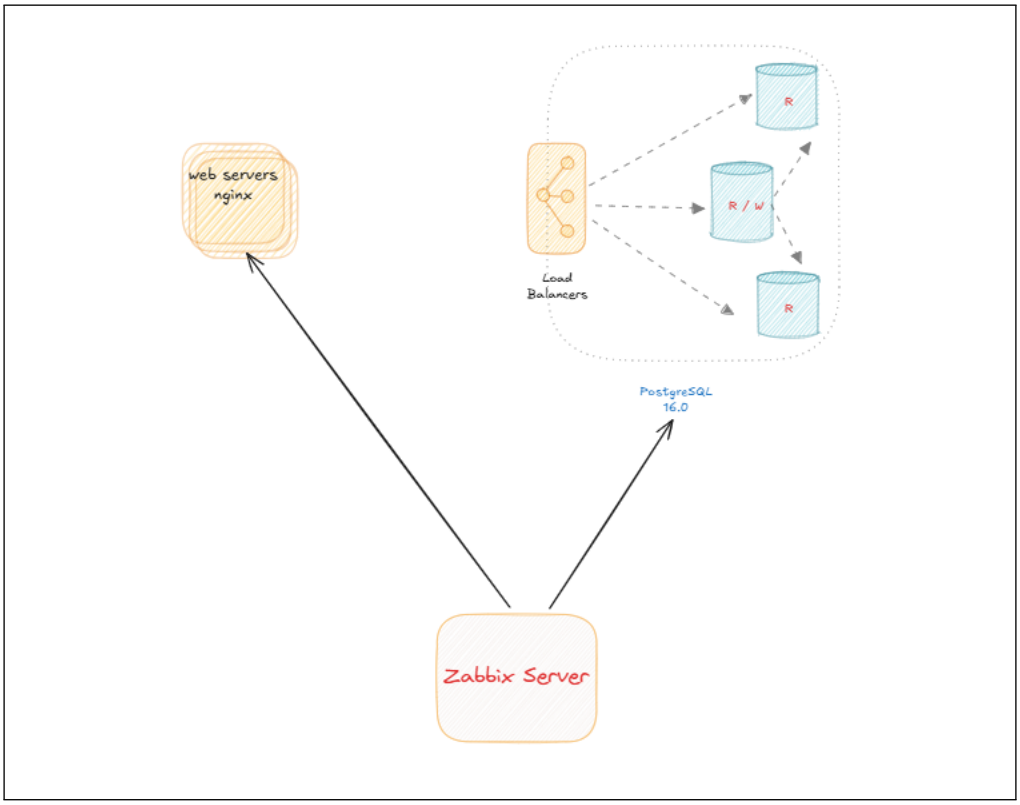


FIGURE 2.4 – Dedicated monitoring of PostgreSQL and Nginx servers using Zabbix

2.4.4 Extension with Security and Automation

In addition to infrastructure monitoring, a separate **Wazuh server** was installed to test SIEM functionalities such as log analysis, intrusion detection, and vulnerability monitoring. For this purpose, a new test VM was provisioned where a Wazuh agent was deployed.

To streamline agent deployment, an **Ansible master node** was configured on another VM. Using Ansible playbooks, the Wazuh agent installation was automated, ensuring reproducibility and consistency across multiple hosts.

2.4.5 Technical Overview

The resulting environment integrated both monitoring and security solutions :

— Cloned Aurora infrastructure (databases and application servers).

— Zabbix server VM for infrastructure monitoring.

— Wazuh server VM for SIEM functionalities.

— Ansible VM for automating Wazuh agent deployment.

— Test VM for validating SIEM functionalities.

This setup provided a realistic yet isolated environment for experimenting with monitoring, incident detection, and automation solutions.

Conclusion

This chapter has outlined the theoretical and technical foundations required to understand the internship project. It first introduced essential cybersecurity principles, emphasizing the importance of protecting information systems against diverse threats through monitoring, detection, and response mechanisms. It then presented monitoring and event management concepts, highlighting the role of SIEM in centralizing and correlating security data. Finally, the chapter detailed the main tools—Zabbix for infrastructure monitoring, Wazuh for security information and event management, and Ansible for automation—that formed the technological basis of the project.

By combining these tools and concepts, it becomes possible to build an integrated approach that improves operational efficiency, strengthens security, and facilitates the management of complex infrastructures. These theoretical insights provide the groundwork for the following chapter, which focuses on the practical implementation of the project.

Chapter

3 Implemented Solutions

Introduction

This chapter presents the practical solutions implemented during the internship. It describes the deployment and configuration of different tools that were used to enhance the monitoring and security of the infrastructure. In particular, it details the setup of the Zabbix monitoring system, the deployment of the Wazuh SIEM solution for detecting potential data exfiltration, and the automation of repetitive tasks using Ansible. Each subsection highlights the installation, configuration, and integration of these tools within the existing environment.

3.1 Zabbix Monitoring Setup

3.1.1 Installation and Initial Configuration

The Zabbix monitoring system was deployed on a dedicated virtual machine to ensure optimal performance and isolation from other services. The deployment process began with the installation of the Zabbix repository, followed by the installation of the server, frontend, and agent components. PostgreSQL was selected as the backend database due to its robustness and compatibility with the existing infrastructure. The database was created with a dedicated user, and proper permissions were assigned to ensure secure access by the Zabbix server.

Once the server and database were operational, the web-based frontend was configured. Initial parameters, including the default administrative user, authentication methods, and basic system settings, were established. This setup allowed for immediate access to the Zabbix interface, enabling the subsequent addition and configuration of monitored hosts. The installation and

initial configuration ensured that the system was ready for agent deployment and the monitoring of key infrastructure components.

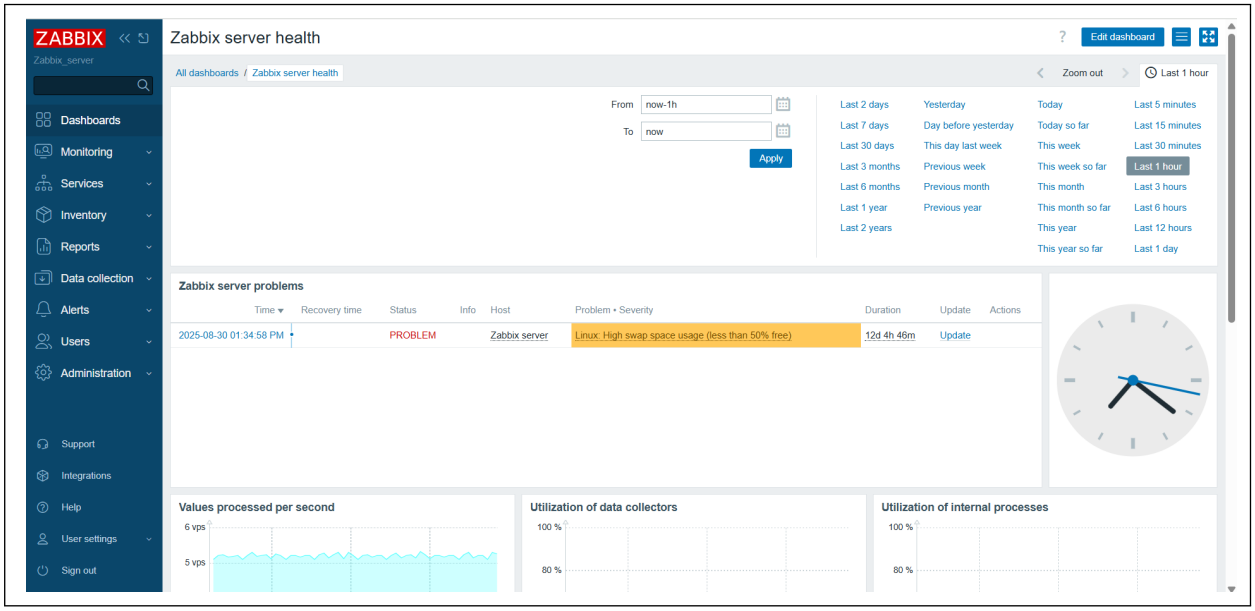


FIGURE 3.1 – Zabbix Web Interface showing the initial dashboard and configuration.

The figure above illustrates the Zabbix web interface after the initial setup and preliminary configuration. It provides a centralized view of the monitoring system, allowing administrators to interact with the server, manage hosts, and prepare for monitoring activities. Although the interface reflects some initial adjustments, it serves as a foundation for adding agents, creating dashboards, and configuring alerts for effective infrastructure monitoring.

3.1.2 Deployment of Zabbix Agents

Zabbix agents were deployed on selected virtual machines, including the Nginx server, PostgreSQL server, Docker host, and Wazuh server, to enable real-time monitoring of system performance and service availability. The agents serve as intermediaries, collecting metrics from the hosts and securely transmitting them to the Zabbix server for processing and visualization.

Agent Installation and Configuration

The Zabbix agent was installed on each host using the official Zabbix packages. After installation, the agent service was enabled to start automatically upon system boot. The configuration

file, `zabbix_agentd.conf`, was customized to specify the Zabbix server address, the hostname of the monitored host, and the parameters for data collection and communication. Encryption and authentication options were configured to ensure secure transmission of monitoring data.

Active vs Passive Agents

Zabbix supports two types of agent communication :

— **Passive Agent :** The Zabbix server polls the agent to request monitoring data. This method is suitable for environments where the server initiates all checks and can control the frequency of data collection.

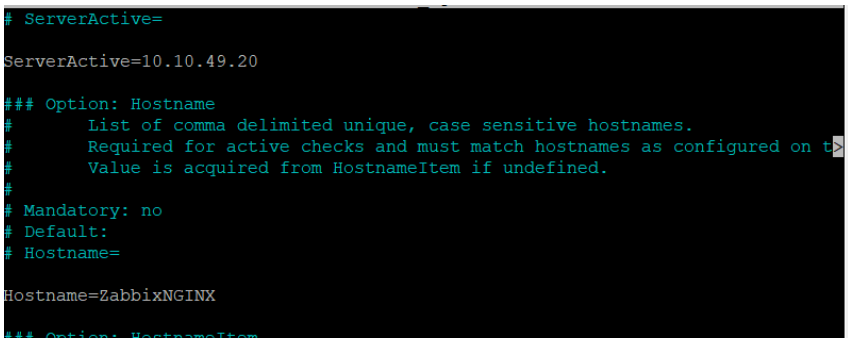
— **Active Agent :** The agent periodically sends data to the Zabbix server without being polled. Active mode is preferred in environments with firewalls or NAT, where incoming connections to the agent may be restricted.

Both types were deployed depending on the network configuration and the nature of the monitored host. Passive agents were mainly used on internal servers with direct connectivity to the Zabbix server, while active agents were used for hosts behind firewalls or with limited incoming network access.

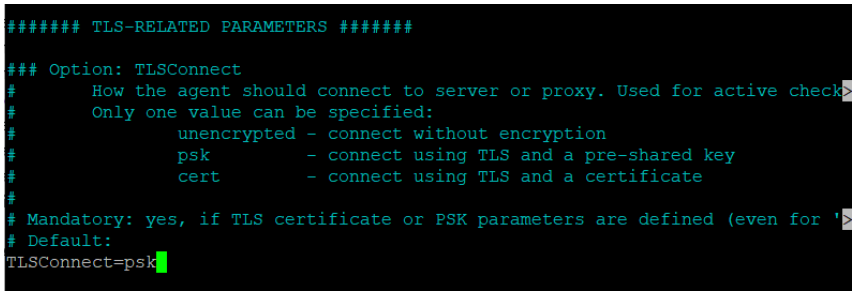
Verification

The successful deployment of agents was confirmed by checking the agent status on each host and verifying that the hosts appeared in the Zabbix web interface as active and reporting data.

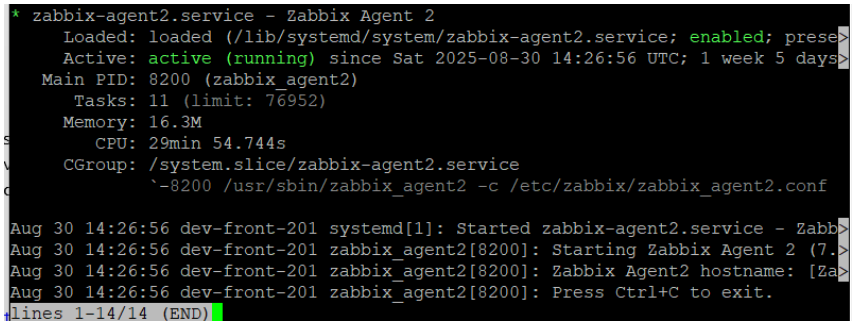
The figure above shows an example of a `zabbix_agentd.conf` file, illustrating key parameters such as server address, hostname, and enabled checks. This configuration ensures that each agent communicates correctly with the Zabbix server and monitors the desired services effectively.



(a) Customized `zabbix_agentd.conf` file.



(b) TLS Encryption config for secure communication.



(c) Zabbix agent status on the host.

(d) Host visible in Zabbix web interface after agent deployment.

FIGURE 3.2 – Overview of Zabbix agent deployment : configuration, agent status, and host registration.

3.1.3 Monitoring of key Components

3.1.3.1 Monitoring of Nginx Server

The Nginx server was monitored using a Zabbix agent installed directly on the machine with IP address `10.10.49.201`. This agent (Agent 1) was configured to collect system and service metrics and send them to the Zabbix server for visualization and alerting.

The following predefined templates were applied to the host :

— **Linux by Zabbix Agent :** Provides monitoring of standard Linux system metrics such as CPU, memory, disk usage, and network statistics.

— **Nginx by Zabbix Agent :** Specifically monitors the Nginx service, including process status, server availability, and request statistics.

The host was registered in Zabbix with the following parameters :

— **Hostname :** ZabbixNGINX

— **Host group :** Linux Servers

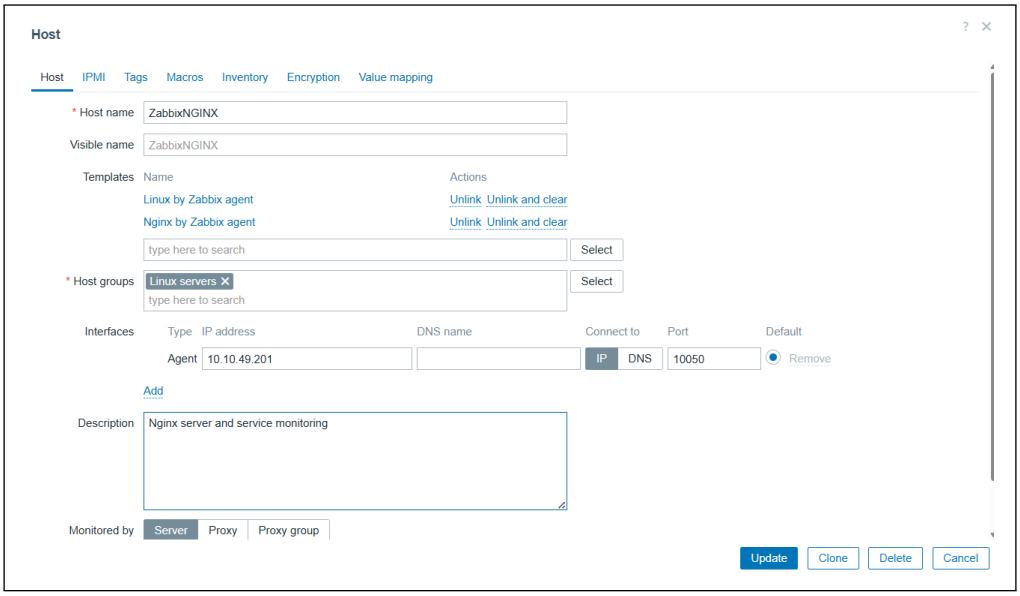


FIGURE 3.3 – Zabbix host configuration for the Nginx server .

This configuration allows the Zabbix server to monitor both the underlying Linux system and the Nginx service, ensuring the availability and performance of the web server are continuously tracked.

3.1.3.2 Monitoring of PostgreSQL and Docker Services

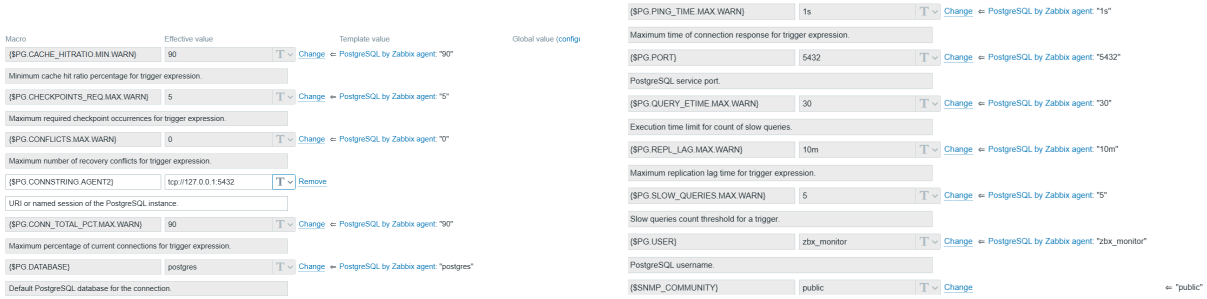
On one of the virtual machines, both **Zabbix Agent (v1)** and **Zabbix Agent 2** were installed in order to collect metrics from different services. The choice to deploy both agents on the same host was motivated by their complementary capabilities : **Zabbix Agent** natively supports

PostgreSQL monitoring, while **Zabbix Agent 2** extends functionality with plugins such as Docker monitoring. This setup ensured that both the database and its containerized environment were monitored efficiently from a single machine.

a) PostgreSQL Monitoring

The PostgreSQL instance was deployed inside a Docker container. To allow Zabbix to collect database-level metrics, a dedicated user (`zbx_monitor`) was created with the `pg_monitor` role. Authentication was automated using a `.pgpass` file, which enabled secure, passwordless access for the agent. In the Zabbix frontend, the host was linked with the template :

— **PostgreSQL by Zabbix Agent :** Provided monitoring of key metrics such as active connections, cache hit ratio, query statistics, and database uptime.



(a) PostgreSQL macros (part 1). **(b) PostgreSQL macros (part 2).**

FIGURE 3.4 – Macros configuration for PostgreSQL monitoring.

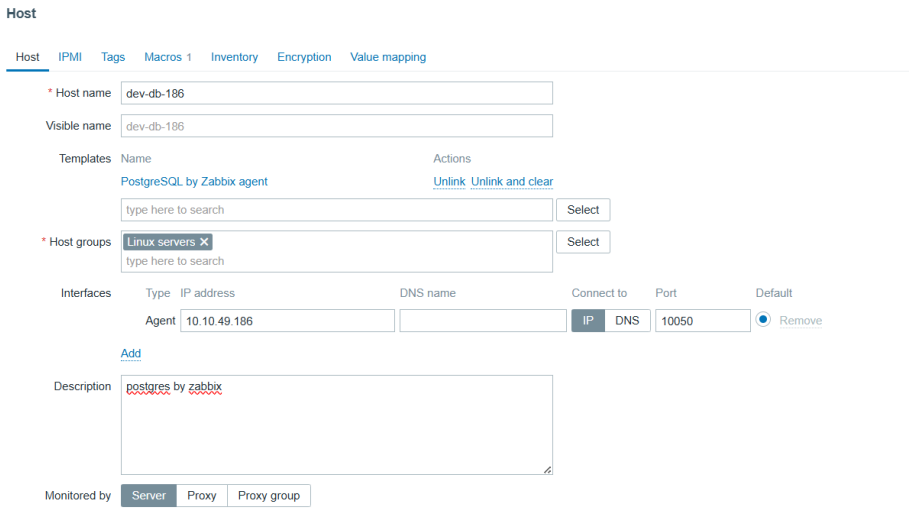


FIGURE 3.5 – Zabbix host configuration and template linking for PostgreSQL monitoring.

b) Docker Monitoring

In parallel, Zabbix Agent 2 was used to monitor the Docker environment. By enabling the Docker plugin and configuring it to communicate with the Docker socket (`/var/run/docker.sock`), the agent was able to collect container-level metrics. The following template was applied :

— **Docker by Zabbix Agent 2 :** Provided visibility into container states, resource utilization (CPU, memory), and Docker engine health.

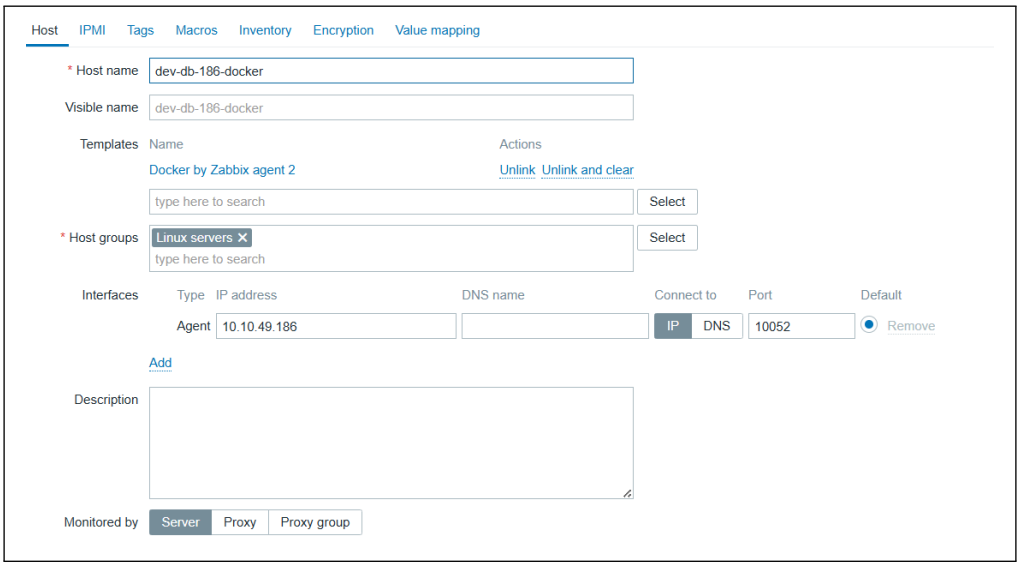


FIGURE 3.6 – Zabbix host configuration for Docker monitoring.

c) Integration Benefits

By deploying both Zabbix Agent (for PostgreSQL) and Zabbix Agent 2 (for Docker) on the same underlying virtual machine, it was possible to create two distinct hosts in Zabbix, each linked to its appropriate template and configuration. This design provided comprehensive monitoring coverage, enabling PostgreSQL performance metrics and Docker container statistics to be collected simultaneously, without requiring additional infrastructure. The separation into two hosts also facilitated clearer visualization and alerting, as each service could be tracked independently while still running on the same machine.

3.1.3.3 Monitoring of the Zabbix Server

The Zabbix server, being the central component of the monitoring infrastructure, was also monitored to ensure its stability and performance. An agent was installed directly on the Zabbix server host to collect system-level metrics in addition to the built-in server health checks provided by Zabbix.

By default, Zabbix provides internal metrics related to the server's health, such as the number of processed values per second, queue size, availability of pollers, and cache usage. To strengthen this monitoring, the template **Linux by Zabbix Agent** was also linked to the Zabbix server host. This template enabled a more detailed visualization of operating system resources, including :

— CPU and memory utilization of the Zabbix server machine.

— Disk usage and filesystem performance.

— Network statistics related to incoming and outgoing traffic.

This configuration ensured that both the internal performance of the Zabbix server application and the underlying system resources were continuously tracked. As a result, administrators gained comprehensive visibility over the monitoring infrastructure itself, reducing the risk of undetected failures within the core monitoring platform.

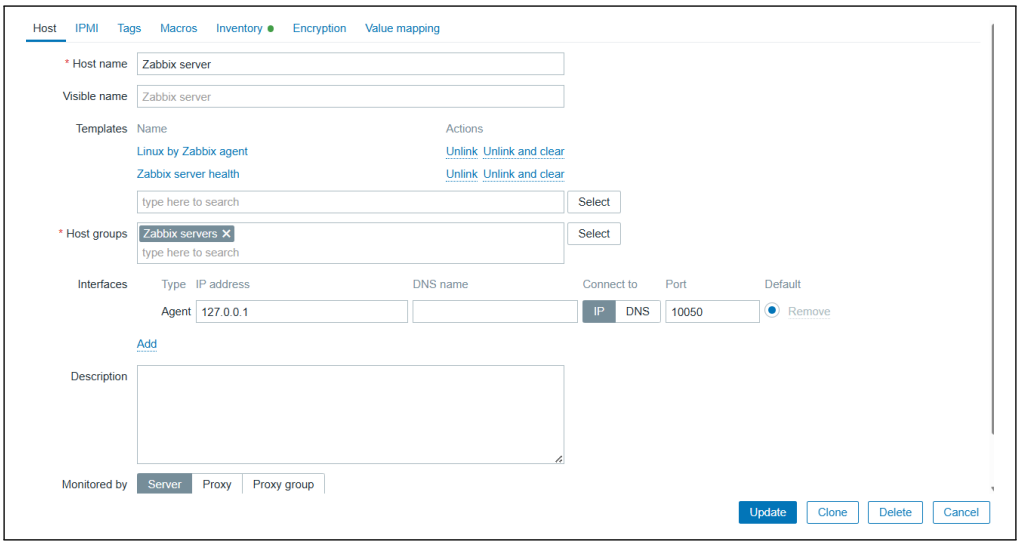


FIGURE 3.7 – Zabbix host configuration for the Zabbix server.

3.2 Wazuh SIEM Implementation

3.2.1 Installation and Initial Setup

The Wazuh SIEM solution was deployed using a **single-node installation** on a dedicated virtual machine. A single-node (all-in-one) setup means that the **Wazuh Manager**, the **Wazuh Indexer**, and the **Wazuh Dashboard** components are installed on the same host. This configuration is recommended for testing environments and small-scale infrastructures, as it simplifies the deployment while still providing the full set of Wazuh functionalities.

In addition to the manager node, a separate virtual machine was provisioned to host the **Wazuh Agent**. The agent was installed to collect security events from the monitored system and forward them securely to the manager for analysis and correlation.

The connection between the agent and the manager was established through the Wazuh agent configuration file (`/var/ossec/etc/ossec.conf`). The following excerpt shows the essential parameters used to link the agent to the manager :

```
<client>
  <server>
    <address>10.10.50.220</address>
    <port>1514</port>
    <protocol>tcp</protocol>
  </server>
  <config-profile>debian, debian7</config-profile>
  <notify_time>10</notify_time>
  <time-reconnect>60</time-reconnect>
  <auto_restart>yes</auto_restart>
  <crypto_method>aes</crypto_method>
</client>
```

FIGURE 3.8 – Wazuh client connection to the server via agent config

After configuring the agent, it was registered with the manager using the `agent-auth` utility, which enables the establishment of a secure communication channel between the two components. Once registered, the agent began sending logs and events to the manager, where they were indexed and visualized through the Wazuh dashboard.

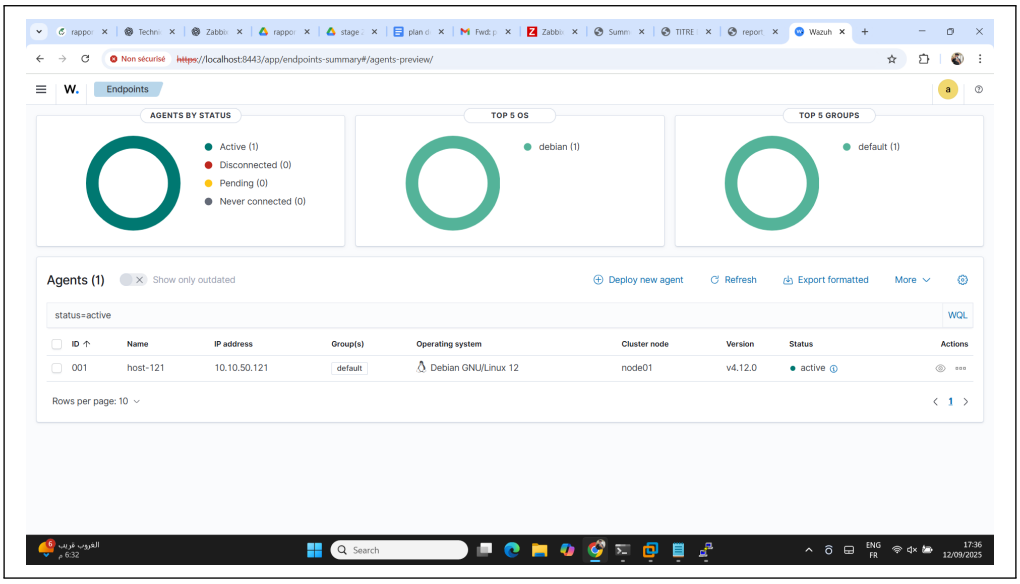


FIGURE 3.9 – Wazuh dashboard after the initial single-node setup and agent registration.

3.2.2 Core Modules Activated

3.2.2.1 Vulnerability Detection

The Vulnerability Detection module was activated to enhance the security posture of the monitored systems by identifying known vulnerabilities in system packages and installed applications. This module leverages continuous feeds of Common Vulnerabilities and Exposures (CVEs), ensuring that the detection engine remains up-to-date with the latest security threats.

In our implementation, the module was configured to scan critical system components periodically and provide alerts when vulnerabilities were discovered. This proactive monitoring allows for timely remediation of identified weaknesses, thereby reducing the risk of exploitation.

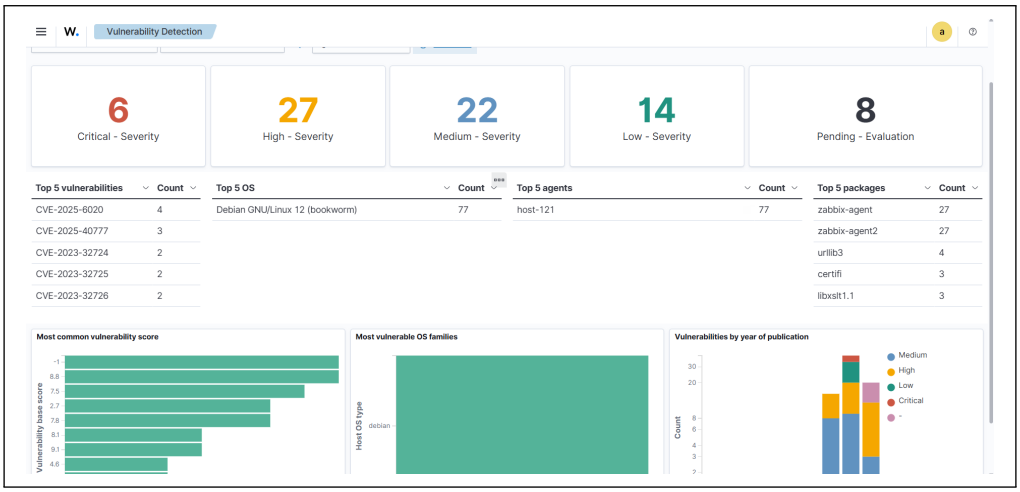


FIGURE 3.10 – Wazuh dashboard for Vunerability Detection.

A screenshot of the vulnerability detection dashboard is provided in Figure 3.10, illustrating detected vulnerabilities and their severity levels.

3.2.2.2 File Integrity Monitoring (FIM)

The File Integrity Monitoring module was implemented to safeguard the integrity of critical system files and directories. It continuously tracks changes such as modifications, deletions, and creations, which could indicate unauthorized activities or system misconfigurations.

In our configuration, particular attention was given to sensitive directories including `/etc/` and `/usr/bin`, as they contain vital configuration files and logging information. Real-time alerts were enabled to ensure that any suspicious file activity could be promptly detected and investigated. This contributes to strengthening system resilience against both external threats and insider misuse.

3.2.2.3 SSH Brute-force Detection

The SSH Brute-force Detection module was enabled to protect the monitored infrastructure from unauthorized access attempts through repeated login failures. Brute-force attacks are a common technique used by attackers to gain entry by systematically guessing user credentials, which makes their early detection critical.

In our setup, the built-in Wazuh ruleset was activated to identify patterns of repeated failed login attempts over SSH. Thresholds were defined to trigger alerts when the number of unsuccessful connections exceeded acceptable limits within a specific time frame. This mechanism provides timely warnings of potential brute-force activity and allows administrators to take preventive actions, such as blocking suspicious IP addresses or reinforcing authentication mechanisms.

3.2.2.4 Command and Privilege Auditing

To secure the monitoring infrastructure itself, this module was applied specifically to the Wazuh manager virtual machine. As the core of the environment, this server should not be used for frequent interactive operations ; therefore, auditing all commands executed on it is essential to detect any misuse or unauthorized activity.

For the implementation, the `auditd` service was installed and configured with rules to capture every command executed on the system. The generated audit logs were then collected and analyzed by Wazuh, enabling the creation of alerts for sensitive or unexpected activity. This setup ensured complete visibility into administrative actions performed on the server and strengthened accountability by associating each command with the user who executed it.

3.2.2.5 Malware Detection

To complement vulnerability and integrity monitoring, the malware detection capability of Wazuh was enabled. This module scans files on monitored systems to identify malicious software, backdoors, or trojans that may compromise system security.

In our configuration, periodic scans were scheduled on critical directories, while real-time monitoring ensured that newly created or modified files were also analyzed. The detection mechanism relies on updated malware signatures, allowing the system to identify known threats effectively. This proactive approach strengthens endpoint protection and provides early warnings of potential infections within the environment.

3.2.2.6 Integration with VirusTotal

As an additional layer of analysis, Wazuh was integrated with the VirusTotal online threat intelligence service. This integration enables suspicious files and indicators of compromise (IoCs), such as file hashes, to be automatically checked against a large database of known malware and malicious artifacts.

Through this setup, Wazuh enhances its detection capabilities by leveraging external intelligence feeds, ensuring that alerts related to suspicious files are validated against up-to-date global threat information. This provides security operators with enriched context for decision-making and helps prioritize incident response actions.

3.2.3 Ansible for Wazuh Agent Deployment

To automate the installation and configuration of Wazuh agents, Ansible was employed as the configuration management tool. A dedicated user named `ansible` was created on both the controller and the target hosts, allowing automation tasks to be executed securely and consistently. Communication between the machines was established via SSH, with the controller's public key added to the target hosts' authorized keys, enabling passwordless execution of playbooks.

The automation was organized around a modular Ansible role named `wazuh_agent`, which encapsulates all the tasks required to deploy and configure an agent. The role is structured as follows :

- **defaults/** and **vars/** : store default and role-specific variables, enabling easy customization of parameters such as the Wazuh manager IP, ports, and package names.
- **tasks/** : contains modular task files, including repository setup, package installation, agent configuration, registration with the manager, and service management. The `main.yml` file orchestrates the execution of these subtasks in a defined sequence.
- **handlers/** : defines service-related actions, such as restarting the agent when configuration changes occur.

— **templates/** : includes template files like `ossec.conf.j2`, allowing dynamic configuration of agents according to the environment.

Two main playbooks were developed to leverage this role : `deploy_agent.yml`, which orchestrates the full agent deployment, and `deploy_ssh_key.yml`, which sets up SSH key-based authentication between the controller and hosts.

The inventory is organized to reflect the monitored infrastructure, with host definitions in the `hosts` file and group-specific variables stored in `group_vars/wazuh_agents.yml`. This structure allows scaling the deployment to multiple hosts while maintaining clear separation of configuration data and operational logic.

Figure 3.11 illustrates the directory structure of the `wazuh_agent` role, highlighting its modular organization and maintainable design.

```
roles/
└── wazuh_agent/
    ├── defaults/
    │   └── main.yml        # Default variables (lowest priority)
    ├── vars/
    │   └── main.yml        # Role variables (higher priority)
    ├── tasks/
    │   ├── main.yml        # Entry point orchestrating subtasks-
    │   ├── repo.yml        # Add repo and keys
    │   ├── install.yml     # Install packages
    │   ├── config.yml      # Template config files
    │   ├── register.yml    # Agent registration logic
    │   └── service.yml     # Manage wazuh service
    ├── handlers/
    │   └── main.yml        # Handlers like restart service
    └── templates/
        └── ossec.conf.j2  # Agent config template
```

FIGURE 3.11 – Directory structure of the Ansible role for Wazuh agent deployment

Conclusion

In this chapter, the monitoring and security infrastructure was successfully deployed and configured. Zabbix agents were installed across key hosts to provide comprehensive visibility

into system and service performance, while Wazuh SIEM was implemented to detect vulnerabilities, monitor file integrity, and identify anomalous activities. The automation of Wazuh agent deployment using Ansible ensured a repeatable and scalable process, reducing manual effort and the risk of configuration errors.

These implementations established a solid foundation for system monitoring and security enforcement, paving the way for the next chapter, which presents the results and effectiveness of these deployments in the monitored environment.

Chapter 4

Results and Analysis

Introduction

This chapter presents the results of the implemented monitoring, security, and automation solutions. It highlights key Zabbix dashboards, Wazuh responses to events and attacks, and the outcomes of automating Wazuh agent deployment with Ansible, demonstrating the effectiveness and reliability of the deployed infrastructure.

4.1 Host Monitoring Results

After configuring all hosts and templates in Zabbix, custom dashboards and graphs were created to provide a clear overview of system and application health, resource usage, and service performance. Only the most representative dashboards are shown for each host to highlight the effectiveness of the monitoring setup.

a) Zabbix Server Dashboards

The Zabbix server host is monitored to ensure both the performance of the server itself and the stability of the monitoring infrastructure. The following dashboards highlight key aspects of system and service health :

— **Zabbix Server Health :** Provides a snapshot of the monitoring system's internal operations, including the number of active checks, agent status, and server uptime. This dashboard confirms that the server is correctly collecting and processing data from all monitored hosts.

— **System Performance :** Displays CPU utilization, memory usage, disk space, and overall server load. Monitoring these metrics ensures that the Zabbix server has sufficient resources to handle all monitoring tasks without bottlenecks.

— **Network Interfaces :** Shows bandwidth usage and packet statistics for each network interface. This allows administrators to detect network saturation, packet loss, or unusual traffic patterns that could impact monitoring reliability.

— **Filesystems :** Monitors disk usage and I/O statistics for critical filesystems. This ensures that the server has sufficient storage for logs, database data, and configuration files, preventing service interruptions due to full disks.

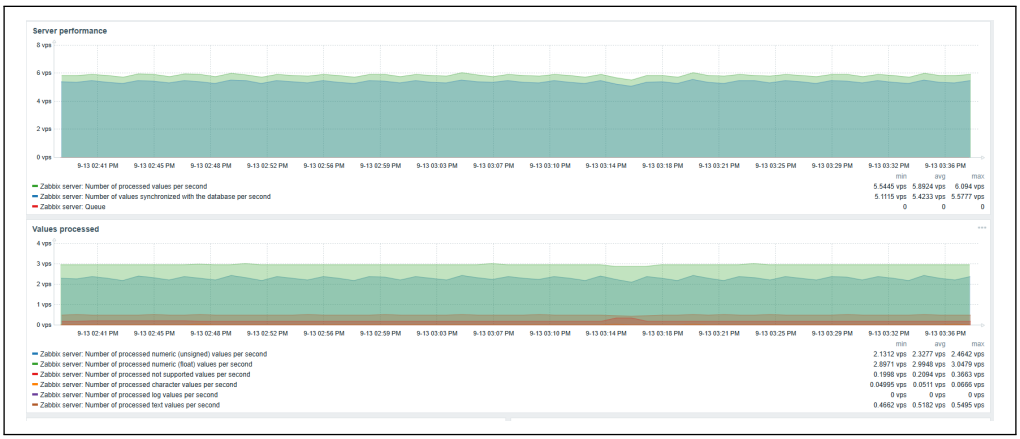


FIGURE 4.1 – Zabbix Server Health Dashboard : shows active checks, agent status, and server uptime.

FIGURE 4.2 – System Performance : CPU, memory, disk usage, and overall server load.

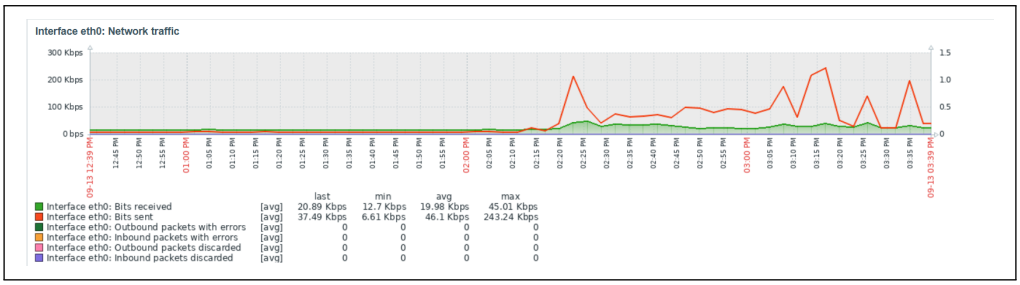


FIGURE 4.3 – Network Interfaces : Network Traffic

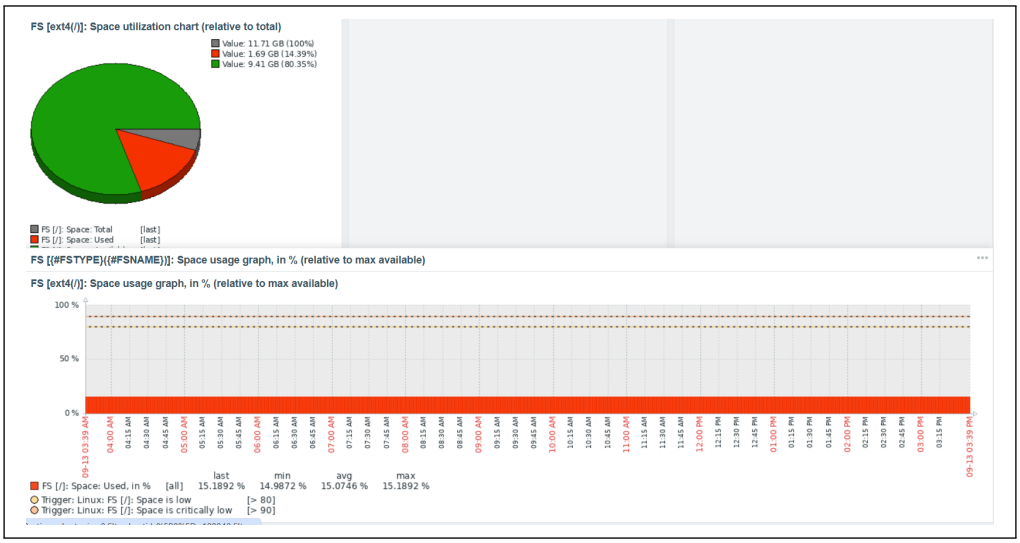


FIGURE 4.4 – Filesystems : disk usage.

Note that system-level dashboards (CPU, memory, disk usage, and network) for the other hosts are not shown, as they are similar to those already presented for the Zabbix server, and including them would be repetitive.

b) Nginx Server Dashboard

The Nginx server is monitored for web service activity using a single dashboard containing three key graphs :

— **Requests per Second :** Shows the number of HTTP requests handled by the server over time, indicating load and traffic patterns.

— **Connections per State :** Displays active connections broken down by state (reading, writing, waiting), helping identify potential bottlenecks.

— **Connections per Second :** Illustrates the rate of new connections, providing insight into server responsiveness and client activity.

FIGURE 4.5 – Nginx Server Dashboard with three graphs : requests per second, connections per state, and connections per second.

c) PostgreSQL Service Dashboard

The PostgreSQL server is monitored using the PostgreSQL Status dashboard, which provides a comprehensive overview of database service health and key performance indicators :

— **Connections :** Number of active connections, idle connections, and connections per state, helping identify load and potential bottlenecks.

— **Transactions :** Transaction metrics including active, prepared, and waiting transactions to assess database responsiveness.

— **Ping :** Database response time to monitor availability and latency.

— **Uptime and Cache Hit Ratio :** Tracks service uptime and cache efficiency for performance analysis.

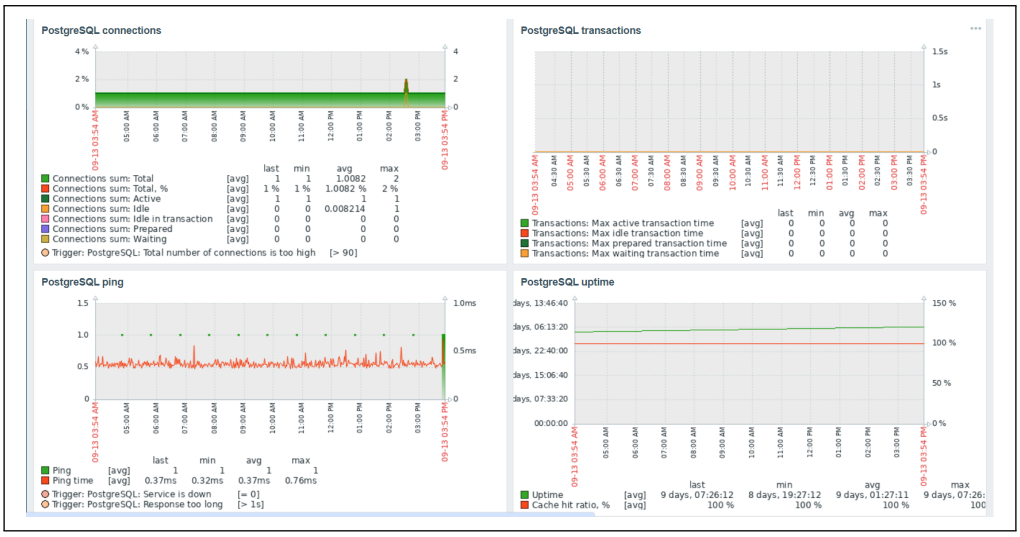


FIGURE 4.6 – PostgreSQL Status Dashboard showing service health, active connections, transactions, ping, and uptime.

d) Docker Dashboards

The PostgreSQL service is deployed inside a Docker container, and therefore monitoring the Docker host provides valuable insights into the performance and stability of the database environment. Three dashboards were selected to illustrate container resource consumption, network activity, and container/image statistics :

— **Container CPU and Memory Usage :** Tracks resource utilization of each running container to identify overloaded or inefficient containers.

— **Docker Network Traffic and Packets :** Monitors bandwidth usage and packet flow to detect abnormal traffic patterns or bottlenecks.

— **Container and Image Numbers, Goroutines :** Displays the number of active containers, images, and goroutines (lightweight threads), providing insight into host workload and Docker engine activity.

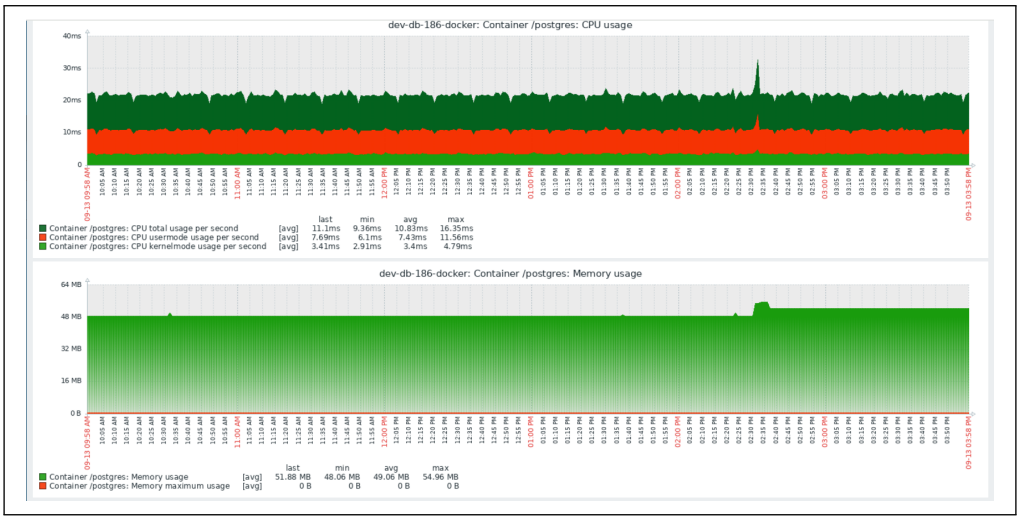


FIGURE 4.7 – Docker Dashboard : Container CPU and Memory Usage.

FIGURE 4.8 – Docker Dashboard : Network Traffic and Packets.

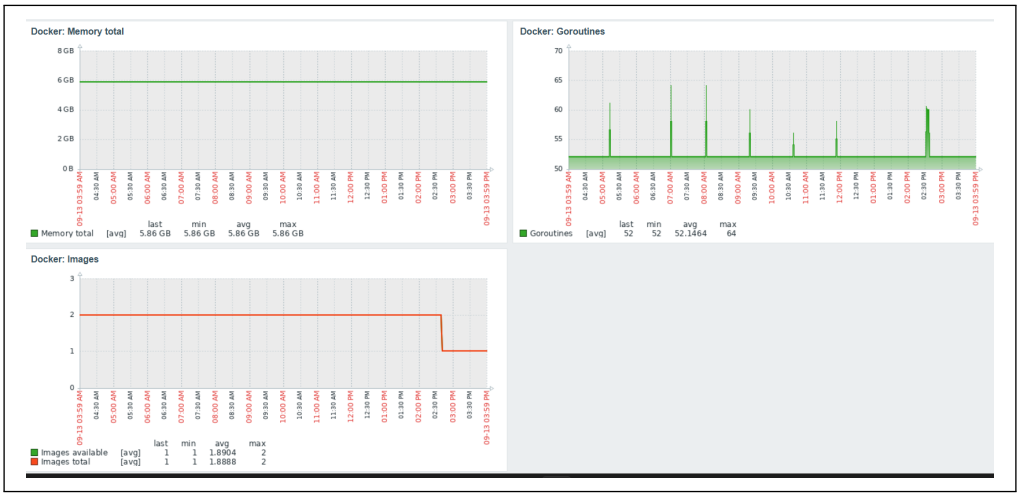


FIGURE 4.9 – Docker Dashboard : Containers, Images, and Goroutines.

e) Alerting and Trends

One of the strengths of Zabbix lies in its alerting capabilities, which ensure that administrators are promptly notified when a monitored resource reaches a critical state. Alerts are triggered based on pre-defined thresholds (known as triggers), each associated with a severity level ranging from *Information* to *Disaster*.

As an example, we configured an alert on the PostgreSQL virtual machine for memory usage. A trigger was defined to fire when memory consumption exceeds 90%, with a severity level set to *Average*. This allows early detection of resource saturation, ensuring corrective actions can be taken before performance degradation or service interruption occurs.

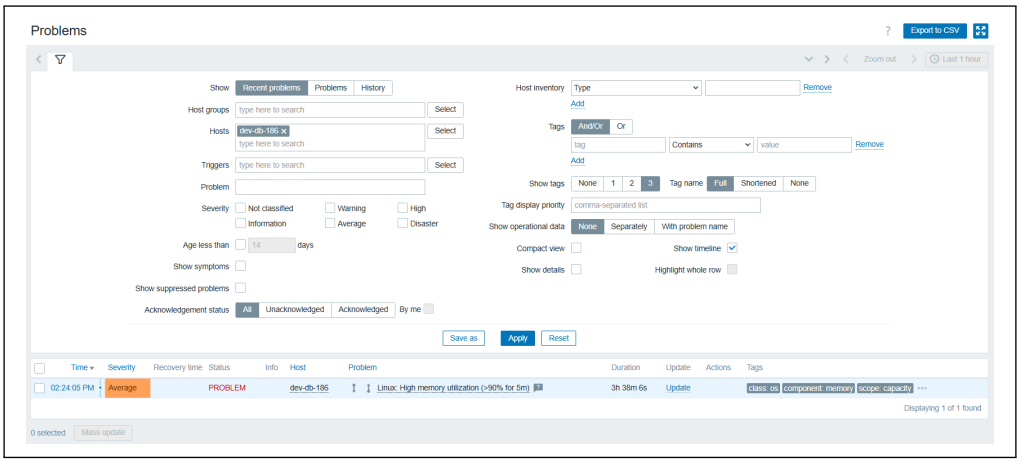


FIGURE 4.10 – Alert configured on the PostgreSQL VM : memory usage exceeds 90% (severity : Average).

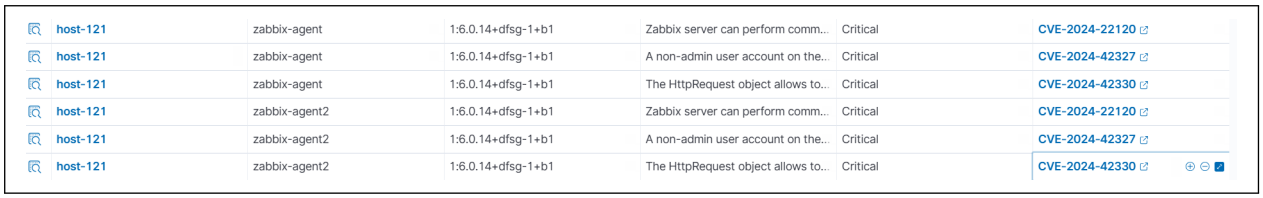
In addition to alerting, Zabbix provides trend analysis through historical data. This allows the visualization of resource usage patterns over time, which is essential for capacity planning and proactive system maintenance.

4.2 Wazuh SIEM Results

4.2.1 Vulnerability Detection

Wazuh includes a vulnerability detection module that scans installed applications and packages on monitored hosts, correlating them with vulnerability databases such as the National Vulnerability Database (NVD). This allows the detection of outdated or vulnerable software components in the infrastructure.

During our deployment, one of the critical vulnerabilities detected was associated with the `zabbix-agent` package installed on the agent VM. This finding is particularly significant since the Zabbix agent is the core component used to monitor our machines. The detection revealed multiple *Critical*-level vulnerabilities (e.g., CVE-2024-22120, CVE-2024-42327, CVE-2024-42330), highlighting the risks of running unpatched versions of the Zabbix agent.



host-121	zabbix-agent	1:6.0.14+dfsg-1+b1	Zabbix server can perform comm...	Critical	CVE-2024-22120
host-121	zabbix-agent	1:6.0.14+dfsg-1+b1	A non-admin user account on the...	Critical	CVE-2024-42327
host-121	zabbix-agent	1:6.0.14+dfsg-1+b1	The HttpRequest object allows to...	Critical	CVE-2024-42330
host-121	zabbix-agent2	1:6.0.14+dfsg-1+b1	Zabbix server can perform comm...	Critical	CVE-2024-22120
host-121	zabbix-agent2	1:6.0.14+dfsg-1+b1	A non-admin user account on the...	Critical	CVE-2024-42327
host-121	zabbix-agent2	1:6.0.14+dfsg-1+b1	The HttpRequest object allows to...	Critical	CVE-2024-42330

FIGURE 4.11 – Example of critical vulnerabilities detected by Wazuh on the `zabbix-agent` package.

This result is an important observation, since we rely on the Zabbix agent for system monitoring. The detection proves the effectiveness of Wazuh in identifying weaknesses in third-party components. It also enables administrators to take corrective actions such as updating the package, applying vendor patches, or hardening the configuration to reduce the attack surface.

4.2.2 File Integrity Monitoring (FIM)

Wazuh provides File Integrity Monitoring (FIM) capabilities, allowing administrators to detect any unauthorized or suspicious changes to files and directories in real time. In this

project, the directory `/home/test/` was configured for monitoring. To test the functionality, we performed the following actions on a test file :

— Created a file named `test.txt`.

— Modified the file content by adding text.

— Encrypted the file, generating `test.txt.enc`.

— Further modified both files (`test.txt` and `test.txt.enc`).

— Deleted the file `test.txt`.

— Changed the ownership and permissions of `test.txt`.

All these operations were detected in real time by the Wazuh FIM module, and corresponding alerts were generated in the Wazuh dashboard.

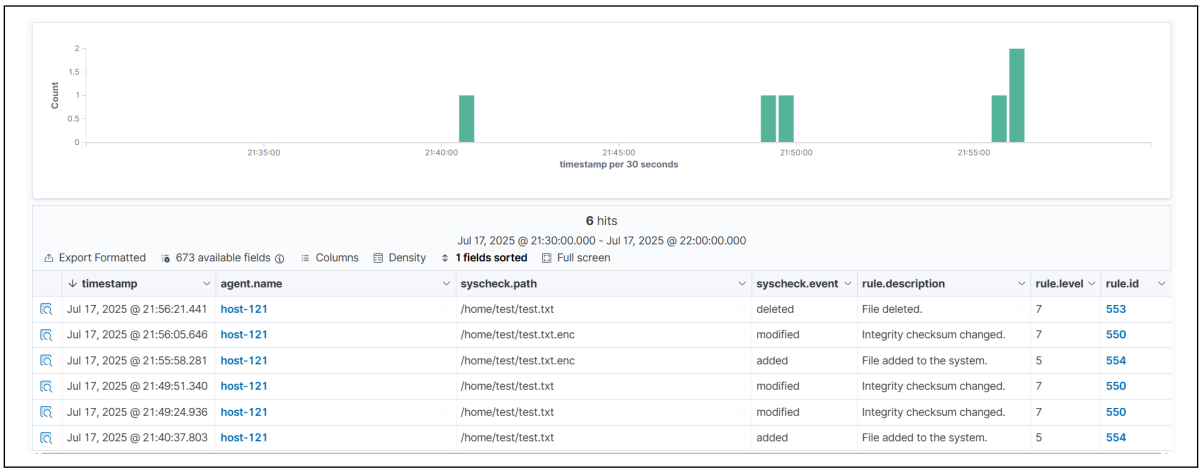


FIGURE 4.12 – Wazuh dashboard displaying FIM alerts generated for the monitored `/home/test/` directory.

To illustrate this in detail, one of the alerts is shown in Figure 4.13. In this case, the file `/home/test/test.txt` was modified, and its permissions were changed from `rw-r-r-` to `rwxrwxrwx`. The alert includes information such as the host, file path, event type (modification), changed attributes (permissions), previous and new values, as well as compliance mappings (e.g., GDPR, HIPAA, PCI-DSS).

t	syscheck.path	/home/test/test.txt
t	syscheck.perm_after	rwxrwxrwx
t	syscheck.perm_before	rw-r--r--
t	syscheck.sha1_after	8bec5385cf0be29808a01f99f7b9735d4a82b5fd
t	syscheck.sha256_after	e8bf1f66db66d25a5326401ac1aa5ee0d0350bf4ffee8837ae6c0d0fb5547896
#	syscheck.size_after	30
t	syscheck.uid_after	0
t	syscheck.uname_after	root
🗓	timestamp	Jul 17, 2025 @ 21:49:51.340

FIGURE 4.13 – Example of a FIM alert : permission change detected on `/home/test/test.txt`.

This demonstrates the effectiveness of Wazuh FIM in detecting unauthorized file changes in real time, ensuring integrity and compliance across the monitored system.

4.2.3 Command and Privilege Auditing

In addition to File Integrity Monitoring, Wazuh was configured to integrate with `auditd` in order to audit all commands executed on the Wazuh Manager server. The objective of this configuration is to detect any suspicious or unauthorized command execution, especially since the Wazuh Manager VM should only be accessed by the system administrator.

All commands executed with root privileges on the server were captured and forwarded to the Wazuh dashboard. This provides visibility into critical activities and allows administrators to quickly identify anomalies or potential compromise.

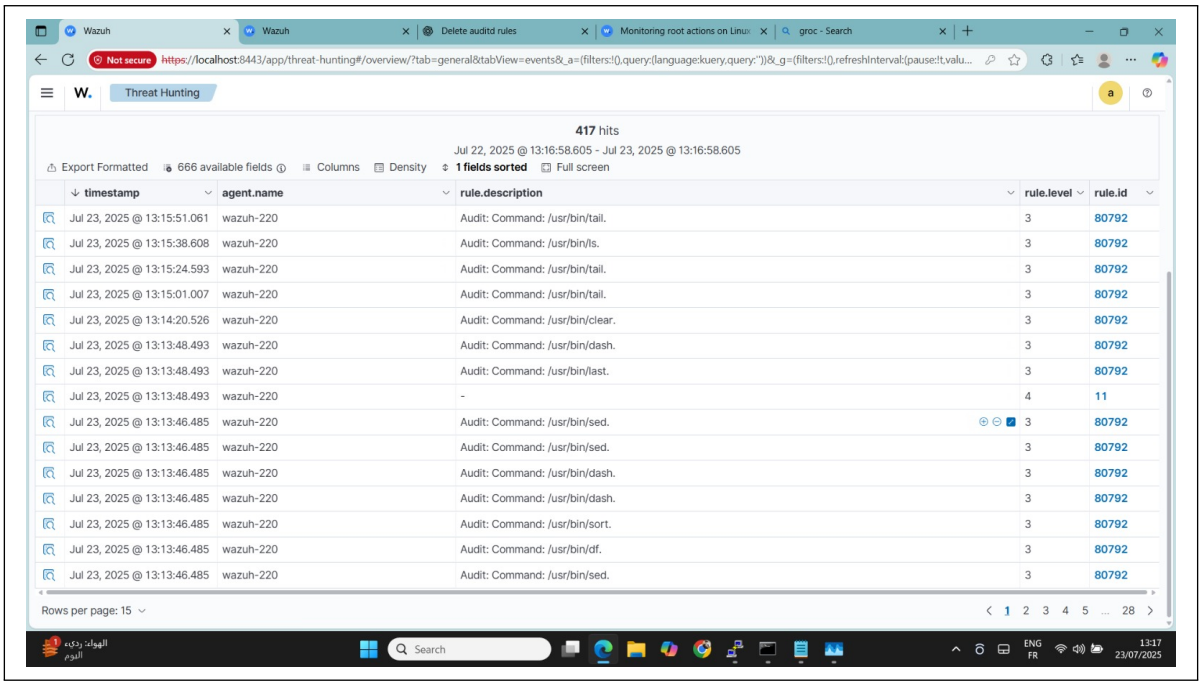


FIGURE 4.14 – Wazuh dashboard displaying audited root commands executed on the manager host.

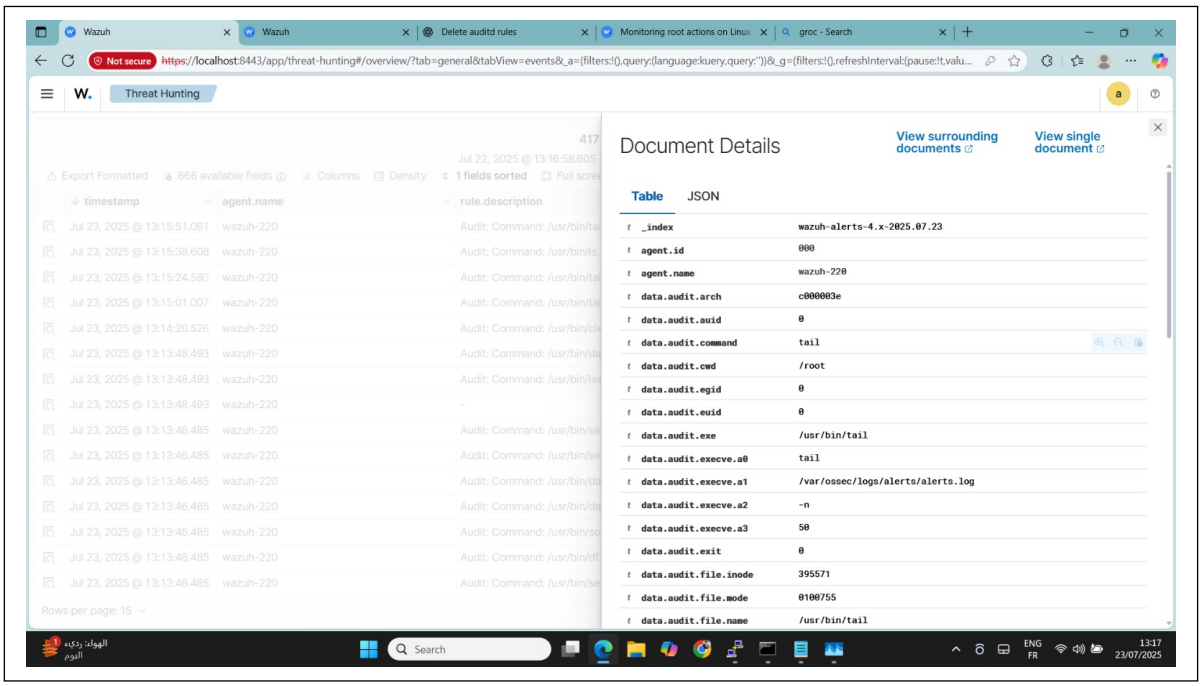


FIGURE 4.15 – Wazuh dashboard displaying audited root commands executed on the manager host.

This setup strengthens the security posture of the environment by ensuring :

— Detection of abnormal or unexpected command executions.

As a result, administrators can detect critical behaviors (e.g., attempts to execute unusual commands) and respond before they escalate into security incidents.

4.2.4 SSH Brute-force & Authentication Anomalies

To validate detection of authentication attacks, a controlled password-guessing test was performed from a host within the same LAN against the monitored agent. The objective was to verify that Wazuh detects repeated failed SSH authentication attempts and raises appropriate alerts for investigation.

— **Test setup (summary) :** A list of usernames and passwords was used to simulate brute-force attempts against the agent. The activity was limited to a lab environment and targeted solely at the test agent to validate detection capabilities.

— **Detection outcome :** Wazuh successfully detected repeated failed login attempts and generated alerts classified as authentication failures and password guessing attempts. Alerts were correlated on the Threat Hunting dashboard, allowing rapid identification of the attacker IP and the targeted internal host.

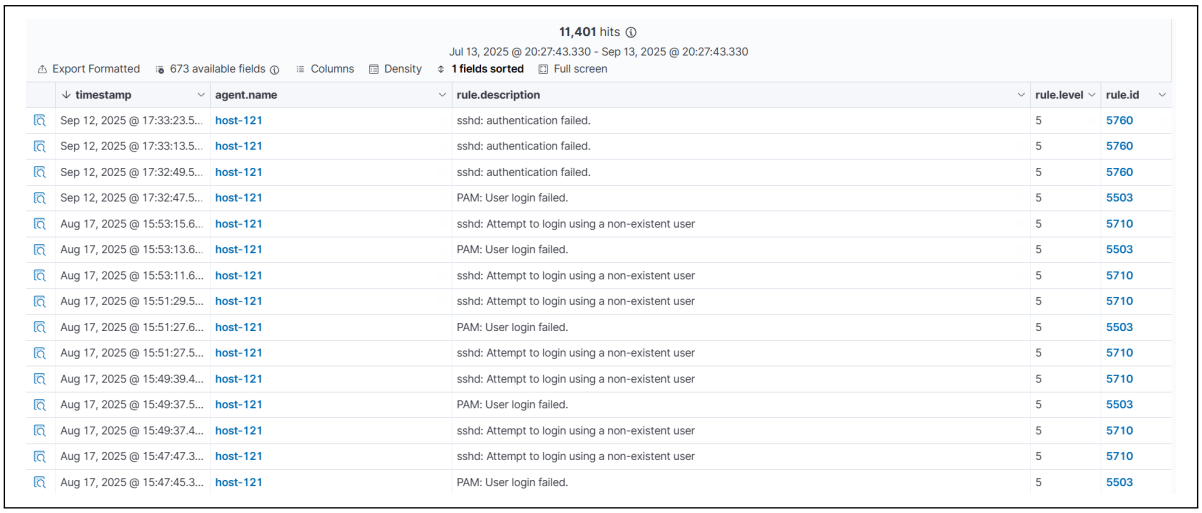
a) Alert Types and Example Details

Wazuh generated two principal alert types during the test :

— **“sshd : Attempt to login using a non-existent user”** — indicates attempts using usernames that do not exist on the target system (useful to detect credential stuffing and reconnaissance).

— **“sshd : Authentication failed”** — indicates valid usernames with incorrect passwords (typical password-guessing behavior).

Figure 4.16 shows the alert types and counts. Figures 4.17 and 4.18 illustrate the details of two different alerts. Each alert provides useful forensic data such as the attacker IP address, the internal target host, timestamp, rule id/level, and mapped MITRE tactic/technique.



11,401 hits

Jul 13, 2025 @ 20:27:43.330 - Sep 13, 2025 @ 20:27:43.330

Export Formatted | 673 available fields | Columns | Density | **1 fields sorted** | Full screen

timestamp	agent.name	rule.description	rule.level	rule.id
Sep 12, 2025 @ 17:33:23.5...	host-121	sshd: authentication failed.	5	5760
Sep 12, 2025 @ 17:33:13.5...	host-121	sshd: authentication failed.	5	5760
Sep 12, 2025 @ 17:32:49.5...	host-121	sshd: authentication failed.	5	5760
Sep 12, 2025 @ 17:32:47.5...	host-121	PAM: User login failed.	5	5503
Aug 17, 2025 @ 15:53:15.6...	host-121	sshd: Attempt to login using a non-existent user	5	5710
Aug 17, 2025 @ 15:53:13.6...	host-121	PAM: User login failed.	5	5503
Aug 17, 2025 @ 15:53:11.6...	host-121	sshd: Attempt to login using a non-existent user	5	5710
Aug 17, 2025 @ 15:51:29.5...	host-121	sshd: Attempt to login using a non-existent user	5	5710
Aug 17, 2025 @ 15:51:27.6...	host-121	PAM: User login failed.	5	5503
Aug 17, 2025 @ 15:51:27.5...	host-121	sshd: Attempt to login using a non-existent user	5	5710
Aug 17, 2025 @ 15:49:39.4...	host-121	sshd: Attempt to login using a non-existent user	5	5710
Aug 17, 2025 @ 15:49:37.5...	host-121	PAM: User login failed.	5	5503
Aug 17, 2025 @ 15:49:37.4...	host-121	sshd: Attempt to login using a non-existent user	5	5710
Aug 17, 2025 @ 15:47:47.3...	host-121	sshd: Attempt to login using a non-existent user	5	5710
Aug 17, 2025 @ 15:47:45.3...	host-121	PAM: User login failed.	5	5503

FIGURE 4.16 – Summary of alert types detected during the brute-force test (non-existent user attempts and authentication failures).

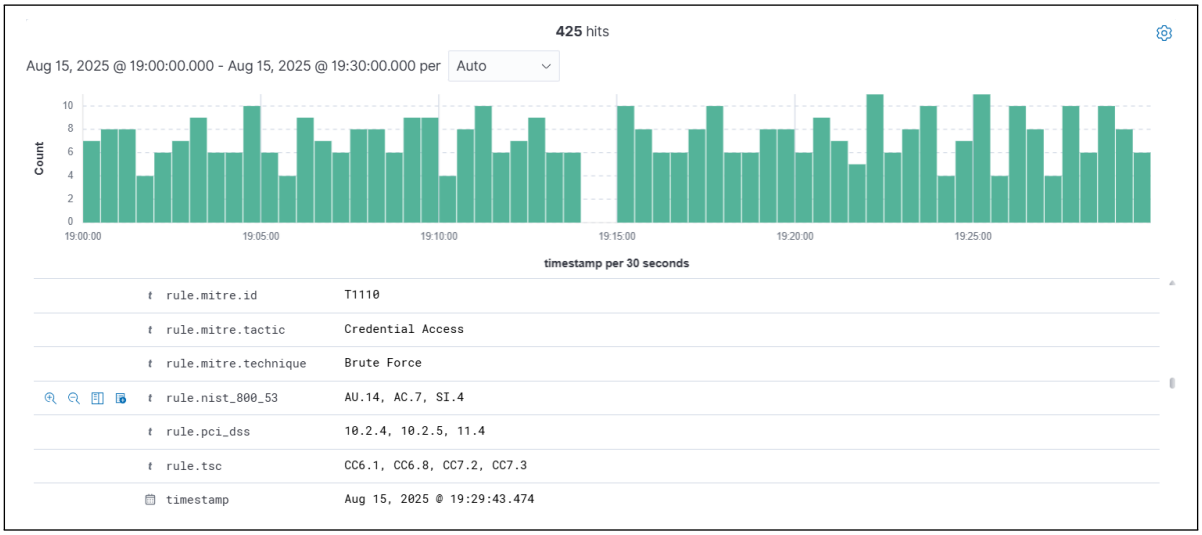


FIGURE 4.17 – Alert detail : Bruteforce alerting

t _index	wazuh-alerts-4.x-2025.09.12
t agent.id	001
t agent.ip	10.10.50.121
t agent.name	host-121
t data.dstuser	root
t data.srcip	196.203.181.122
t data.srcport	53252
t decoder.name	sshd
t decoder.parent	sshd
t full_log	Sep 12 16:32:49 host-121 sshd[924867]: Failed password for root from 196.203.181.122 port 53252 ssh2
t id	1757694769.18130
t input.type	log
t location	journald
t manager.name	wazuh-220
t predecoder.hostname	host-121
t predecoder.program_name	sshd
t predecoder.timestamp	Sep 12 16:32:49
t rule.description	sshd: authentication failed.

FIGURE 4.18 – Alert detail : authentication failed for an existing user with incorrect password. Shows attacker IP, internal host, and mapped MITRE tactic (Credential Access).

b) Notes and operational value

— The alerts provide both immediate operational signals (who is attacking which host and when) and contextual enrichment (rule id, severity, and MITRE mapping) to help prioritize response.

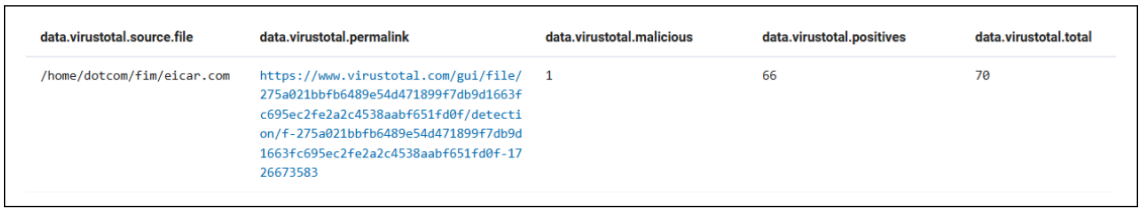
— Because the alert contains the attacker's IP and the targeted internal IP, it is possible to distinguish between internal and external threats and apply appropriate containment (e.g., firewall blocklists, network segmentation).

— Recommended follow-up actions (brief) : confirm whether the source is internal or external, block malicious IPs at the perimeter if external, enforce account lockout policies, and consider automated blocking for repeated failures (e.g., fail2ban or Zabbix actions).

4.2.5 Malware Detection and VirusTotal Integration

Wazuh provides malware detection capabilities by integrating with the VirusTotal service. Suspicious files identified on monitored hosts are automatically checked against VirusTotal's database, which aggregates results from multiple antivirus engines.

For demonstration, the well-known EICAR test malware file was placed on the agent . Immediately, Wazuh detected it as a potential threat and generated an alert on the dashboard, as shown in Figure 4.19. This confirms that the Wazuh agent can successfully identify and report malicious files in real time.



data.virustotal.source.file	data.virustotal.permalink	data.virustotal.malicious	data.virustotal.positives	data.virustotal.total
/home/dotcom/fim/eicar.com	https://www.virustotal.com/gui/file/275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f/detection/f-275a021bbfb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f-1726673583	1	66	70

FIGURE 4.19 – Wazuh dashboard showing an alert generated by the detection of the EICAR malware test file.

The alert was automatically enriched with VirusTotal data. As shown in Figure 4.20, Wazuh collected the file hash and submitted it to VirusTotal. The dashboard displayed the permalink to the VirusTotal report, with results showing that **66 out of 70 antivirus engines flagged the file as malicious**. This seamless integration allows security teams to quickly validate and investigate potential malware.

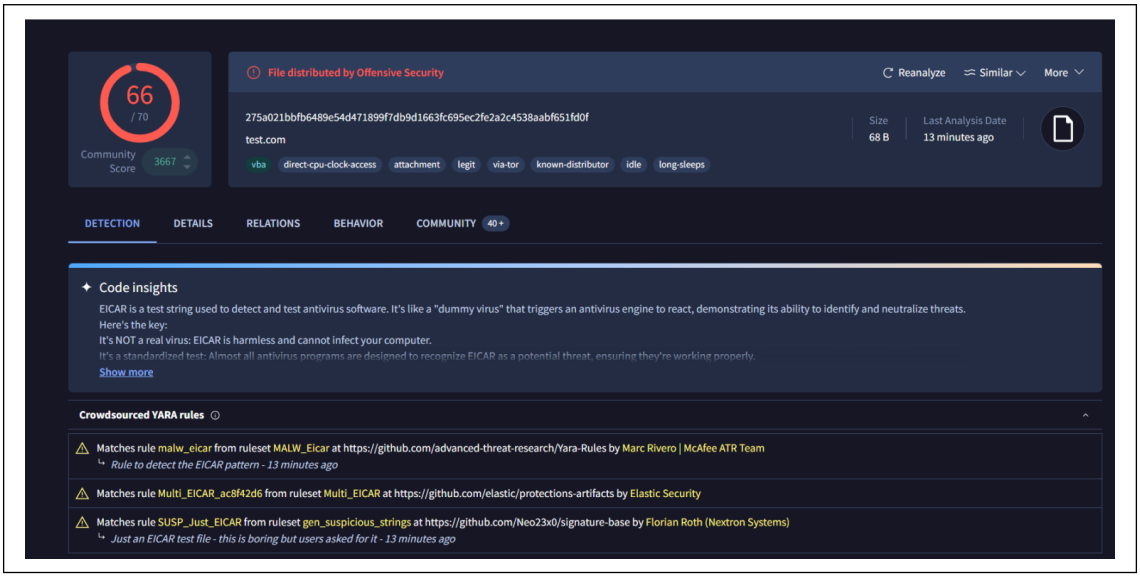


FIGURE 4.20 – Wazuh dashboard enriched with VirusTotal results and permalink for the scanned EICAR test file.

This successful detection demonstrates that both Wazuh and VirusTotal are functioning correctly in identifying and responding to malware threats, thereby enhancing the overall security of the monitored infrastructure.

4.2.6 Automation of Wazuh Agent Installation using Ansible

To streamline the deployment process, the installation of the Wazuh agent was automated using **Ansible playbooks**. This approach eliminates repetitive manual steps, ensures consistency across different systems, and reduces the risk of configuration errors.

Figure 4.21 illustrates the execution of the Ansible playbook. As shown, each task was completed successfully, confirming that the installation process was fully automated and repeatable. Although the Wazuh dashboard screenshot is not available here,the successful execution of the playbook confirmed that the agent was correctly installed and later appeared as active on the Wazuh manager.

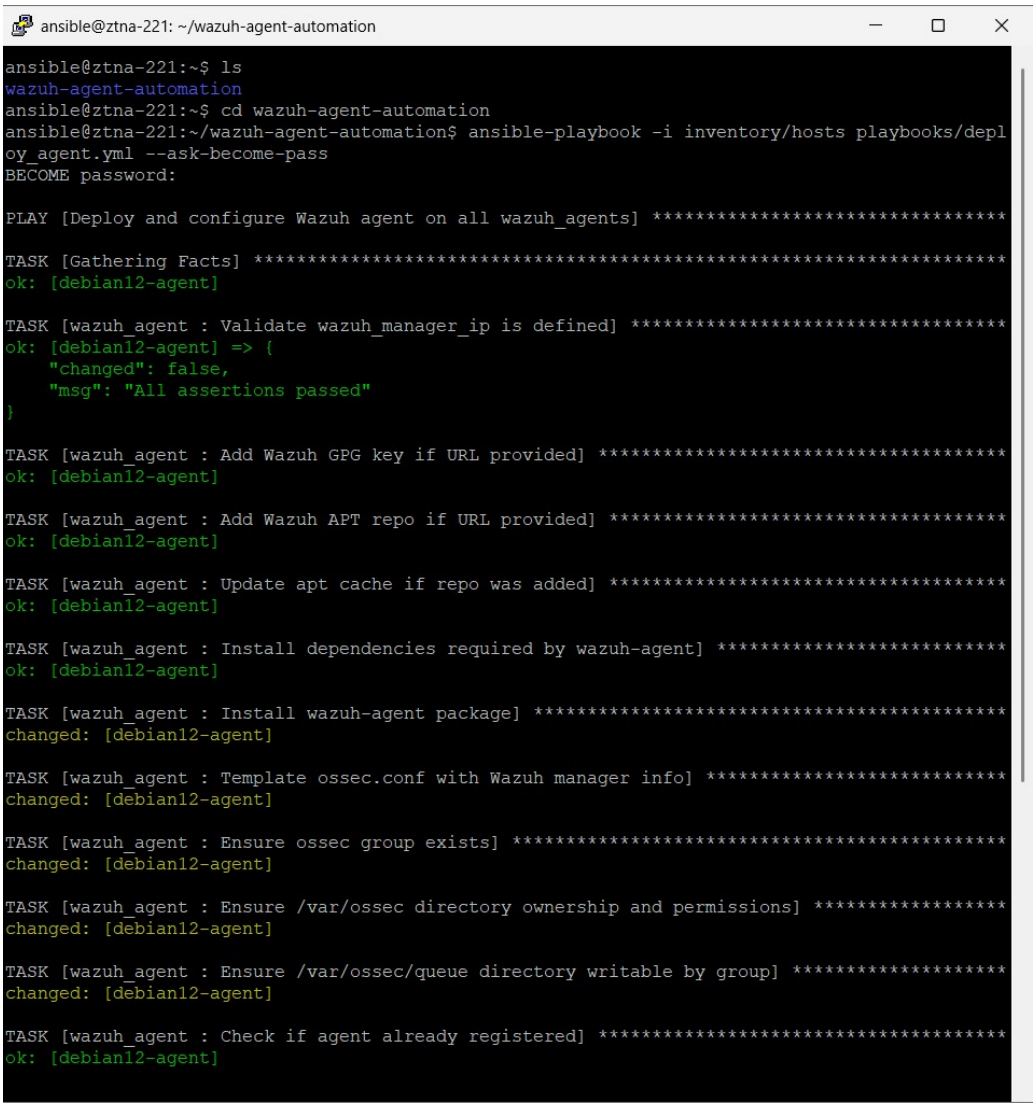


FIGURE 4.21 – Execution of Ansible playbook for automated Wazuh agent installation

Conclusion

The integrated setup proved effective in monitoring system performance and detecting security threats. Zabbix ensured infrastructure visibility, while Wazuh provided strong security detection capabilities. Additionally, automating Wazuh agent installation with Ansible simplified deployment and improved scalability. Overall, the solution is efficient, secure, and easy to manage.

Chapter

5 Challenges and Future Perspectives

Introduction

Throughout this internship, several technical and organizational challenges were encountered, which shaped both the progress and the final outcomes of the project. At the same time, these experiences opened the door for meaningful future improvements and extensions.

5.1 Challenges Faced

During the internship several practical constraints limited the scope and depth of the implementation :

— **Disk space limitations on VMs :** Initial attempts to install the Zabbix server and the Wazuh stack failed or were constrained due to insufficient disk capacity on the provided virtual machines. This required adjustments to the deployment plan and careful management of storage resources.

— **Late access to the final development environment :** Most of the work was performed in a separate test environment. The final clone of the Aurora development environment was provided very late (final week), which prevented full deployment of the SIEM across the entire architecture. As a result, large-scale integration tests and broad attack simulations could not be executed against the real environment.

— **Limited time for extended validation :** The late handover combined with the internship timeline reduced opportunities to perform extended red-team/blue-team exercises, long-term trend analysis, and complex correlation tuning on production-like traffic.

Despite these challenges, the delivered solution was fully functional in the test environment and demonstrated the intended monitoring and detection capabilities.

5.2 Future Perspectives

The next phase of the project would focus on scaling, hardening, automation, and validation. The following items are proposed (ordered roughly by impact and logical progression) :

1. **Full deployment across the architecture :** Install and register Wazuh agents and Zabbix agents on all production and staging hosts of the Aurora environment to obtain complete observability and security coverage.
2. **High-availability and scalability :** Migrate the Wazuh stack and Zabbix server to HA/topology designs (e.g., clustered indexer, replicated manager, load-balanced frontends) to support production workloads and fault tolerance.
3. **SOAR / Orchestration integration :** Integrate a SOAR solution (or lightweight automation tooling) to automate containment and response for specific, well-tested alerts (for example : isolate host, block offending IP, rotate credentials) while keeping human-in-the-loop controls for high-risk actions.
4. **Threat intelligence and enrichment :** Add curated external feeds and internal threat intel to enrich alerts (beyond VirusTotal), improving detection quality and reducing investigation time.
5. **Advanced rule tuning and correlation :** Continue tuning Wazuh rules and Zabbix triggers to reduce false positives and create higher-confidence correlation rules that combine multiple signals (FIM + authentication anomalies + network indicators).

a) Suggested short roadmap (next steps)

— Complete deployment of agents across the Aurora environment and validate connectivity.

— Configure a minimal set of automated responses (SOAR) for one or two high-confidence alerts and test in an isolated environment.

— Run a focused red-team exercise to validate detection coverage and iterate on rule thresholds.

Conclusion

Addressing the above points would convert the current proof-of-concept into a production-grade monitoring and security platform, improving detection capabilities, reducing time to respond, and enabling proactive defense across the full architecture.

General Conclusion

My internship at DOTCOM has been an invaluable and transformative experience, significantly enhancing my professional development.

Technically, I gained extensive expertise in cybersecurity monitoring and threat management. I implemented and configured Wazuh in a single-node environment, deploying agents to monitor critical systems such as Nginx and PostgreSQL. This setup enabled me to detect and respond to threats, including SSH brute-force attacks and authentication anomalies, while integrating vulnerability scanning and malware detection tools, including VirusTotal, to strengthen system security.

Beyond technical skills, this internship strengthened my interpersonal and professional abilities. Working closely with Mr. Haythem Benna and the cybersecurity team improved my communication, teamwork, and project management skills. I learned to navigate complex projects, set clear priorities, and anticipate challenges effectively.

I am deeply grateful to DOTCOM for this enriching opportunity. The knowledge, skills, and practical experience I acquired will undoubtedly guide my future career in cybersecurity, and I am eager to apply these learnings to real-world projects.

Bibliography

[1] **Wazuh**. Wazuh Documentation **[online]**. Available at :
https://documentation.wazuh.com/current/index.html

[2] **Zabbix**. Zabbix Documentation **[online]**. Available at :
https://www.zabbix.com/documentation/current/en/manual

[3] **Ansible**. Ansible Documentation **[online]**. Available at :
https://docs.ansible.com/

[4] **Wazuh**. Configuration Management of Endpoints using Ansible **[online]**. Available at :
https://wazuh.com/blog/configuration-management-endpoints-using-ansible/?highlight=ansible%20automation

[5] **Wazuh**. Monitoring Root Actions on Linux using Auditd and Wazuh **[online]**. Available at :
https://wazuh.com/blog/monitoring-root-actions-on-linux-using-auditd-and-wazuh/?highlight=auditd

[6] **IBM**. Intrusion Detection System (IDS) **[online]**. Available at :
https://www.ibm.com/think/topics/intrusion-detection-system

[7] **Wazuh**. Enhancing IT Security with Anomaly Detection **[online]**. Available at :
https://wazuh.com/blog/enhancing-it-security-with-anomaly-detection/?highlight=anomaly%20detection

[8] **SBCode**. Zabbix PostgreSQL Template **[online]**. Available at :
https://sbcode.net/zabbix/postgresql-template/

[9] **Zabbix**. Zabbix GitHub Repository **[online]**. Available at :
https://github.com/zabbix/zabbix.git