
**Bangladesh University of Engineering and Technology**

**Department of Computer Science and Engineering**


Academic Year 2023–2024

## CSE 406
## Computer Security Sessional

---

# Wazuh: A Comprehensive Look at its XDR and SIEM Capabilities for Enhanced Security

---


**Submitted by:**

1905001 — Mohammad Sadat Hossain

1905004 — Asif Azad

1905005 — Md. Ashrafur Rahman Khan

**Supervisor:** Abdur Rashid Tushar


**Submission Date:** March 9, 2024

# Contents

# Wazuh: A Comprehensive Look at its XDR and SIEM Capabilities for Enhanced Security

## 1 INTRODUCTION TO WAZUH

### 1.1 WHAT IS WAZUH?

Wazuh stands as a free and open-source security platform, wielding the combined power of XDR (extended detection and response) and SIEM (security information and event management). This potent combination safeguards data across diverse environments, from traditional on-premise setups to the modern world of cloud, virtual, and containerized systems.

Wazuh builds upon the capabilities of OSSEC (an open-source intrusion detection system), further enhancing its functionality with additional features, richer APIs, and improved integration capabilities. Trusted by organizations of all sizes, Wazuh offers a reliable defense against ever-present security threats.

### 1.2 WAZUH COMPONENTS

Wazuh primarily comprises of 2 components: the Wazuh Agent and the Wazuh Manager.

#### 1.2.1 WAZUH AGENT

The Wazuh agent, a multi-platform component, runs on user-designated endpoints for monitoring purposes. It transmits data to the Wazuh server in near real-time via an encrypted and authenticated channel. Designed with performance in mind for diverse endpoints, the agent supports popular operating systems (like Windows, Linux, macOS, Solaris etc.) and requires a modest average of 35 MB RAM.

The Wazuh agent empowers users with a range of security-enhancing features, including:

- Log collection
- Command execution
- File integrity monitoring (FIM)
- Security configuration assessment (SCA)
- System inventory
- Malware detection
- Active response

- Container security
- Cloud security

### 1.2.2 WAZUH MANAGER

The Wazuh Manager, also known as the 'Central Component' acts as the core of the Wazuh system. It comprises three key elements:

1. **Wazuh Indexer:** This highly scalable engine serves as a full-text search and analytics platform. It indexes and stores alerts generated by the Wazuh server, enabling efficient retrieval and analysis.

2. **Wazuh Server:** Functioning as the data processing center, the Wazuh server analyzes information received from agents. It employs decoders, rules, and threat intelligence to identify potential security breaches based on known indicators of compromise (IOCs). A single server can handle data from hundreds or thousands of agents, with the capability to scale horizontally in a cluster configuration. Additionally, the Wazuh server manages the agents, allowing for remote configuration and upgrades.

3. **Wazuh Dashboard:** This web-based user interface provides a platform for data visualization and analysis. Pre-configured dashboards offer insights into security events, regulatory compliance (PCI DSS, GDPR, CIS, HIPAA, NIST 800-53, etc.), detected vulnerabilities, file integrity monitoring data, configuration assessment results, cloud infrastructure events, and more. It also facilitates Wazuh configuration management and status monitoring.

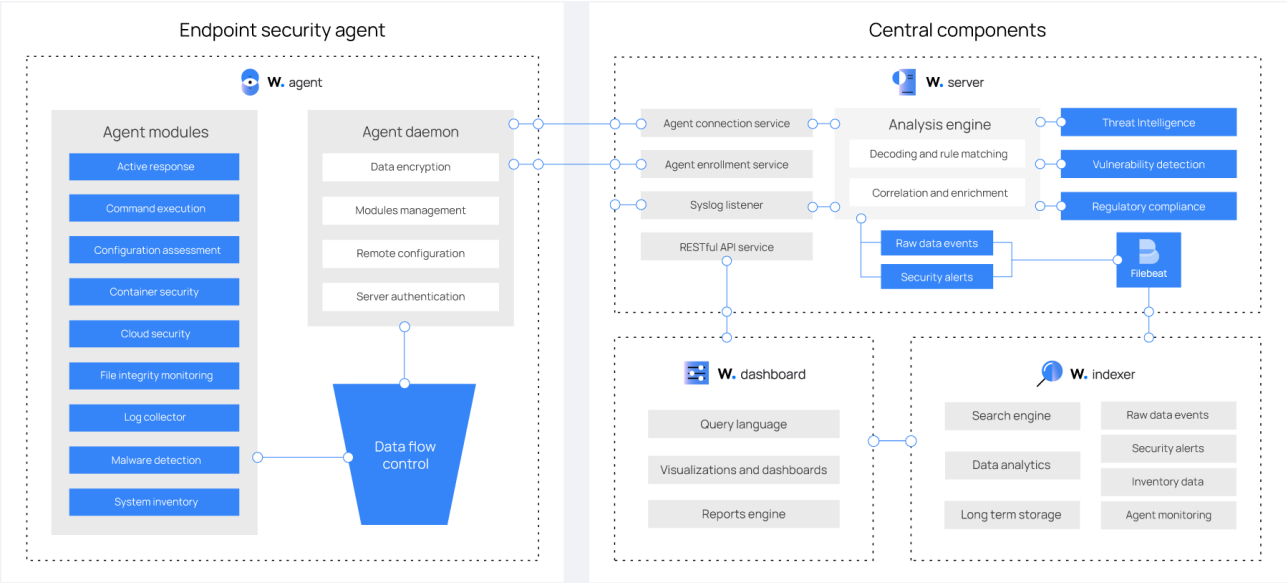


**Figure 1:** Wazuh Components and Data flow

## 1.3 WAZUH ARCHITECTURE

The foundational structure of the Wazuh system hinges on two primary components: agents and servers. Agents, installed on monitored systems, relay security data back to the centralized server. The system also accommodates agentless devices like firewalls and routers, enabling these to transmit log data through various protocols such as Syslog and SSH, or directly via APIs.

Upon receipt, the central server undertakes the decoding and analysis of this data, thereafter dispatching it to the Wazuh indexer. The indexer, potentially a single-node for smaller setups or a multi-node cluster for larger, data-intensive operations, is tasked with data indexing and preservation.

Particularly in production settings, segregating the server and indexer onto separate platforms enhances system integrity. Within this framework, Filebeat plays a critical role, securely shuttling alerts and archives from the Wazuh server to the indexer, all the while safeguarded by TLS encryption.

Illustrated below, the deployment architecture schema delineates the interplay between server and indexer within the ecosystem, underscoring the potential for cluster configurations to achieve scalability and fault tolerance.

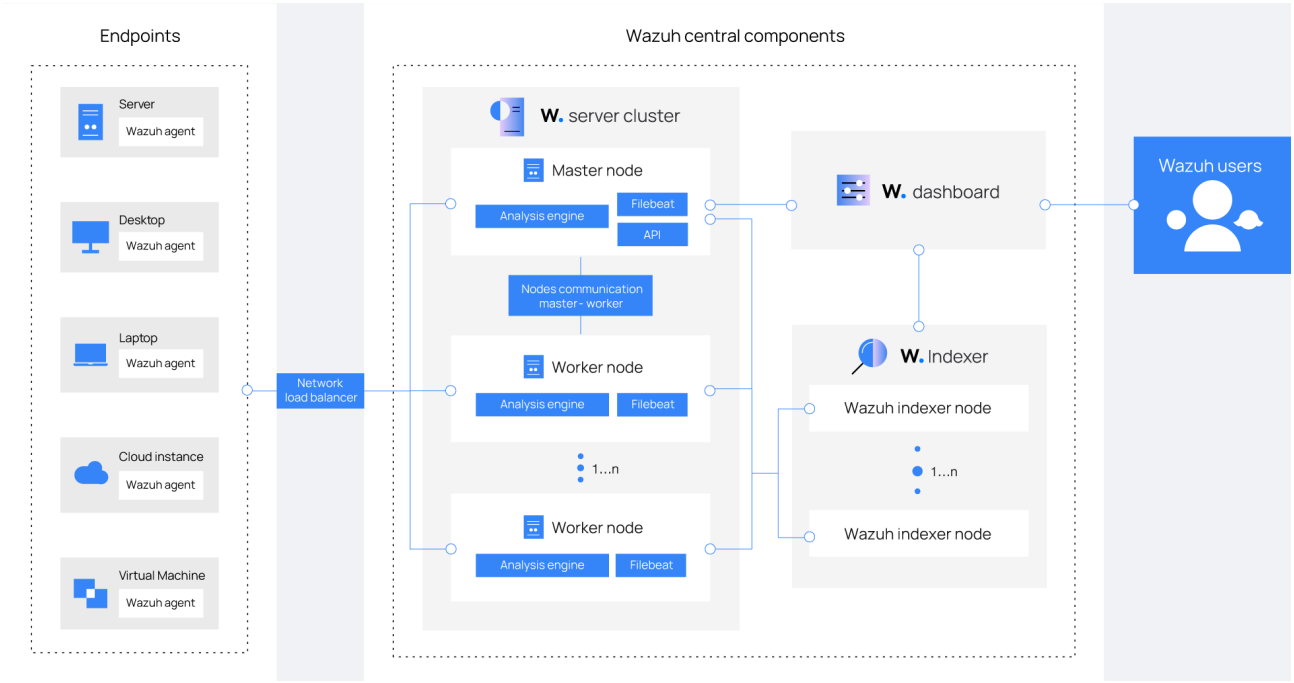


**Figure 2:** Overview of Wazuh Deployment Architecture

# 2 INSTALLATION PREREQUISITES

## 2.1 SYSTEM REQUIREMENTS

### 2.1.1 HARDWARE SPECIFICATIONS

The scale of hardware requisite directly correlates with the quantity of endpoints and cloud services to be secured. This correlation aids in estimating the volume of data analysis and the accumulation of security alerts.

For typical use cases, the consolidation of the Wazuh server, indexer, and dashboard within a single host configuration usually suffices, as this is adequate for supervising no more than 100 endpoints and maintaining ninety days of accessible alert data. The following table delineates the advisable hardware for an initial setup:

| **Endpoints** | **CPU** | **RAM** | **Storage (90 days)** |
|---|---|---|---|
| 1–25 | 4 vCPU | 8 GiB | 50 GB |
| 25–50 | 8 vCPU | 8 GiB | 100 GB |
| 50–100 | 8 vCPU | 8 GiB | 200 GB |

**Table 1:** Recommended Hardware for Quickstart Deployment

In scenarios involving broader infrastructures, a segmented deployment is suggested. The Wazuh server and indexer can be configured into multi-node clusters to enhance scalability and facilitate load distribution.

### 2.1.2 OPERATING SYSTEM COMPATIBILITY

The Wazuh core components necessitate a 64-bit Linux-based installation environment. The subsequent versions of operating systems are endorsed in the official documentation:

- Amazon Linux 2
- CentOS 7, 8
- Red Hat Enterprise Linux 7, 8, 9
- Ubuntu 16.04, 18.04, 20.04, 22.04

### 2.1.3 WEB BROWSER SUPPORT

The Wazuh dashboard is compatible with the following browsers:

- Chrome 95 or newer

- Firefox 93 or newer
- Safari 13.7 or newer

## 2.2 CONFIGURING THE MACHINES

### 2.2.1 WAZUH SERVER

- **Computer Name:** wazuh-server
- **Operating System:** Linux 20.04 (V1 x64)
- **Size:** Standard B2s, 2 VCPUs, 4GB RAM
- **Public IP:** 20.2.220.92
- **Private IP:** 10.0.0.5

### 2.2.2 WAZUH AGENTS

**Agent ID: 001**

- **Computer Name:** wazuh-agent-linux-1
- **Operating System:** Ubuntu 22.04.3 LTS
- **Size:** Standard B2s, 2 VCPUs, 4GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.6

**Agent ID: 002**

- **Computer Name:** wazuh-agent-win
- **Operating System:** Microsoft Windows 11 Pro 10.0.22000.2538
- **Size:** Standard B2s, 2 VCPUs, 4GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.4

**Agent ID: 007**

- **Computer Name:** seed-vm
- **Operating System:** Ubuntu 20.04.6 LTS
- **Size:** Standard B2s, 2 VCPUs, 4GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.4

**Agent ID: 008**

- **Computer Name:** Sadat-Linux
- **Operating System:** Ubuntu 20.04.6 LTS
- **Size:** Standard B2s, 2 VCPUs, 4GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.4

**Agent ID: 009** Understandably, macOS integration could not be done on a virtual machine. We used a physical machine for this purpose.

- **Computer Name:** fahad-air-42
- **Operating System:** macOS 13.5.2
- **Size:** Apple M1, 8-core CPU, 8GB RAM
- **Public IP:** N/A
- **Private IP:** 192.168.0.197

# 3 INSTALLATION

## 3.1 SETTING UP THE WAZUH SERVER

There are two methods to setup the Wazuh Server:

### 3.1.1 QUICKSTART INSTALLATION

We adopted this way to install the Wazuh Server. This is a straightforward all-in-one installation and is suitable for small-scale deployments. The following steps are involved in the installation process:

1. Download and run the Wazuh installation assistant.

```
curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh && sudo bash
↪  ./wazuh-install.sh -a
```

2. Once the assistant finishes, the output will display the access credentials and confirm successful installation.

```
INFO: --- Summary ---
INFO: You can access the web interface https://<wazuh-dashboard-ip>
User: admin
Password: <ADMIN_PASSWORD>
INFO: Installation finished.
```

Make sure to save the credentials for future usage. It will be used to access the dashboard.

3. Access the Wazuh web interface at https://<wazuh-dashboard-ip> using the provided credentials:

```
Username: admin
Password: <ADMIN_PASSWORD>
```

4. Upon first access, a browser warning about the certificate may appear. This is normal because the certificate was not issued by a recognized authority. You may accept the certificate as an exception or configure a certificate from a trusted authority.

5. The passwords for all Wazuh indexer and Wazuh API users can be found in the file named `wazuh-passwords.txt`, which is inside `wazuh-install-files.tar`. To display them, execute:

```
sudo tar -O -xvf wazuh-install-files.tar &&
↪  wazuh-install-files/wazuh-passwords.txt
```

6. To uninstall Wazuh's central components, execute the installation assistant with the option `-u` or `--uninstall`.

### 3.1.2 STEP-BY-STEP INSTALLATION

Please refer to the Wazuh official documentation page for the step-by-step installation of the Wazuh Server components. This provides more in-depth insight and fine-grained control over different details of the installation process.

## 3.2 REGISTERING AGENTS

Registering new agents becomes way too easy once the server is set up. The procedure is stated as follows:

- Navigate to `Agents > Deploy New Agents` as shown in the following image:

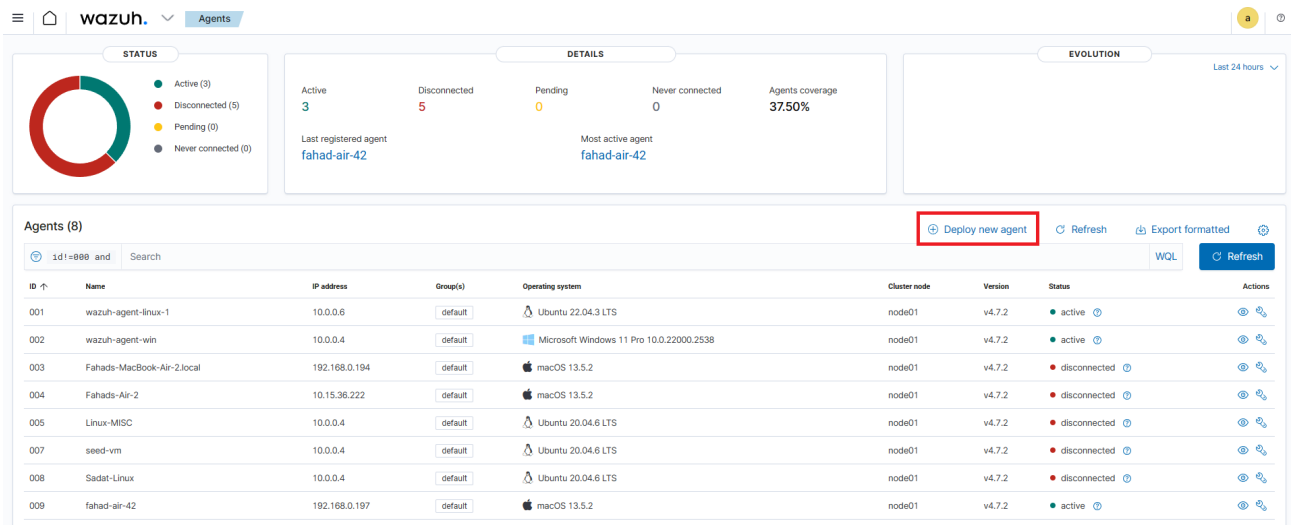


**Figure 3:** Wazuh Dashboard - Deploy New Agent

- There, provide the necessary information like Agent OS, Server address, Agent name and Agent group (last two are optional).

- Finally, two sets of commands will be shown, running which should be enough to install and initiate Wazuh Agent on the given machine.

### 3.2.1 LINUX

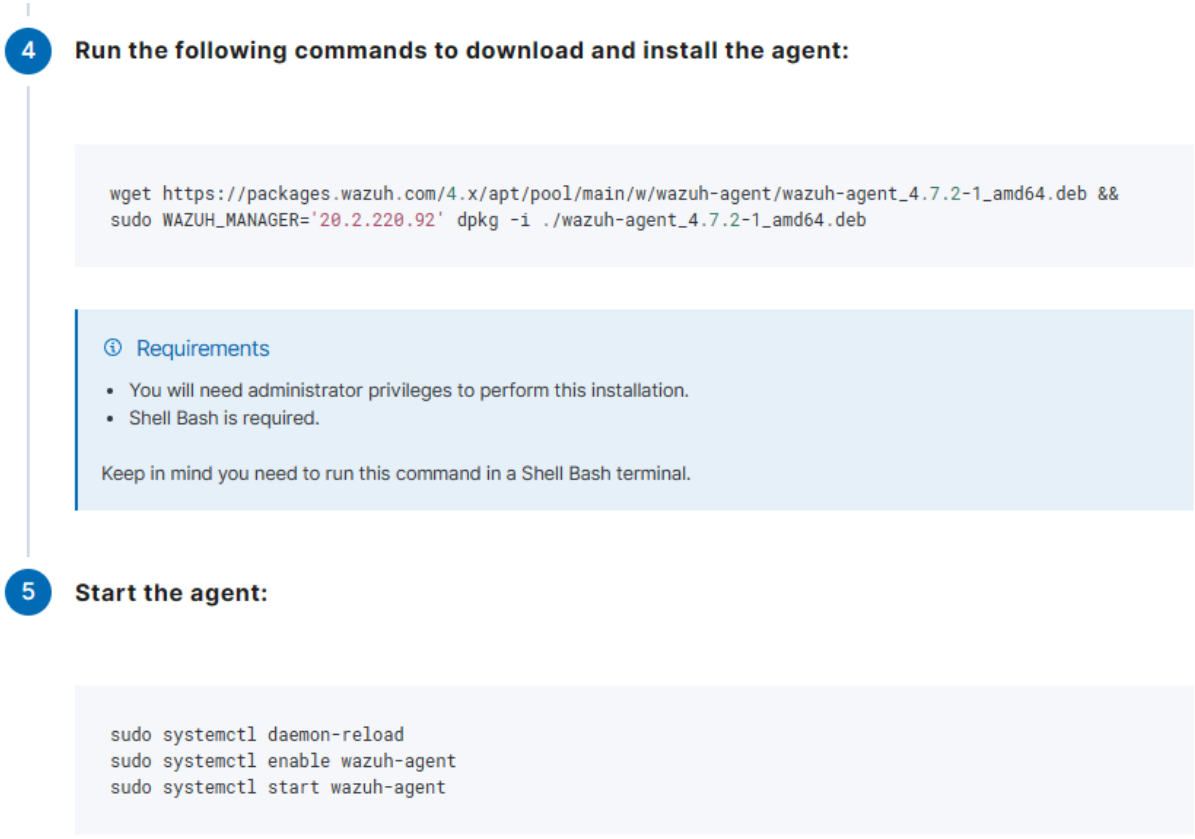


**Figure 4:** Wazuh Agent Installation Commands for a Linux Machine

The commands in the picture go as follows:

```
wget https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-a⌋
↪  gent_4.7.2-1_amd64.deb && sudo WAZUH_MANAGER='20.2.220.92' dpkg -i
↪  ./wazuh-agent_4.7.2-1_amd64.deb
sudo systemctl daemon-reload
sudo systemctl enable wazuh-agent
sudo systemctl start wazuh-agent
```

### 3.2.2 MACOS

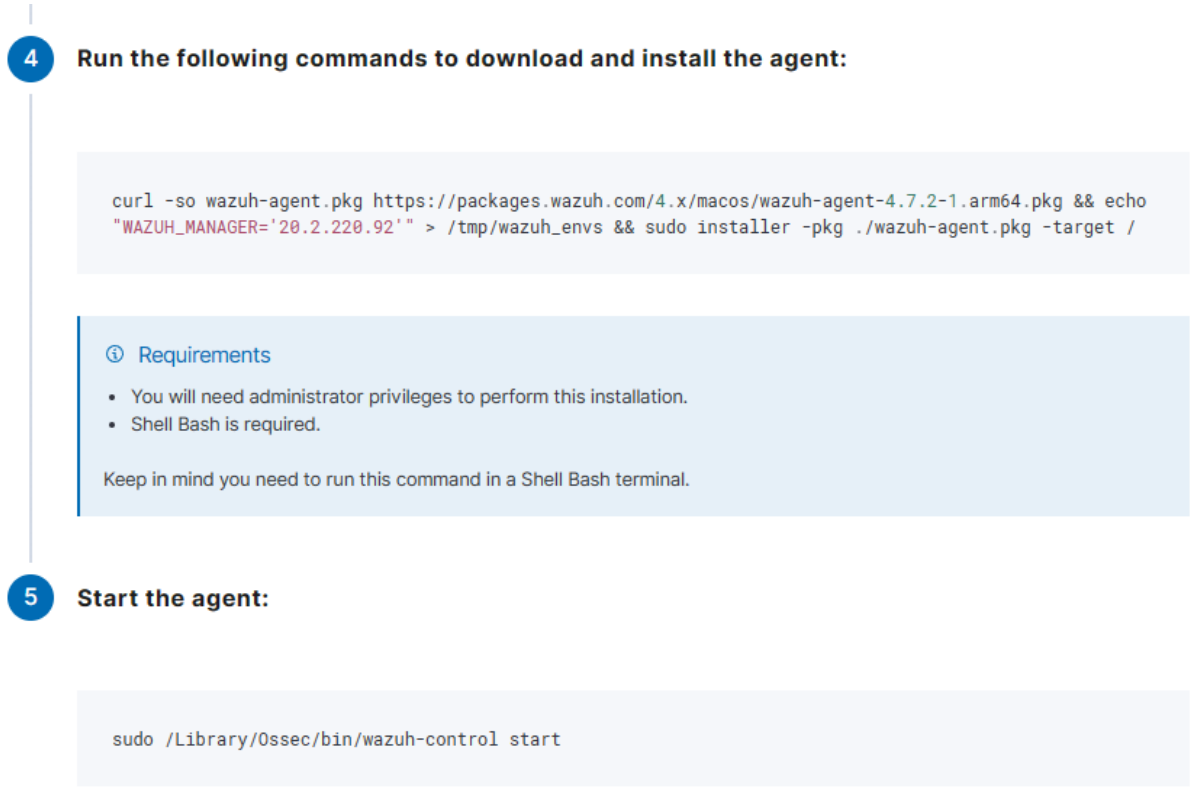


**Figure 5:** Wazuh Agent Installation Commands for a macOS Machine

The commands are:

```
curl -so wazuh-agent.pkg
↪  https://packages.wazuh.com/4.x/macos/wazuh-agent-4.7.2-1.arm64.pkg
↪  && echo "WAZUH_MANAGER='20.2.220.92'" > /tmp/wazuh_envs && sudo
↪  installer -pkg ./wazuh-agent.pkg -target /
sudo /Library/Ossec/bin/wazuh-control start
```

### 3.2.3 WINDOWS

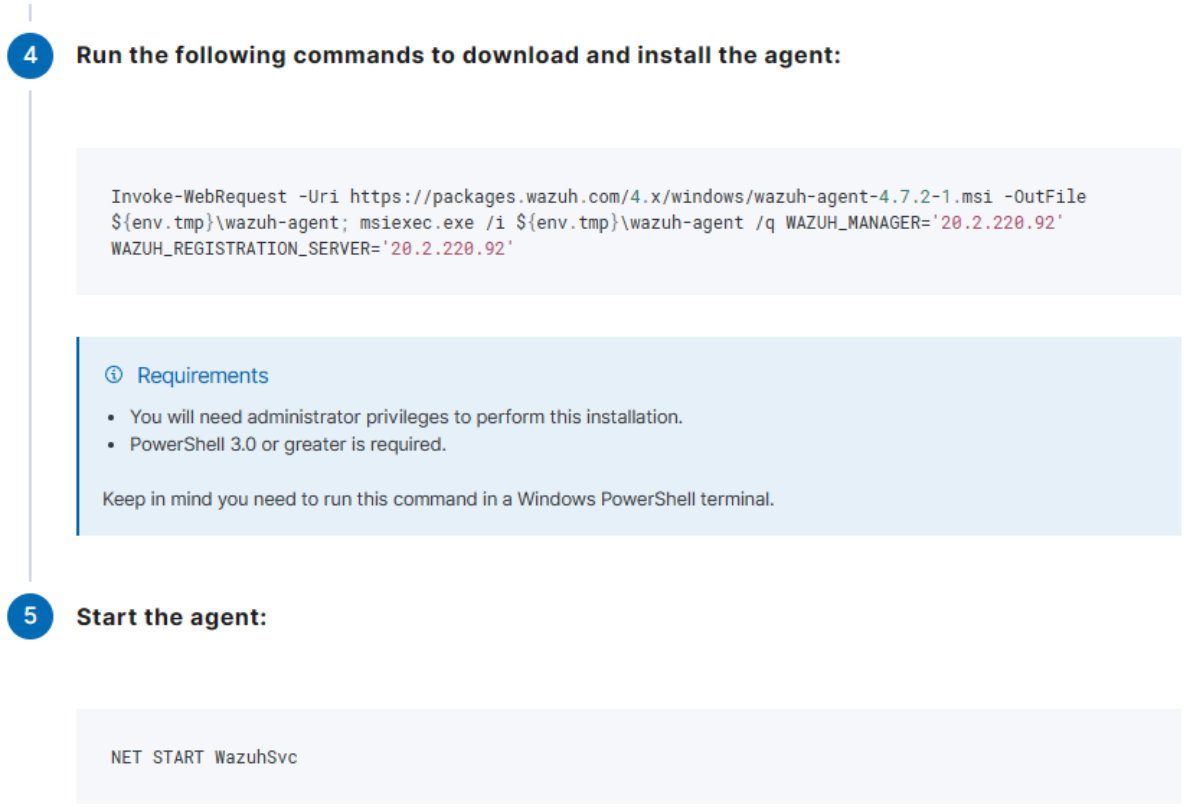


**Figure 6:** Wazuh Agent Installation Commands for a Windows Machine

The commands are compiled here:

```
Invoke-WebRequest -Uri
↪  https://packages.wazuh.com/4.x/windows/wazuh-agent-4.7.2-1.msi
↪  -OutFile ${env.tmp}\wazuh-agent; msiexec.exe /i
↪  ${env.tmp}\wazuh-agent /q WAZUH_MANAGER='20.2.220.92'
↪  WAZUH_REGISTRATION_SERVER='20.2.220.92'
NET START WazuhSvc
```

We installed all three types of agents, as said earlier. There were multiple iterations of setting up the agents. In some instances, the agent had to be reinstalled in the same device with a different name.

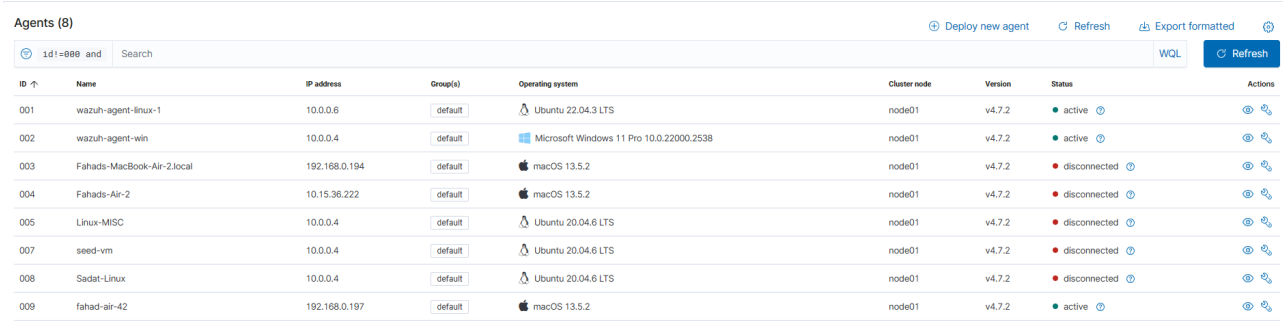


**Figure 7:** Installed Agents

Finally, we ended up working with the agent IDs mentioned in 2.2.2.

# 4 WAZUH FEATURES AND USE-CASES

Wazuh provides several use-cases for monitoring the endpoints and data analysis. These include:

- Configuration assessment
- Malware detection
- File integrity monitoring
- Threat hunting
- Log data analysis
- Vulnerability detection
- Incident response
- Regulatory compliance
- IT hygiene
- Container security
- Posture management
- Cloud workload protection

The following features have been explored in this report.

## 4.1 MALWARE DETECTION

## 4.2 LOG DATA ANALYSIS

Log data collection involves gathering information from various sources like endpoints, applications, and network devices. This data is essential for monitoring system activities and identifying potential security threats. Log data analysis, on the other hand, is the process of examining this collected data to extract useful information and identify patterns or anomalies.

Wazuh collects, analyzes, and stores logs from endpoints, network devices, and applications. The Wazuh agent, running on a monitored endpoint, collects and forwards system and application logs to the Wazuh server for analysis. Additionally, it is possible to send log messages to the Wazuh server via syslog, or third-party API integrations.

### 4.2.1 HOW IT WORKS

Wazuh uses the `Logcollector` module to collect logs from monitored endpoints, applications, and network devices. The Wazuh server then analyzes the collected logs in real-time using decoders and rules. Wazuh extracts relevant information from the logs and maps them to appropriate fields using decoders. The `Analysisd` module in the Wazuh server evaluates the decoded logs against rules and records all alerts in `/var/ossec/logs/alerts/alerts.log` and `/var/ossec/logs/alerts/alerts.json` files.

The Wazuh server also receives `syslog` messages from devices that do not support the installation of Wazuh agents, ensuring seamless integration and coverage across the entire network environment.

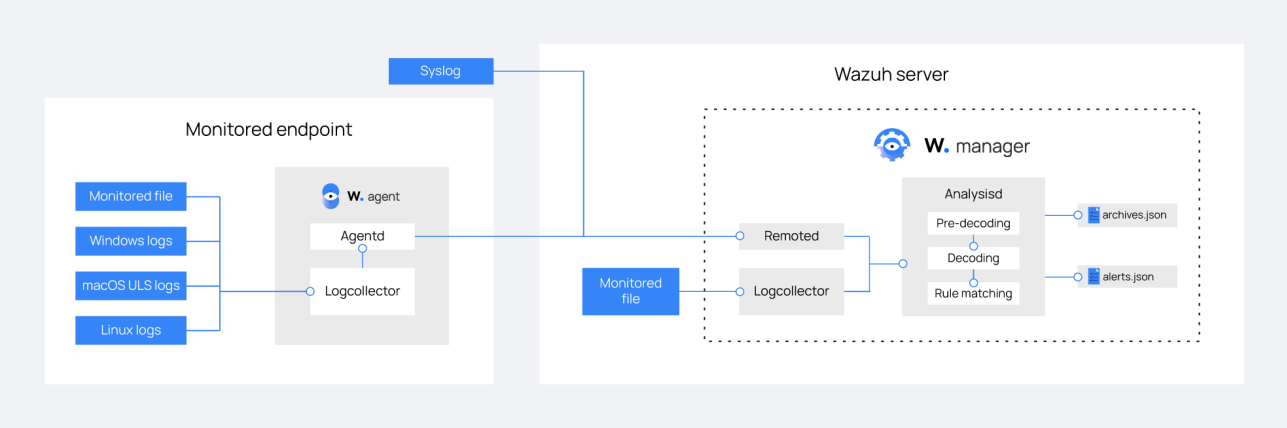


**Figure 8:** The flow of log data collection and analysis in Wazuh

The log data collection process consists of 3 essential phases:

- **Pre-decoding Phase:** This initial stage involves the preliminary processing of collected logs. Here, generic information such as timestamp, hostname, and log source is extracted. The purpose of pre-decoding is to standardize the log format, which enables more detailed analysis.

- **Decoding Phase:** In this critical phase, the pre-decoded log data is converted to a more structured and readable format. The Wazuh decoders parse each log to extract detailed information and map it to specific fields. This process involves processing the log content to identify and categorize elements such as user IDs, source IP addresses, and error codes. The decoding phase transforms raw data into structured information, making precise security monitoring possible from log data analysis.

- **Rule Matching Phase:** Following decoding, the logs are matched against a comprehensive set of predefined rules in the `Analysisd` module. This phase is fundamental to

identifying security incidents or policy violations. Each log is scrutinized, and if certain criteria are met, an alert is generated. This matching process not only identifies potential threats but also categorizes them based on severity, relevance, and type, enabling targeted response mechanisms and efficient threat mitigation.

By default, the Wazuh server retains logs and does not delete them automatically. However, the user can choose when to manually or automatically delete these logs according to their legal and regulatory requirements.

In addition to alert logs, Wazuh stores all collected logs in dedicated archive log files, specifically `archives.log` and `archives.json` in `/var/ossec/logs/archives/`. These archive log files comprehensively capture all logs, including those that do not trigger any alerts. This feature ensures a comprehensive record of all system activities for future reference and analysis.

### 4.2.2 CONFIGURATION

Wazuh supports two primary methods of log data collection.

**Using Wazuh Agent** On devices where Wazuh Agent can be installed, log files can be monitored by simply changing the agent configuration.

**Monitoring Basic Log Files** Configuration for monitoring basic log files involves inserting the `localfile` XML blocks into the `ossec.conf` file of the Wazuh agent. The following is an illustrative example:

```
<localfile>
    <location>/path/to/log/file.log</location>
    <log_format>syslog</log_format>
</localfile>
```

**Monitoring Date-based Log Files** To adapt to dynamic file naming based on dates, the configuration supports strftime format. An example configuration is shown below:

```
<localfile>
    <location>/path/to/log/file-%y-%m-%d.log</location>
    <log_format>syslog</log_format>
</localfile>
```

**Monitoring Using Wildcard Patterns** Wazuh allows for the use of wildcard patterns to monitor multiple log files within a directory. An example of such a configuration is:

```
<localfile>
    <location>/path/to/logs/file*.log</location>
    <log_format>syslog</log_format>
</localfile>
```

**Utilizing Environment Variables in Log Monitoring** Particularly on Windows, Wazuh configurations can incorporate environment variables within log file paths, adding flexibility to the monitoring setup:

```
<localfile>
    <location>%WINDIR%\Logs\CustomLog.log</location>
    <log_format>syslog</log_format>
</localfile>
```

**Using Syslog** The Wazuh server can be configured to listen for incoming syslog messages on predefined ports, enabling support for devices without support for Wazuh Agent. The primary configuration adjustments are made using the `ossec.conf` file located on the server.

**Listening for Syslog Messages** The essential part of the configuration involves defining a `<remote>` block within the `ossec.conf` file of the Wazuh server. An example configuration is as follows:

```
<remote>
    <connection>syslog</connection>
    <port>514</port>
    <protocol>tcp</protocol>
    <allowed-ips>192.168.2.15/24</allowed-ips>
    <local_ip>192.168.2.10</local_ip>
</remote>
```

In this context:

- `<connection>` defines the connection type.

- `<port>` specifies the listening port.
- `<protocol>` indicates the communication protocol.
- `<allowed-ips>` designates permitted sender IP addresses.
- `<local_ip>` is the server's IP address that will listen for log messages.

For changes to take effect, the Wazuh manager requires a restart. This is typically performed via the command:

```
systemctl restart wazuh-manager
```

### 4.2.3 SIMULATION

We demonstrate the following two use-cases of Log Data Analysis.

**Linux Log Data Analysis using `rsyslog`** In this use case, we configure a `Ubuntu 20.04.6` endpoint to forward logs using rsyslog to the Wazuh server for analysis. On the `Ubuntu 20.04.6` endpoint, we create and delete the user account Alice. Wazuh has default rules that generate alerts for the creation and deletion of user accounts.

**Ubuntu endpoint**

1. We edit the `/etc/rsyslog.conf` file and add the following configuration

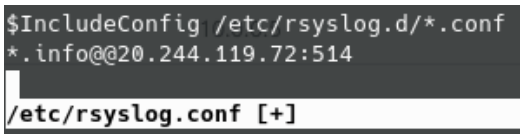


**Figure 9:** `rsyslog` configuration

Here `20.244.119.72` is the IP address of our Wazuh Server.

2. We restart the `rsyslog` service to apply changes.


**Figure 10:** Restart `rsyslog`

**Wazuh server**

1. We edit the `/var/ossec/etc/ossec.conf` file and add the following configuration in between the `<ossec_config>` tags:

```
<remote>
  <connection>syslog</connection>
  <port>514</port>
  <protocol>tcp</protocol>
  <allowed-ips>74.225.241.81</allowed-ips>
</remote>
```

**Figure 11:** Wazuh server configuration

   Here `74.225.241.81` is the IP address of the Ubuntu endpoint.

2. We restart Wazuh Manager for the configuration to take effect.

We test the configuration in the next sub-section.

**Windows Log Data Analysis using Wazuh Agent** In this use case, we configure a `Windows 11` device running Wazuh Agent for log data analysis. On `Windows 11` we install the software Dr. Memory. On the Wazuh Server we create rules for generating alerts when new software is installed.

**Windows endpoint**

1. We edit the Wazuh Agent configuration file at `C:/Program Files (x86)/ossec-agent/ossec.conf` and add the following block inside the `<ossec_config>` tag.

```
<localfile>
  <location>Application</location>
  <log_format>eventchannel</log_format>
</localfile>
```

**Figure 12:** Wazuh Agent configuration

2. We restart Wazuh Agent for the change to apply.

**Wazuh server**

1. We create or modify the following rule at `/var/ossec/ruleset/rules/0585-win-application_rules.xml` to generate alerts when new application is installed.

```
<rule id="60612" level="3">
  <if_sid>60609</if_sid>
  <field name="win.system.eventID">^11707$|^1033$</field>
  <options>no_full_log</options>
  <description>Application installed $(win.eventdata.data).</description>
</rule>
```

**Figure 13:** Wazuh server configuration

2. We restart Wazuh Manager for the configuration to take effect.

### 4.2.4 DASHBOARD UPDATE

**Testing Linux Log Data Analysis using `rsyslog`**

1. We add the new user `Alice`

```
root@seed-vm:~# useradd Alice
```

**Figure 14:** Adding new user

2. We delete the user `Alice`

```
root@seed-vm:~# userdel Alice
```

**Figure 15:** Deleting the new user

3. We navigate to the `Modules > Security events` tab in the Wazuh Dashboards to view the alerts.

| | | | | |
|---|---|---|---|---|
| Mar 9, 2024 @ 07:29:10.500 | Sadat999 | Group (or user) deleted from the system. | 3 | 5903 |
| Mar 9, 2024 @ 07:29:05.369 | Sadat999 | New group added to the system. | 8 | 5901 |
| Mar 9, 2024 @ 07:29:05.369 | Sadat999 | New user added to the system. | 8 | 5902 |

**Figure 16:** Alerts for user/group creation and deletion

4. We expand the alert to see more details.

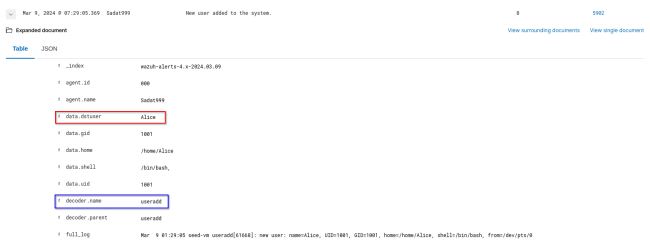


**Figure 17:** The name of the new user (red), the decoder used to process the log (blue)

| | | |
|---|---|---|
| t | rule.description | New user added to the system. |
| # | rule.firedtimes | 1 |
| t | rule.gdpr | IV_35.7.d, IV_32.2 |
| t | rule.gpg13 | 4.13 |
| t | rule.groups | syslog, adduser |
| t | rule.hipaa | 164.312.b, 164.312.a.2.I, 164.312.a.2.II |
| t | rule.id | 5902 |
| # | rule.level | 8 |
| ◕ | rule.mail | false |
| t | rule.mitre.id | T1136 |
| t | rule.mitre.tactic | Persistence |
| t | rule.mitre.technique | Create Account |
| t | rule.nist_800_53 | AU.14, AC.7, AC.2, IA.4 |
| t | rule.pci_dss | 10.2.7, 10.2.5, 8.1.2 |
| t | rule.tsc | CC6.8, CC7.2, CC7.3 |

**Figure 18:** Details of the rule used to generate this alert

### Windows Log Data Analysis using Wazuh Agent

1. We download the software Dr. Memory.
2. We install the application on the `Windows 11` machine.

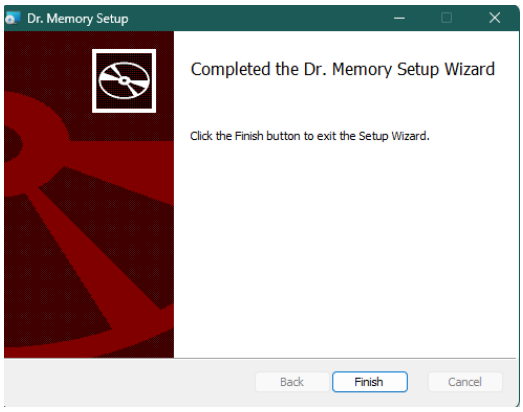


**Figure 19:** Installation of Dr. Memory

3. We navigate to the `Modules > Security events` tab in the Wazuh Dashboards to view the alert.

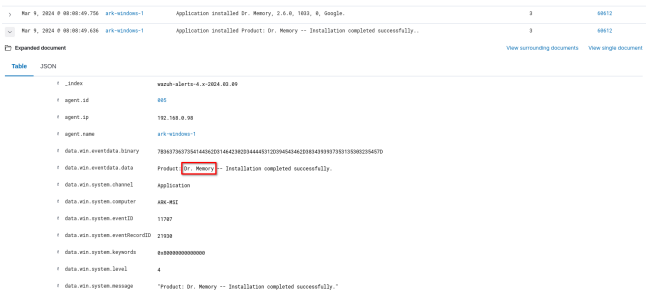


**Figure 20:** Alert for new software installation

# 5 SOURCE CODE

Wazuh source code is publicly available on github. There are 24 public repositories associated with Wazuh, each containing modules for the core back-end, search index, front-end, documentation etc. We currently focused our attention towards the front-end.

## 5.1 REPOSITORIES

Below are the four primary repositories associated with the Wazuh project:

**Wazuh**

**Repository URL:** https://github.com/wazuh/wazuh

**Description:** This repository contains the backend source code for Wazuh Managers and Agents written in C, C++ and Python.

**Wazuh Dashboard**

**Repository URL:** https://github.com/wazuh/wazuh-dashboard

**Description:** Wazuh dashboard is a fork of the OpenSearch Dashboards which incorporate changes to make it easier to use for Wazuh users. It doesn't provide any specific UI, rather it is the platform on which Wazuh web UI runs on.

**Wazuh Dashboard Plugins**

**Repository URL:** https://github.com/wazuh/wazuh-dashboard-plugins

**Description:** This repository contains a set of plugins for Wazuh dashboard. Essentially providing all the UI components used on the Wazuh Web app.

**Wazuh Indexer**

**Repository URL:** https://github.com/wazuh/wazuh-dashboard

**Description:** This repository contains a highly scalable, full-text search and analytics engine. This Wazuh central component indexes and stores alerts generated by the Wazuh server and provides near real-time data search and analytics capabilities.

## 5.2 COMPILING THE FRONT-END FROM SOURCE

From the repository structure and descriptions, it was evident that `wazuh-dashboard-plugins` repository hosted all of the front-end source code.

We followed the contributor's guide and documentation to compile the repository and create a development environment for the front-end. The steps to recreate the environment is outlined below-

1. Remove or disable standalone Docker Engine (if installed). Install Docker Desktop.
2. Configure the docker environment.

```
docker network create devel
docker network create mon
docker plugin install grafana/loki-docker-driver:latest \
    --alias loki --grant-all-permissions
```

3. Assign enough resources to Docker Desktop. At least -
   - 8 GB of RAM
   - 4 CPU Cores
4. Save the path to the `plugins` folder inside `wazuh-dashboard-plugins` repository code as an environment variable, by exporting this path on .bashrc, .zhsrc or similar.

```
./bashrc
export WZ_HOME=~/code/wazuh-dashboard-plugins/plugins
```

5. The Docker volumes will be created by the internal Docker user, making them read-only. Which will prevent us from modifying the source code while running the environment. To prevent this, a new group named `docker-desktop` and GUID 100999 needs to be created, then added to the user and the source code folder:

```
sudo groupadd -g 100999 docker-desktop
sudo useradd -u 100999 -g 100999 -M docker-desktop
sudo chown -R $USER:docker-desktop $WZ_HOME
sudo chmod -R 774 $WZ_HOME
sudo usermod -aG docker-desktop $USER
```

6. Clone the repository.

```
git clone https://github.com/wazuh/wazuh-dashboard-plugins.git
cd wazuh-dashboard-plugins
```

7. Checkout to tag `v4.7.2-2.8.0`, corresponding to `Wazuh v4.7.2` release with `OpenSearch Dashboards 2.8.0`.

```
git checkout v4.7.2-2.8.0
```

8. The `docker` folder inside the repository contains various docker images to create development and testing environments. We use the `osd-dev` environment.

```
cd docker/osd-dev
```

9. Use the `dev.sh` script to call `docker-compose` and spin up the containers required for the development environment.

```
./dev.sh 2.8.0 2.8.0 $WZ_HOME/main up server 4.7.2
```

where,

- `os_version=2.8.0` is the OpenSearch version
- `osd_version=2.8.0` is the OpenSearch Dashboard version
- `os_version=$WZ_HOME/main` is the path to the Wazuh Application source code
- `action=up` is the action to do (one of `up`, `down` or `stop`.
- `server` to create an environment with a Wazuh Server running.
- `server_version` version of the Wazuh server.

10. Also, add a agent container with the command:

```
docker run --name os-dev-280-agent-$(date +%s) \
    --network os-dev-2.8.0 \
    --label com.docker.compose.project=os-dev-280 \
    --env WAZUH_AGENT_VERSION=4.7.2 \
```

```
    -d ubuntu:20.04 bash -c
    'apt update -y
    apt install -y curl lsb-release
    curl -so \wazuh-agent-${WAZUH_AGENT_VERSION}.deb \
        "https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/"\
        "wazuh-agent_${WAZUH_AGENT_VERSION}-1_amd64.deb" \
        && WAZUH_MANAGER='wazuh.manager' WAZUH_AGENT_GROUP='default' \
        dpkg -i ./wazuh-agent-${WAZUH_AGENT_VERSION}.deb
    /etc/init.d/wazuh-agent start
    tail -f /var/ossec/logs/ossec.log'
```

11. Attach a shell to the `os-dev-280-osd-1` docker container to go inside the development environment.

```
docker exec -it os-dev-280-osd-1 /bin/bash
```

12. Install the dependencies using:

```
yarn install
```

13. Run the Web server on `https://0.0.0.0:5601/` using:

```
yarn start --no-base-path
```

The server usually takes a few moments to load all the comments. Once it's loaded login using credentials `admin:admin`.