Bangladesh University of Engineering and Technology

Course: CSE-406 (Computer Security Sessional)
Term Project on Wazuh

Supervisor:
Abdur Rashid Tushar
Lecturer, Dept. of CSE BUET

Submitted by:
Tanjeem Azwad Zaman (1805006)
Ahmed Mahir Sultan Rumi (1805015)

Date of Submission: 14.09.2023

Contents

1 Introduction to wazuh.

1.1 What is Wazuh?

Wazuh is a free, open-source security framework that integrates both XDR (Extended Detection and Response) and SIEM (Security Information and Event Management) functionalities. It offers protection for various computing environments, ranging from traditional on-site setups to cloud-based, virtual, and containerized systems. It builds upon the capabilities of OSSEC (Open Source HIDS SECurity), which is also an open-source Host-based Intrusion Detection System (HIDS), and enhances its functionalities with additional features, richer APIs, and improved integration capabilities

Utilized globally by a diverse range of organizations, from small companies to large corporations, Wazuh serves as a reliable solution for safeguarding valuable data from potential security risks.

1.2 Wazuh Components

Wazuh mainly consists of 2 components, namely the Wazuh Agent and the Wazuh Manager.

1.2.1 Wazuh Agent:

The Wazuh agent operates across various platforms and is installed on the endpoints you wish to monitor. It sends data to the Wazuh server almost instantaneously via a secure and authenticated connection.

Designed with versatility in mind, the agent is intended to monitor a broad range of endpoints while minimizing the impact on their performance. It is compatible with most commonly used operating systems (like windows, linux, macOS, solaris etc.) and typically uses around 35 MB of RAM.

1.2.2 Wazuh Manager:

This is also known as the "central component". It consists of three main parts:

1. Wazuh Indexer:
This is a highly scalable, full-text search and analytics engine. This central component indexes and stores alerts generated by the Wazuh server.

2. Wazuh Server:
THis analyzes data received from the agents. It processes it through decoders and rules, using threat intelligence to look for well-known indicators of compromise (IOCs). A single server can analyze data from hundreds or thousands of agents, and scale horizontally when set up as a cluster. This central component is also used to manage the agents, configuring and upgrading them remotely when necessary.

3. Wazuh Dashboard:
This is the web user interface for data visualization and analysis. It includes out-of-the-box dashboards for security events, regulatory compliance (e.g., PCI DSS, GDPR, CIS, HIPAA, NIST 800-53), detected vulnerable applications, file integrity monitoring data, configuration assessment results, cloud infrastructure monitoring events, and others. It is also used to manage Wazuh configuration and to monitor its status.

1.3 Wazuh Architecture

- Main concept is agents running on monitored endpoints send data to a central server. But agentless devices like firewalls, switches, routers and access points can also communicate and transfer data to servers using Syslog, SSH or through API's. The server analyses and passes results to the Indexer
- The indexer cluster is collection of nodes that together perform read-write operations on indices. A single-node cluster is enough for a small Wazuh deployment that does not process large amounts of data. Many monitored endpoints, or large volumes of data need multi-node clusters for indexing

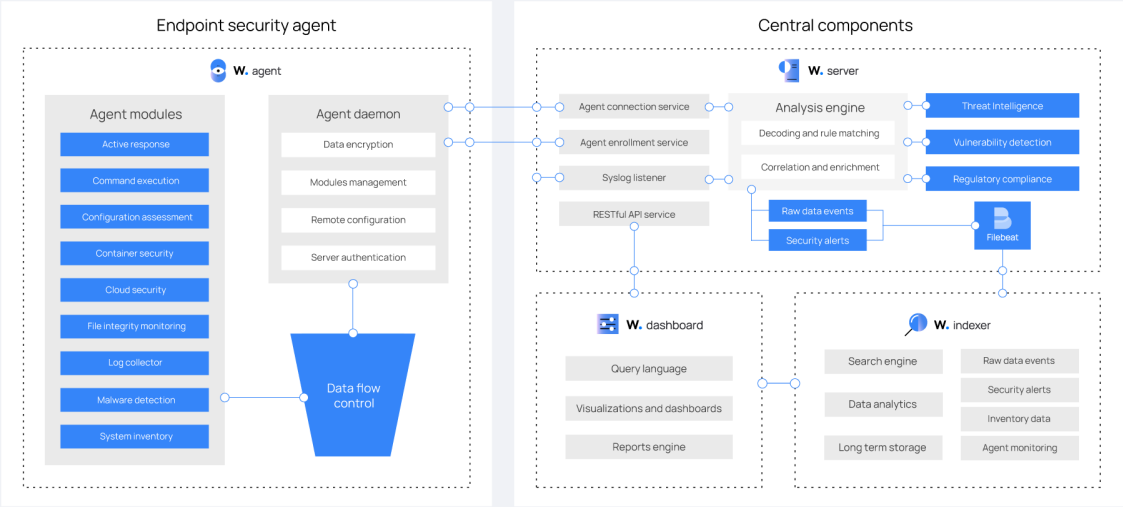


Figure 1: Wazuh Components and Data flow

- In production environment, it is recommended to deploy the server and the indexer on different hosts. In such cases, Filebeat securely forwards alerts and archived events to the Wazuh indexer cluster using TLS encryption.

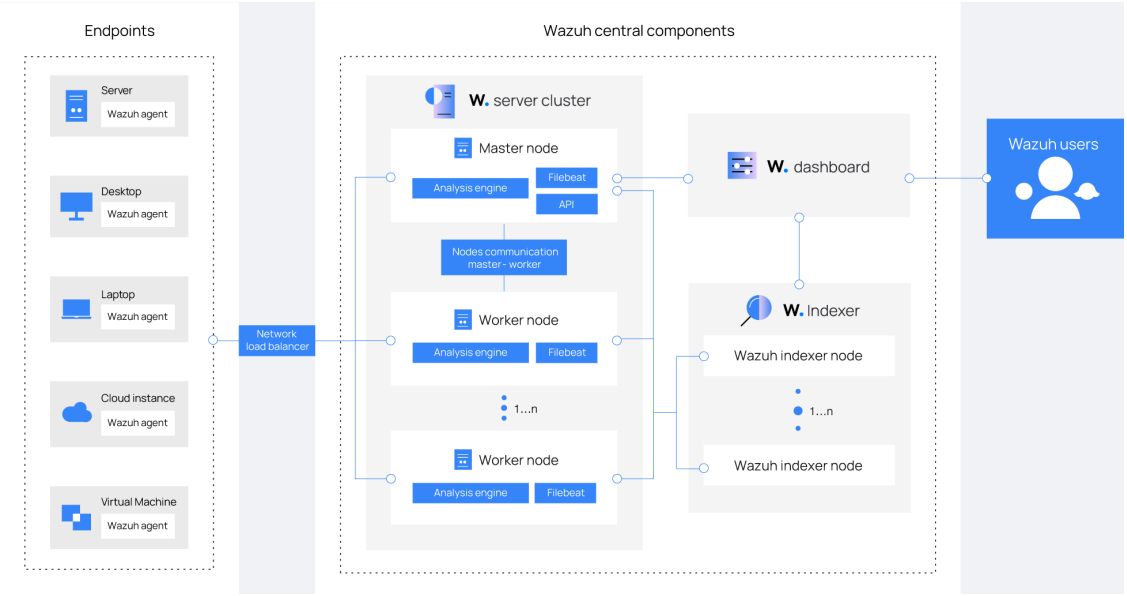


Figure 2: Wazuh Architecture

1.4 Requirements

1.4.1 Hardware

Taken directly from their website for the quickstart guide:
Hardware requirements are highly variable and depend on the number of endpoints and cloud workloads being protected. This estimate is useful for anticipating the amount of data that will be analyzed and the volume of security alerts that will be stored and indexed.

Their quickstart guide assumes that you are deploying the Wazuh server, the Wazuh indexer, and the Wazuh dashboard on a single host (This is what has been done in this project). This configuration should suffice for monitoring up to 100 endpoints and retaining 90 days of queryable/indexed alert data. The table below provides hardware recommendations for a quickstart deployment:

Agents	**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

For larger deployments, we recommend a distributed setup. Multi-node cluster configurations are available for both the Wazuh server and the Wazuh indexer, providing high availability and load balancing.

1.4.2 Operating System

Wazuh's core components are compatible with 64-bit Linux operating systems. We recommend any of the following versions:

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

1.4.3 Browser Compatibility

The Wazuh dashboard is compatible with the following web browsers:

- Chrome 95 or later
- Firefox 93 or later
- Safari 13.7 or later

Other Chromium-based browsers may also be compatible. Internet Explorer 11 is not supported.

2 Configuring the VMs

2.1 Wazuh Server

- **Computer Name:** CSE406-wazuh-server
- **Operating system** Linux 20.04 (V1 x64)
- **Size:** Standard B2s, 2 VCPUs, 4GB RAM
- **Public IP:** 4.213.17.59
- **Private IP:** 10.0.0.5

2.2 Wazuh Agent

- **Computer Name:** CSE406-wazuh-agent-linux
- **Operating system** Linux 20.04 (V2 x64)
- **Size:** Standard B2s, 2 VCPUs, 4GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.4

Issue faced: 6.1

```
Each user had only 2 VMs available per account. So VNET peering was necessary.
Could not integrate a Windows agent into the same vnet as server and linux agent, due to
issues with Azure Active Directory.
```

3 Setting up Wazuh Components

3.1 Setting up Wazuh Server

There are two methods to set up the Wazuh server:

3.1.1 Quickstart:

- Download and Run the installation agent:

  ```
  curl -sO https://packages.wazuh.com/4.5/wazuh-install.sh && sudo bash ./wazuh-install.sh -a
  ```

- Upon successful installation of the server, the following message will be displayed:

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

  Make sure to note down the admin password. It will be used to access the dashboard

- Access Dashboard by going to

  ```
  https://<wazuh-server/dashboard-public-ip>
  ```

And use the following credentials:

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

3.1.2 Step-by-step installation of the Wazuh Indexer, Manager and Dashboard:

Please refer to the Wazuh installation guide at Component-wise Wazuh Manager Installation. This process provides more in-depth insight and control over minor changes in the installation process.

3.2 Setting up Wazuh Agent

Please refer to the Wazuh agent installation guide at Wazuh Agent Installation. This is straightforward and following the steps will ensure a Wazuh agent on the machine

```
*** THE SERVER AND THE AGENT MUST BE ON SEPARATE HOSTS.
ELSE AN ERROR WILL OCCUR STATING THE SAME HOST CANNOT HAVE BOTH AN AGENT AND A MANAGER ***
```

3.2.1 Registering the agent:

- **From the client side:**
  Add the following block in the `var/ossec/etc/ossec.conf` file: Then ping the server. The client will

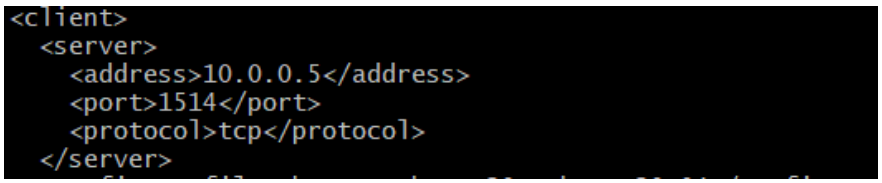


Figure 3: Agent-side modification for agent registration

  automatically request a key from the server and complete the registration process.

- **From the Server Side:** Use the following command: `sudo /var/ossec/bin/manage_agents`
  This will give the following options. Add agent option will help you add the agent. (NB this method did not work for us **??**)

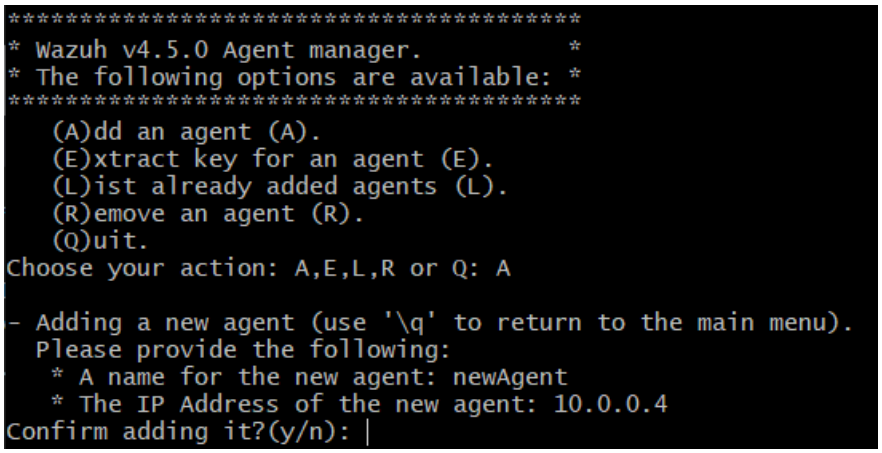


Figure 4: Server-side modification for agent registration

4 Wazuh Features and Use-Cases

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

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

We have explored the following use-cases:

4.1 File Integrity Management

4.1.1 How it works:

The File Integrity Monitoring (FIM) component conducts regular scans on designated paths and actively watches specific folders for any alterations. Configuration settings in both the Wazuh agents and the manager allow you to specify which directories to keep an eye on.

FIM keeps a record of file checksums and various attributes in a local database. When a scan occurs, any differences found in the monitored directories are reported by the Wazuh agent to the Wazuh server. By comparing current and stored file checksums and attributes, the FIM component identifies file changes and triggers an alert if any inconsistencies are detected.

For data collection related to file integrity events, such as file creation, alteration, or removal, Wazuh FIM employs two databases. One is a local SQLite database on the monitored system, stored at:

$$/var/ossec/queue/fim/db$$

on linux.

The other is an agent-specific database on the Wazuh server, managed by the wazuh-db daemon. This daemon creates a unique database for each agent based on their ID and keeps them stored at

$$/var/ossec/queue/db.$$

The FIM module synchronizes the Agent and Server databases. This allows servicing FIM related API queries.Only the changes checksums are updated each time.

4.1.2 Configuration:

- You'll need to define the specific directories or individual files that the File Integrity Monitoring (FIM) component should keep an eye on for any changes such as creation, modification, or deletion. This configuration can be set in both the Wazuh server and Agent configuration files

  $$/var/ossec/etc/ossec.conf$$

  , and it can also be controlled centrally through a unified configuration file

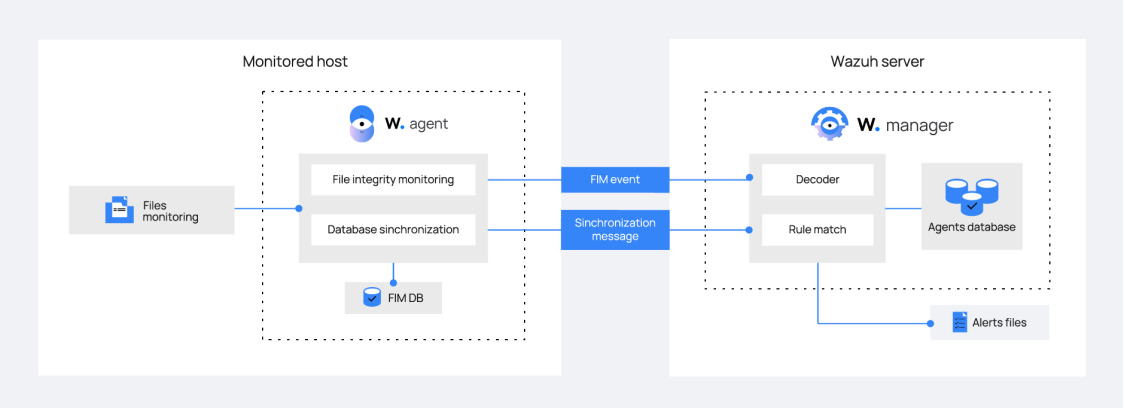


Figure 5: Wazuh FIM flow

- To designate which files and folders should be monitored, use the 'directories' options in the configuration. Multiple entries for files and folders can be added either on the same line separated by commas or on different lines. Shell-style wildcards like * and ? can also be used for directory configuration, similar to how you'd use them in a Command Prompt or shell terminal. For instance, C:\Users \Downloads can be specified.
- When a scan is executed by the FIM module, alerts are generated if there are any altered files or attributes. These alerts can be viewed on the Wazuh dashboard.

Below is a guide on how to configure the FIM module for a particular file and folder. Replace the placeholders with your specific file and folder paths.

- To set up monitoring, insert the following lines into the Wazuh agent's configuration file, substituting the directory placeholders with your actual directory paths:

```
<ossec_config>
    <syscheck>
        <directories>FILEPATH/OF/MONITORED/FILE</directories>
        <directories>FILEPATH/OF/MONITORED/DIRECTORY</directories>
    </syscheck>
<ossec_config>
```

- Then the agent needs to be restarted:

```
systemctl restart wazuh-agent
```

4.1.3 Simulation/ Our implementation:

- Make a new directory at ***/home/seed/Desktop/wazuh/malwaretestdir2*** with mkdir
- Switch to root user with sudo su root

```
seed@CSE406-wazuh-agent-linux:~/Desktop/wazuh/malwaretestdir2$ sudo su root
root@CSE406-wazuh-agent-linux:/home/seed/Desktop/wazuh/malwaretestdir2#
```

Figure 6: Switch to root user

- Open the */var/ossec/etc/ossec.conf* file for editing:

```
root@CSE406-wazuh-agent-linux:/home/seed/Desktop/wazuh/malwaretestdir2# nano /var/ossec/etc/ossec.conf
```

Figure 7: Opening the config file

```
<ossec_config>
        <syscheck>
                <disabled>no</disabled>
                <directories check_all="yes" realtime="yes">/home/seed/Desktop/wazuh/malwaretestdir2</directories>
        </syscheck>
</ossec_config>
```

Figure 8: Including the new file to be monitored in the ossec.conf file

- Enter the directory of a file to be monitored, in our case it was ***/home/seed/Desktop/wazuh/malwaretestdir2***

  Explanation:

  - `<ossec_config>`: The root element of the configuration, encompassing the settings for the Wazuh agent or server.
  - `<syscheck>`: Starts the FIM configuration block. Settings between this tag and the corresponding `</syscheck>` are specific to File Integrity Monitoring (FIM).
    - `<disabled>no</disabled>`: Indicates that the FIM module is enabled and actively running. Setting this to `yes` would disable the FIM module.
    - `<directories check_all="yes" realtime="yes">/home/seed/Desktop/wazuh/malwaretestdir2</direc` Specifies the directory to be monitored. The attributes have the following meanings:
      - `check_all="yes"`: Instructs the FIM module to check all attributes of files in the directory, including permissions, content, and ownership.
      - `realtime="yes"`: Enables real-time monitoring. Wazuh will check for modifications immediately rather than at scheduled intervals.
      - `/home/seed/Desktop/wazuh/malwaretestdir2`: The full path of the directory to monitor.
  - `</syscheck>`: Ends the FIM configuration block.
  - `</ossec_config>`: Closes the root element of the configuration.
- Restart the wazuh agent with `systemctl restart wazuh-agent`
- Create a new file in the directory with touch `/home/seed/Desktop/wazuh/malwaretestdir2/test.txt` and edit with echo:

```
root@CSE406-wazuh-agent-linux:/home/seed/Desktop/wazuh/malwaretestdir2# touch /home/seed/Desktop/wazuh/malwaretestdir2/test.txt
root@CSE406-wazuh-agent-linux:/home/seed/Desktop/wazuh/malwaretestdir2# echo "FIM test" >> /home/seed/Desktop/wazuh/malwaretestdir2/test.txt
```

Figure 9: Creating and changing file in monitored Directory

4.1.4 Dashboard Update:

- Go to dashboard at ***https://4.213.17.59/*** (or the public IP of the server machine)
- Login with name: admin and Password (the ADMIN_PASSWORD from the quickstart installation)
- Go to the dropdown and select "Modules" - "Integrity Monitoring"

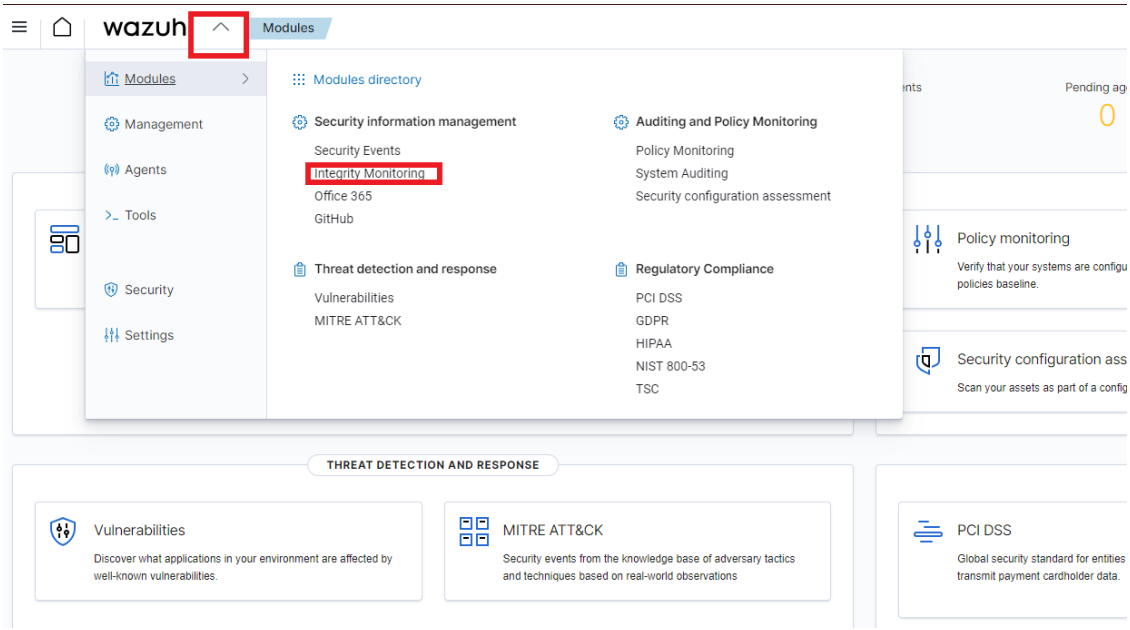


Figure 10: Wazuh Dashboard for FIM demo

- You can see the status of the monitored directories for the registered agents here. In the demo, an old file was deleted and another file was modified seperately. As such, number of creations was 1, deletions was 1 and modifications was 2. The user should get 1 creation and 1 modification.

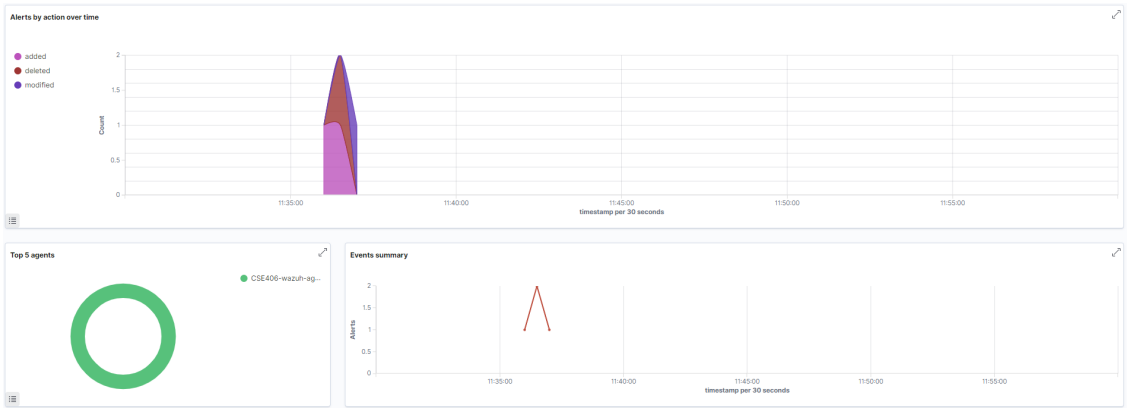


Figure 11: Wazuh Dashboard for FIM demo - 2

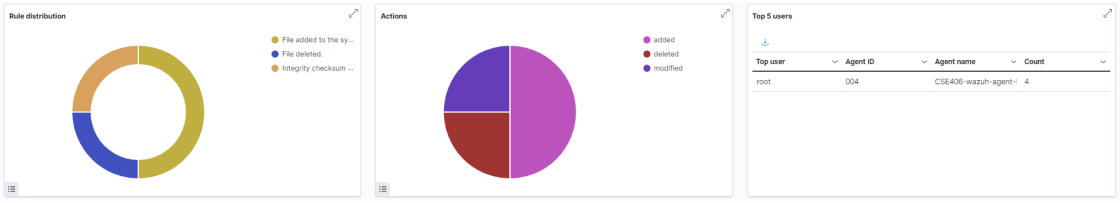


Figure 12: Wazuh Dashboard for FIM demo - 3

4.2 Malware Detection

4.2.1 How it works:

- Malware detection refers to the process of analyzing a computer system or network for the existence of malicious software and files. Detection methods include: checking for signatures of known malware and detecting suspicious behavior from software activity.
- Wazuh does this with the help of the following two modules:
  - In conjunction with File Integrity Management Module (FIM)
    - CTI sources like Virustotal
    - CDB (Centralized Databases) lists with file hashes
    - YARA scans
  - Anomaly Detection with Rootcheck Module:
    - Continuously monitors endpoints
    - Generates alerts when anomaly detected
    - Ensures security even if signature based searching fails

  We've explored Virustotal integration and CDB list malware detection
- **CDB list malware detection:**
  - A CDB (Centralized Database) is populated with signatures (like checksum, hash) of known malware and/or IPs of known malicious websites
  - The FIM monitors directories in an agent. If a file creation/modification takes place, then the FIM calculates the hash of the new object and issues a low-level alert
  - The server's wazuh analysis agent upon receiving the alert, compares the FIM's hash with the hashes stored in the CDB
  - If it finds a match then the analysis agent issues/suppresses a warning based on user configuration. This is usually a high-level warning, since presence of a well-known threat has likely been discovered here.
- **Virus Total Integration:** VirusTotal is a complimentary service with a range of valuable functionalities, particularly useful for our objectives:
  - The platform retains all its scan records, enabling users to look up file hashes. By submitting the hash to VirusTotal's engine, you can determine if that particular file has previously been analyzed and review the associated report.
  - Additionally, VirusTotal offers an API for direct data retrieval without using its web-based interface. Usage of this API is governed by their Terms of Service, which we will touch on in the subsequent section.
  - Similar to the CDB detection, this method looks up hashes on the Virus-Total database, by making an HTTP POST request through the VirusTotal API. And generates one of the following alerts:
    - Error: Check credentials.
    - Error: Public API request rate limit reached.
    - Alert: No records in VirusTotal database.
    - Alert: No positives found.
    - Alert: X engines detected this file. X is the number of antivirus engines.

    Wazuh logs the triggered alert in the `/var/ossec/logs/integration.log` file and stores it in the `/var/ossec/logs/alerts.log` file with all other alerts.

4.2.2 Configuration

- **CDB matching Malware Detection:**

  For the Server:

  – Create CDB list "malware-hashes" for known malware hashes and save to ***"/var/ossec/etc/lists"***

  ```
  nano /var/ossec/etc/lists/malware-hashes
  ```

  – Add known malware hashes to file as key:value pairs (we're using the known MD5 hashes of Mirai and Xbash)

  ```
  E0ec2cd43f71c80d42cd7b0f17802c73:mirai
  55142f1d393c5ba7405239f232a6c059:Xbash
  ```

```
root@CSE406-wazuh-server:/home/wazuh-server# cat /var/ossec/etc/lists/malware-hashes
e0ec2cd43f71c80d42cd7b0f17802c73:mirai
55142f1d393c5ba7405239f232a6c059:Xbash
```

Figure 13: Malware-list creation

  – Add reference to CDB list in wazuh manager config file `/var/ossec/etc/ossec.conf` By specifying in the ⟨*ruleset*⟩ block:

```
<ruleset>
  <!-- Default ruleset -->
  <decoder_dir>ruleset/decoders</decoder_dir>
  <rule_dir>ruleset/rules</rule_dir>
  <rule_exclude>0215-policy_rules.xml</rule_exclude>
  <list>etc/lists/audit-keys</list>
  <list>etc/lists/amazon/aws-eventnames</list>
  <list>etc/lists/security-eventchannel</list>
  <list>etc/lists/malware-hashes</list>

  <!-- User-defined ruleset -->
  <decoder_dir>etc/decoders</decoder_dir>
  <rule_dir>etc/rules</rule_dir>
</ruleset>
```

Figure 14: Add list to ruleset

  – Create a custom rule in the `/var/ossec/etc/rules/local_rules.xml` file on the Wazuh server This generates alert when a modified/new file matches a hash in the CDB list. (rules 554 and 550 must match first for creation and modification)

```
<group name="malware,">
  <rule id="110002" level="13">
    <!-- The if_sid tag references the built-in FIM rules -->
    <if_sid>554, 550</if_sid>
    <list field="md5" lookup="match_key">etc/lists/malware-hashes</list>
    <description>File with known malware hash detected: $(file)</description>
    <mitre>
      <id>T1204.002</id>
    </mitre>
  </rule>
</group>
```

Figure 15: Change local rules

  – Restart the Server with `systemctl restart wazuh-manager`

- On the Client:
  - Just add a folder (s) you wish to monitor.

```
<ossec_config>
        <syscheck>
                <disabled>no</disabled>
                <directories check_all="yes" realtime="yes">
/home/seed/Desktop/wazuh/malwaretestdir2</directories>
        </syscheck>
</ossec_config>
```

Figure 16: Add folder to monitor

- Virustotal Integration:
  - Get a public key from their website
  - Add the key to an */langleintegration/rangle* block in ossec.conf in the server.

```
<ossec_config>
   <integration>
     <name>virustotal</name>
     <api_key>[redacted]
api_key> <!-- Replace with your VirusTotal API key -->
     <group>syscheck</group>
     <alert_format>json</alert_format>
   </integration>
</ossec_config>
```

Figure 17: Add folder to monitor

4.2.3 Simulation

We can simulate both detection methods by one experiment. Downloading a virus into our monitored directory on the agent:

```
sudo curl https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai \
        --output /home/seed/Desktop/wazuh/malwaretestdir2/mirai
```

4.2.4 Dashboard Update

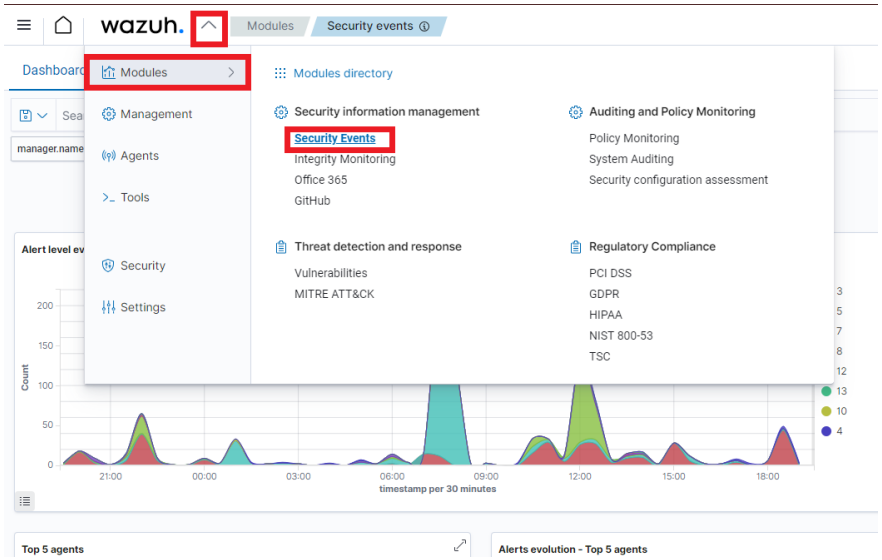


Figure 18: Navigate to Security events

- After the malware has been downloaded, the number of level 12+ threats should go up by 2

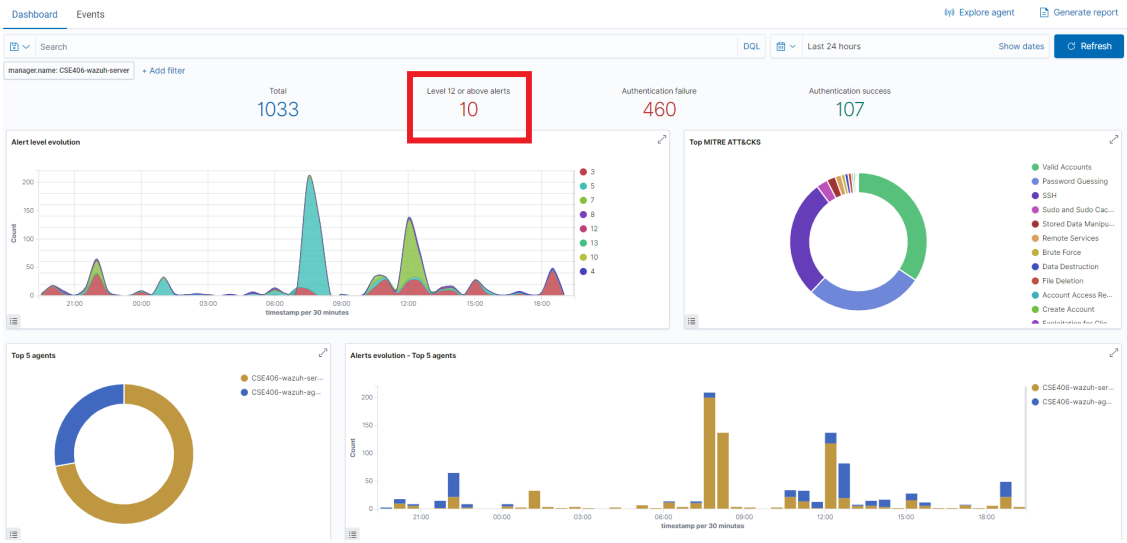


Figure 19: Before malware download

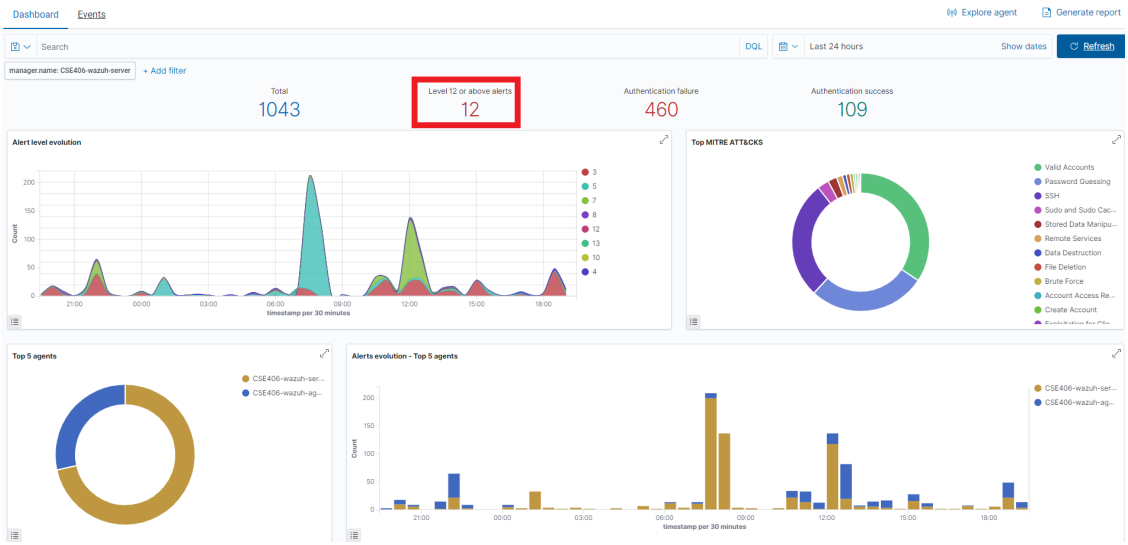


Figure 20: After malware download

- 1 for the malware detection by CDB comparison
- The other for the detection by Virustotal

Figure 21: 2 new "events" reported

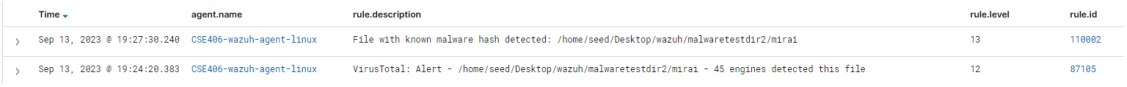


Figure 22: More details by clicking "Events" tab beside "Dashboard"

4.3 Active Response

4.3.1 How it works

Security teams often encounter problems in incident response such as addressing high severity events in a timely manner or providing complete mitigation actions. They might struggle to collect relevant information in real time, which makes it challenging to understand the full scope of an incident. Wazuh SIEM and XDR platform improves this by:

- Providing real-time visibility into security events
- Automating response actions to threats
- Providing out-of-the-box response scripts
- Reducing alert fatigue

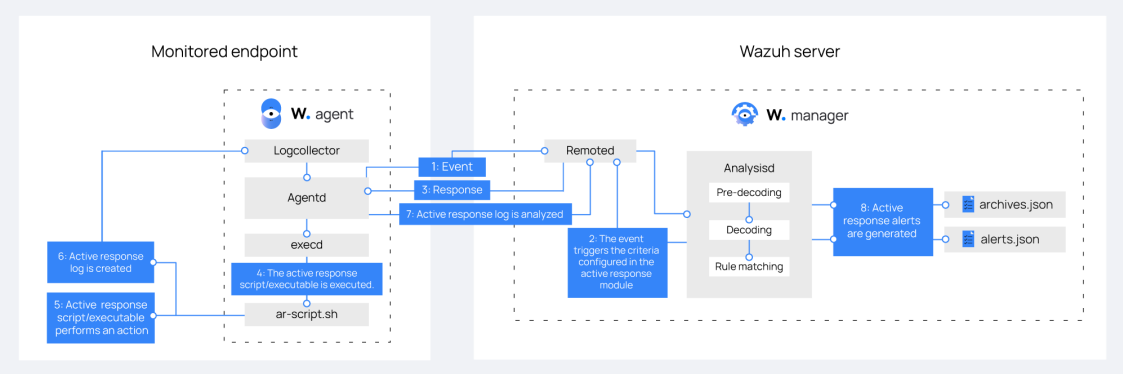


Figure 23: Mechanism of Active Response

Wazuh has an active response module that helps security teams automate response actions based on specific triggers, enabling them to manage security incidents effectively.

Automating response actions ensures that high-priority incidents are addressed and remediated promptly and consistently. This is especially valuable in environments where security teams are resource-constrained and need to prioritize their response efforts. The Wazuh active response module executes these scripts on monitored endpoints when an alert of a specific rule ID, level, or rule group triggers. We can set any number of scripts to initiate in response to a trigger.

4.3.2 Configuration

1. Configuring the Wazuh server

   - Add a new `<command>` block in the Wazuh server `/var/ossec/etc/ossec.conf` configuration file as shown below.

     ```
     <command>
       <name>host-deny</name>
       <executable>host-deny</executable>
       <timeout_allowed>yes</timeout_allowed>
     </command>
     ```

     Explanation:

     - `executable`: Specifies the active response script or executable that must run upon a trigger
     - `timeout_allowed`: Allows a timeout after a period of time

   - Add an `<active-response>` block within the `<ossec_config>` tag in the same Wazuh server `/var/ossec/etc/ossec.conf` file. The `<active-response>` block defines when and where a command executes.

     ```
     <ossec_config>
       <active-response>
         <disabled>no</disabled>
         <command>host-deny</command>
         <location>defined-agent</location>
         <agent_id>001</agent_id>
         <level>10</level>
         <timeout>180</timeout>
       </active-response>
     </ossec_config>
     ```

Explanation:

- `location`: Specifies where the command must execute. The options are: local, server and defined agent (agent id can be specified)
- `command`:Specifies the command to configure. This is the command name defined in the previous step.

- Restart the Wazuh manager to apply all the changes made

```
sudo systemctl restart wazuh-manager
```

2. Configuring the Wazuh agent

   - No configuration is required using out-of-the-box active response scripts
   - For using custom active response scripts, add your custom active response script or executable to the `/var/ossec/active-response/bin` directory on Linux/Unix endpoints

4.3.3 Simulation

Our goal is to use the restart-wazuh active response script to restart the Wazuh agent on a monitored endpoint. We configure it to restart the Wazuh agent whenever the `/var/ossec/etc/ossec.conf` configuration file changes. The steps are shown below:

- Open the Wazuh server `/var/ossec/etc/ossec.conf` file. There should already be a command block with the restart-wazuh script.

```
<command>
  <name>restart-wazuh</name>
  <executable>restart-wazuh</executable>
</command>
```

Figure 24: restart-wazuh script

- Add the ¡active-response¿ block below to the Wazuh server `/var/ossec/etc/ossec.conf` configuration file. The block must be within the `<ossec_config>` tag.

```
<active-response>
  <command>restart-wazuh</command>
  <location>local</location>
  <rules_id>100009</rules_id>
</active-response>
```

Figure 25: active response block to activate the script

- Add a new rule to the `/var/ossec/etc/rules/local_rules.xml` configuration file.

```
<group name="restart,">
  <rule id="100009" level="5">
    <if_sid>550</if_sid>
    <match>ossec.conf</match>
    <description>Changes made to the agent configuration file - $(file)</description>
  </rule>
</group>
```

Figure 26: New rule for which the script will be run

- Restart the Wazuh manager service to apply changes

```
root@CSE406-wazuh-server:/var/ossec/etc/rules# systemctl restart wazuh-manager
root@CSE406-wazuh-server:/var/ossec/etc/rules#
```

Figure 27: Restart the Wazuh manager

- Go to the Linux endpoint. Edit the `/var/ossec/etc/ossec.conf` file and add the following configuration to the `<syscheck>` section.

```
<directories realtime="yes">/var/ossec/etc/ossec.conf</directories>
```

Figure 28: Add config file for monitoring

- Restart the Wazuh agent.
- Now change the configuration file anyway you like. Add the following block in the ¡syscheck¿ block of the Wazuh agent `/var/ossec/etc/ossec.conf` configuration file and save it.

4.3.4 Dashboard Update

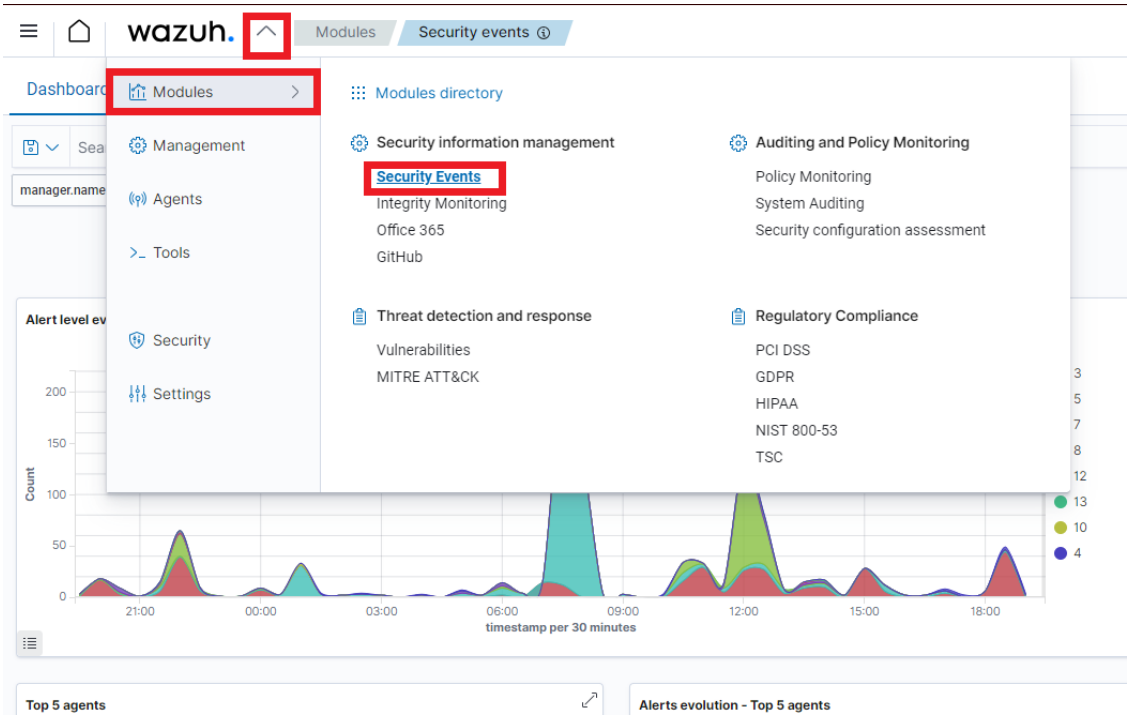


Figure 29: Navigate to Security Events section

Check the security events section. You can visualize the alert data on the Wazuh dashboard. We shall see an alert for the configuration file change and two more alerts for the restart.

Figure 30: Alerts for restart of the agent

4.4 Log Data Analysis

4.4.1 how it works

Log data collection is the process of gathering and centralizing logs from various sources within a network. This crucial practice aids security teams in achieving regulatory compliance, uncovering and addressing security threats, and pinpointing application errors and other security-related issues.

Wazuh plays a vital role in this process by effectively collecting, analyzing, and storing logs originating from endpoints, network devices, and applications. The Wazuh agent, which operates on monitored endpoints, is responsible for gathering and transmitting system and application logs to the Wazuh server for thorough analysis. Additionally, it is possible to send log messages to the Wazuh server through syslog or by leveraging third-party API integrations.

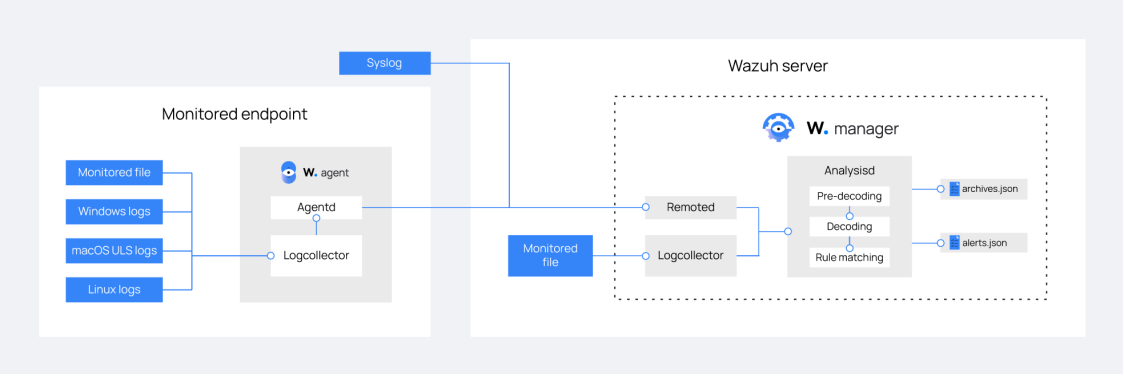


Figure 31: Mechanism of Log Data Collection

Wazuh uses the Logcollector module to collect logs from monitored endpoints, applications, and network devices. The Analysis 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 your entire network environment.

Log data analysis within Wazuh consists of three essential phases:

- **Pre-decoding**: During this initial phase, the log data is prepared for analysis. This involves collecting raw log data generated by various sources such as network devices, endpoints, and applications. This data is then organized and pre-processed to make it suitable for further analysis.
- **Decoding**: In the decoding phase, the pre-processed log data is translated into a structured and readable format. This step is crucial because different sources may use various formats or encoding methods. Decoding ensures that the log data is normalized and consistent, making it easier to analyze and work with.
- **Rule Matching**: The final phase involves applying predefined rules to the decoded log data. These rules are designed to detect specific patterns, anomalies, or security threats within the log data. When

a rule matches a pattern or condition in the log data, it triggers an alert or action, enabling IT teams to respond promptly to security incidents, performance issues, compliance violations, or other relevant events.

4.4.2 Configuration

We can configure sending log data from agent to server in two ways which are desribed below.

1. Configuration for monitoring log files through Wazuh agent
   - Add the following settings in between the `<ossec_config>` tags of the Wazuh agent configuration file:

     ```
     <localfile>
       <location>/<FILE_PATH>/file.log</location>
       <log_format>syslog</log_format>
     </localfile>
     ```

     Explanation:
     - `location`: the full path of the monitored file
     - `log format`: represents the format of the log

     You can configure Wazuh to dynamically monitor log files on endpoints, adapting to changes based on the date. Add the following settings in between the `<ossec_config>` tags of the Wazuh agent configuration file

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

   - Restart the Wazuh agent
2. Configuration for rsyslog on Wazuh server and agent
   - Add the following configuration to the `/etc/rsyslog.conf` file on the Linux endpoint:

     ```
     *.info@@<WAZUH_SERVER_IP_ADDRESS>:514
     ```

   - Add the configuration below in between the `<ossec_config>` tags of the `/var/ossec/etc/ossec.conf` file on the Wazuh server. This configuration allows the Wazuh server to listen for remote syslog messages from the Linux endpoint.

     ```
     <remote>
       <connection>syslog</connection>
       <port>514</port>
       <protocol>tcp</protocol>
       <allowed-ips>192.168.2.15</allowed-ips>
       <local_ip>192.168.2.5</local_ip>
     </remote>
     ```

     Explanation:
     - `connection`: specifies the type of connection to accept.
     - `port`: is the port used to listen for incoming events from the Linux endpoint
     - `allowed-ips`: ips of the Linux endpoints
     - `local-ip`: local ip of the Wazuh server
   - Restart the Wazuh manager

4.4.3 Simulation

We are going to demonstrate log data collection by adding and deleting an user in the agent which will create an entry in the auth.log file.

- Add the auth.log file in the `/var/ossec/etc/ossec.conf` file.

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/ossec/logs/active-responses.log</location>
</localfile>

<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/auth.log</location>
</localfile>
```

Figure 32: Log files to be monitored

- If you want to use rsyslog, edit the `/etc/rsyslog.conf` file in the agent and add the following configuration

```
$IncludeConfig /etc/rsyslog.d/*.conf
*.info@@4.213.17.59:514
```

Figure 33: Configuration for rsyslog

- Restart Wazuh agent
- 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>10.0.0.4</allowed-ips>
<local_ip>10.0.0.5</local_ip>
</remote>
```

Figure 34: Configuration for rsyslog

- Restart the Wazuh manager for the changes to take effect
- Add and delete user Rumi

```
root@CSE406-wazuh-agent-linux:/# useradd Rumi
root@CSE406-wazuh-agent-linux:/# userdel Rumi
root@CSE406-wazuh-agent-linux:/#
```

Figure 35: User creation and deletion

There will be new entries in the auth.log file which will be collected by the agent and sent to the server for analysis.

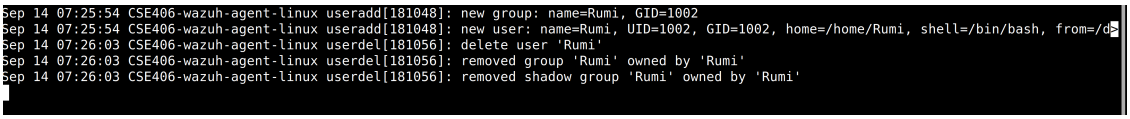
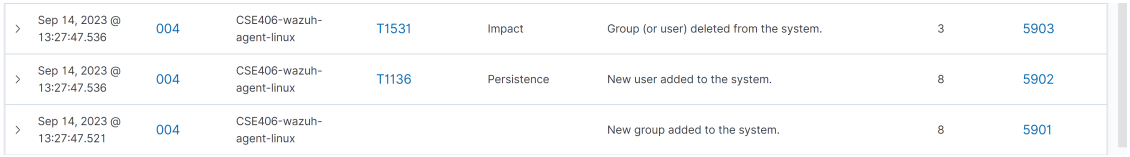


Figure 36: Configuration for rsyslog

4.4.4 Dashboard Update

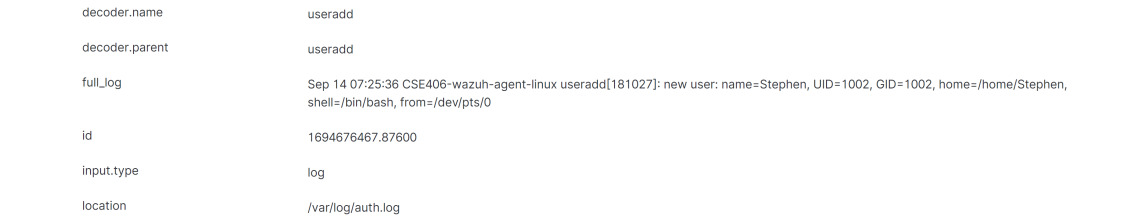
- After adding and deleting user, the dashboard will show new security event about the creation and deletion of user.



›	Sep 14, 2023 @ 13:27:47.536	004	CSE406-wazuh-agent-linux	T1531	Impact	Group (or user) deleted from the system.	3	5903
›	Sep 14, 2023 @ 13:27:47.536	004	CSE406-wazuh-agent-linux	T1136	Persistence	New user added to the system.	8	5902
›	Sep 14, 2023 @ 13:27:47.521	004	CSE406-wazuh-agent-linux			New group added to the system.	8	5901

Figure 37: New alerts

- Expand the alert to see the full log and decoders used for analysis.



decoder.name	useradd
decoder.parent	useradd
full_log	Sep 14 07:25:36 CSE406-wazuh-agent-linux useradd[181027]: new user: name=Stephen, UID=1002, GID=1002, home=/home/Stephen, shell=/bin/bash, from=/dev/pts/0
id	1694676467.87600
input.type	log
location	/var/log/auth.log

Figure 38: Full Log and decoders

- All the details of rule matching step will also be visible

rule.description	New user added to the system.
rule.firedtimes	2
rule.gdpr	IV_35.7.d, IV_32.2
rule.gpg13	4.13
rule.groups	syslog, adduser
rule.hipaa	164.312.b, 164.312.a.2.I, 164.312.a.2.II
rule.id	5902
rule.level	8
rule.mail	false
rule.mitre.id	T1136

Figure 39: Rule Matching

5 Codebase

We give a high-level overview of the codebase in this section. The link of the codebase: Wazuh Codebase. we take one section of the codebae at a time and give a very brief description of it.

- **Architecture**: In this part, wazuh-db folder contains a daemon that wraps the access to SQLite database files. It provides automatic database upgrades, serialized and parallel queries to the database and so on. Metrics folder contains includes some metrics that help to understand the behaviour of Wazuh components. The wazuh agents are able to collect interesting and valuable system information regarding processes, hardware, packages, OS, network and ports. syscollector folder implements this. It contains modules named `Syscollector, Data Provider, DBSync, RSync`.
- **API**: This folder contains codes to handle all Wazuh apis and Wazuh api installer functions.
- **Active-response**: This section handles the active response scripts. Most of the files in this folder are implementations of default Wazuh scripts like restart-wazuh, host-deny, disable-account and and so on.
- **addagent**: This section contains codes for managing Wazuh agents. Management of keys of the agents, connections of agents with servers- all of these are present here.
- **agentlessd**: Agentless entry in the host through SSH connections is handled in this section.
- **analysisd**: This is a very important section. The analysis of the events that take place in the agent and analysis of the collected log are coded here. alerts folder stores the different kinds of alerts we see in the security event section of the dashboard. We know that analysis comprises three important steps. Pre-decoding, decoding and rule-matching. All the codes for default decoders and pre-decoders are present in this folder and internal subfolders. The compiled-rules subfolder contains all the rules to be matched and relevant codes. Moreover, this section analyzes the events and creates and stores new events to show on the dashboard.
- **agent-client**: This folder contains codes to handle the state of the agent residing in the client endpoint. Restarting the client, storing and updating the state of the client. forwarding security events that take place in clients are managed here.
- **error_messages**: This folder contains some header files that comprise various error messages used throughout the Wazuh components.
- **headers**: Comprised of miscellaneous headers and utility functions. Applications of these range from cryptography to even file queue.
- **init**: This folder handles new user and group creation and deletion and removing old users. Registration of new agents to servers are also a part of this.
- **logcollector**: Log Data Collection is an important feature of Wazuh. We can configure this in two ways. Either we can collect data by the agents in the endpoint or send the logs directly to the server using the protocol rsyslog. Both of these procedures are contained in the folder.
- **monitord**: Monitors logs and generates reports based on logs.
- **remoted**: This folder possesses implementations of remote communication.
  - sending message
  - syslog protocol
  - shared download
- **reportd**: Generates report based on several input parameters.
- **Rootcheck**: Rootcheck allows defining policies to check if the agents meet the specified requirement. The rootcheck engine can perform the following checks:

  - Check if a process is running
  - Check if a file is present
  - Check if the content of a file contains a pattern or if a Windows registry key contains a string or is simply present

  This feature is implemented in this section.

- **utils:** Contains helper functions and code related to the following:
  - agent_control - (eg reconfiguring agent after restart)
  - list_agents - Lists (prints/displays) the agents and their status (Active, not Active, available etc). May be related to manage_agents function in server.
  - verify-agent-config - performs checks on the integrity of an agent and reports them. Also houses switch-cases (for navigation from the command line, it seems)
- **wazuh_db**: Contains schema definitions for the different databases in wazuh, for example:
  - Agent
  - Modules (covered later)
  - upgrades (all versions)
  - rootcheck etc.
- **wazuh_modules** Houses some main modules, like
  - agent_upgrade - facilitates upgrade of an agent (for example, by agentID).
  - syscollector - seems to gather information about the following:
    * Operating System information
    * List of running processes
    * Installed software and versions
    * Open ports and established network connections
    * Hardware details like CPU, memory, and disk information
    * Network interface and routing details
  - task manager - coordinates and schedules tasks that need synchronization between the manager and the agent like
    * Upgrades (actual upgrade scheduling, success/error messages, cancel upgrade, get upgrade status etc)
    * Message sending to database (also may parse data responses in this context) etc

6 Problems and Solutions:

6.1 Unable to add 3rd VM with Wazuh Windows Agent:

- We initially planned to work with a Wazuh Server (Linux) and 2 wazuh agents (a windows one and a linux one)
- The Azure VM students' subscription only allows 4 VCPU's. These were used up after creating the Wazuh server and the linux agent.
- We then proceeded to open the windows agent on another account and then link the 3 under the same vnet. This required vnet-peering
- But due to complicacies with the Azure Active Directory, this approach failed.

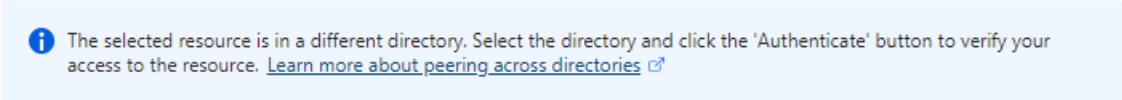


Figure 40: AD complexities and authentication error

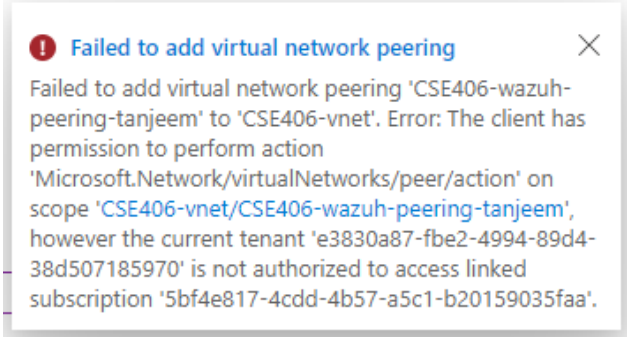


Figure 41: AD complexities and authentication error

6.2 Server-side Agent Registration:

- As previously discussed, after an agent and a server have been created, the agent must be registered to the server.
- This registration can take place from either the server side or the agent side.
- The client-side method worked (required an additional ping from agent to server), but the server-side method did not.
- After executing the client side, an "anomaly" was noticed. By default, the agent's IP is stored as "any" in the server's list (Probably to accommodate DHCP IP allocation in real-life scenarios). We can manually change this to the actual static private IP of the agent, but we've kept it at "any" for now.
- The server-side approach showed an entry in the agents list after adding an agent. But, when accessed from the dashboard, it showed `Agent Never Connected`
- No solution to this problem was found.

7 Reference(s):

- The Wazuh Official Documentation