GUJARAT UNIVERSITY
BATCH 2021-2024

SENTINEL SECURE

SAFEGUARDING NETWORKS WITH WAZUH GUARD

Submitted By :
Sunil Upadhyay

Guided By :
Pooja Mathur

DEPARTMENT OF ANIMATION, IT IMS & MOBILE APPLICATIONS, GUJARAT UNIVERSITY

CERTIFICATE

Enrolment Number : 202118100194

THIS IS TO CERTIFY THAT **MR. Upadhyay Sunil Prahladbhai** STUDENT OF B.SC. IT IMS AND CS (INTEGRATED) SEMESTER – 6, HAS DULY COMPLETED HIS/HER TERM WORK FOR THE SEMESTER ENDING IN JUNE 2024, IN THE SUBJECT OF PROJECT TOWARDS PARTIAL FULFILMENT OF HIS DEGREE OF BACHELOR PROGRAM.

DATE OF SUBMISSION

MENTOR

Pooja Mathur

ACKONWLEDGEMENT

Primarily I would thank God for being able to complete this project with success. Then I would like to thank all those who helped me directly or indirectly in our research and work. I would like to thank my project guide **Pooja Mathur ma'am** , whose valuable guidance has been the ones that helped me patch this project and make it full proof success his suggestion and instructions has served as the major contribute towards the completion or the project.

I also want to thank the pillar, inspiration, and support. Last but not the least I would like to thank my classmates who have helped me a lot.

Project Overview

- Our project is about implementing Wazuh, an open-source security monitoring platform, in our organization's digital infrastructure.
- We'll cover various aspects of Wazuh, including its core components like the Wazuh Manager and Wazuh Agents, as well as services such as Elasticsearch, Filebeat, and Kibana.
- Understanding installation and deployment requirements, hardware/software prerequisites, and configuration are crucial for a smooth implementation.
- Communication between Wazuh Agents and the Wazuh Manager is vital for real-time monitoring and response.
- Browsing security alerts provides insights into potential threats, while integrity monitoring ensures critical files and configurations are secure.
- Detecting vulnerable software helps in proactive security measures, and system auditing tracks user activity and changes.
- Integrating with Outlook and Telegram expands notification capabilities, and automated monitoring ensures system health and performance.
- Lastly, integrating with Windows Defender alerts enhances endpoint security posture, providing comprehensive threat detection and response capabilities.

TABLE OF CONTENTS

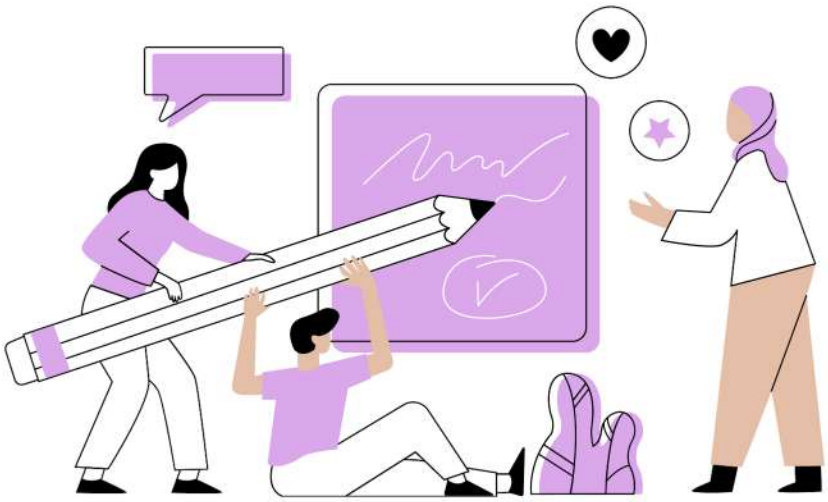
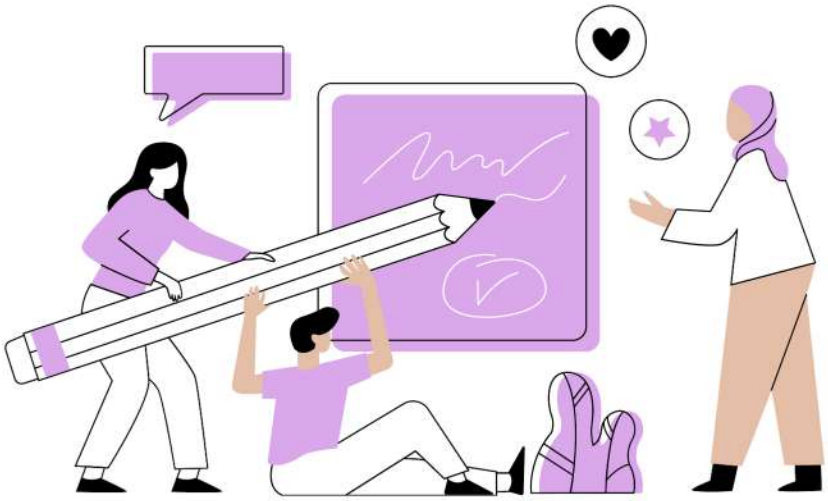


TABLE OF CONTENTS



Chapter 1
Introduction to Wazuh

- Wazuh is an advanced open-source security monitoring platform designed to provide comprehensive threat detection, incident response, and compliance management capabilities.
- It offers a robust set of features and functionalities to help organizations enhance their security posture and effectively combat cyber threats.

1.1 Overview of Wazuh as an Advanced Security Monitoring Platform

1.1.1 What is Wazuh and Why Choose it Over Alternatives

- Wazuh combines the capabilities of an intrusion detection system (IDS), a log analysis system, and a host-based intrusion detection system (HIDS) into one powerful platform.
- Unlike some proprietary solutions, Wazuh is open-source, providing transparency, flexibility, and community-driven support. Here's why choosing Wazuh over alternatives makes sense:

1.1.1.1 Cost-effectiveness

- Wazuh is open-source, meaning it's free to use and doesn't come with licensing fees.
- This makes it a cost-effective solution, particularly for organizations with limited budgets.

1.1.1.2 Flexibility and Customization

- Wazuh's open-source nature allows for extensive customization and integration with existing security tools and systems.
- Organizations can tailor Wazuh to their specific needs and environments, ensuring it fits seamlessly into their existing infrastructure.

1.1.1.3 Comprehensive Security Coverage

- Wazuh offers a wide range of features, including log analysis, file integrity monitoring (FIM), intrusion detection, vulnerability detection, and incident response automation.

1.1.1.4 Scalability

- Wazuh is highly scalable and can adapt to the needs of organizations of all sizes, from small businesses to large enterprises.
- It can handle high volumes of data and can be deployed in diverse environments, including on-premises, in the cloud, and in hybrid setups.

1.1.2 Alternatives to Wazuh Comparison with Other Tools

- While Wazuh offers a comprehensive set of features and advantages, it's essential to consider alternatives and compare them based on specific requirements and use cases.
- Here's a comparison of Wazuh with some popular alternatives:

1.1.2.1 Wazuh vs. Splunk

- Splunk is a powerful commercial SIEM solution known for its data analytics and visualization capabilities.
- Wazuh, being open-source, is more cost-effective, but Splunk offers more advanced analytics and reporting features.
- Splunk may be preferable for organizations requiring extensive data analysis and visualization capabilities, while Wazuh is suitable for those prioritizing cost-effectiveness and flexibility.

1.1.2.2 Wazuh vs. OSSEC

- OSSEC is another open-source HIDS solution, from which Wazuh was forked, but Wazuh extends its capabilities significantly.
- Wazuh includes additional features such as centralized management, file integrity monitoring, and integration with popular tools like ELK Stack (Elasticsearch, Logstash, Kibana).
- While OSSEC is suitable for basic host-based intrusion detection, Wazuh provides a more comprehensive security monitoring solution.

2.2 Highlight its features and capabilities in threat detection and response

- Wazuh offers a wide range of features and capabilities focused on threat detection and response, empowering organizations to effectively defend against cyber threats. Here are some key features:

2.2.1 Intrusion Detection System (IDS)

- Wazuh includes an intrusion detection system that monitors network traffic and system logs for signs of intrusion attempts or malicious activity.
- It detects and alerts on known attack signatures, zero-day exploits, and abnormal behavior.

2.2.2 Vulnerability Detection

- Wazuh scans systems for vulnerabilities, identifying outdated software, misconfigurations, and potential security weaknesses.
- It provides actionable insights to remediate vulnerabilities and reduce the attack surface.

2.3.3 Threat Hunting and Incident Response

- Wazuh enables security teams to proactively hunt for threats within their environment, conducting searches and investigations based on custom queries and threat indicators.
- It facilitates incident response by providing actionable insights and automating response actions.

2.3.4 Integration with SIEM and Threat Intelligence

- Wazuh integrates with SIEM (Security Information and Event Management) solutions, enhancing visibility and correlation capabilities.
- It also integrates with threat intelligence feeds to enrich security data and improve threat detection accuracy.

Chapter 2
Architecture Components

2.1 Roles and Responsibilities of Wazuh Manager & Agent

2.1.1 Wazuh Manager

- The Wazuh Manager serves as the central hub for security monitoring, analysis, and response within the organization's digital infrastructure.
- The Wazuh Manager centrally manages and orchestrates security monitoring activities across all connected endpoints.
- It aggregates security-related data collected from Wazuh Agents deployed across the network.
- The Manager normalizes and parses incoming data into a standardized format, ensuring consistency and compatibility for analysis.
- It correlates security events and analyzes them for signs of suspicious behavior, potential threats, or compliance violations.
- The Wazuh Manager generates alerts based on detected security incidents and triggers automated response actions.
- It provides dashboards and reports to visualize security events, trends, and metrics.

2.1.2 Wazuh Agent

- The Wazuh Agent is deployed on endpoints throughout the organization's network to collect security data and send it to the Wazuh Manager.
- Wazuh Agents collect security-related data from endpoints, including logs, events, and system information.
- They monitor activities such as file changes, user logins, network connections, and application usage.
- Agents continuously monitor endpoint activities and events in real-time, detecting any suspicious behavior or security threats.
- They detect security incidents and anomalies based on predefined rules and configurations
- Agents forward collected logs and security data to the Wazuh Manager for centralized analysis and storage.
- Wazuh Agents integrate with various systems and applications, enabling seamless communication and collaboration.

2.2 Explanation of how Wazuh Manager collects and analyzes security data from agents

- The Wazuh Manager acts as the central component for collecting, processing, and analyzing security data from Wazuh Agents.
- Here's how it works:

- Wazuh Agents send collected security data to the Wazuh Manager in real-time or at predefined intervals.
- This data includes logs, events, and alerts generated by the agents' monitoring activities.
- The Wazuh Manager aggregates incoming data from multiple agents into a centralized repository.
- It organizes the data for efficient storage and retrieval, allowing for easy access and analysis.
- Security data collected from agents may come in different formats and structures.
- The Wazuh Manager normalizes this data into a standardized format for consistency and ease of analysis. Normalization ensures that all security events are interpreted uniformly, regardless of the source.
- Once the data is aggregated and normalized, the Wazuh Manager analyzes it for signs of security threats, anomalies, or compliance issues.
- It applies predefined correlation rules, threat intelligence feeds, and behavioral analysis techniques to identify potential security incidents.
- When the Wazuh Manager detects a security incident or anomaly, it generates alerts for further investigation and response.
- These alerts can be sent to security analysts, administrators, or other designated personnel via email, SMS, or integration with third-party tools.

Chapter 3
Essential Services in the Wazuh Ecosystem

3.1 Understanding Elasticsearch, Filebeat, and Kibana in Wazuh

3.1.1 Elasticsearch

- What is Elasticsearch? Elasticsearch is a distributed, RESTful search and analytics engine built on top of Apache Lucene.
- It is designed for storing, searching, and analyzing large volumes of data in real-time.
- Role in Wazuh: Elasticsearch serves as the primary data store for Wazuh, storing all security-related data collected from Wazuh Agents.

3.1.2 Filebeat

- What is Filebeat? Filebeat is a lightweight shipper for forwarding log files to Elasticsearch or Logstash for indexing and analysis.
- It is part of the Beats family, which is used for various data shipping and collection tasks.
- Role in Wazuh: Filebeat is responsible for collecting logs and forwarding them to Elasticsearch.
- It ensures that log data from Wazuh Agents is efficiently transported to Elasticsearch for storage analysis.

3.1.3 Kibana

- What is Kibana? Kibana is an open-source data visualization dashboard for Elasticsearch.
- It provides visualization capabilities, including charts, graphs, and maps, to help users understand and explore their data.
- Role in Wazuh: Kibana acts as the user interface for Wazuh, allowing administrators and analysts to visualize and analyze security data stored in Elasticsearch.
- It provides dashboards and visualizations to monitor security events, investigate incidents, and generate reports.

3.2 Explanation of their roles in data storage, log collection, and visualization

3.2.1 Elasticsearch

Data Storage:

- Elasticsearch stores all security-related data collected by Wazuh Agents, including logs, events, and alerts.
- It indexes the data for fast and efficient retrieval and supports real-time searching and analysis

3.2.2 Filebeat

Log Collection:

- Filebeat collects logs from Wazuh Agents and other sources and forwards them to Elasticsearch.
- It monitors log files, processes log events, and sends the data to Elasticsearch for indexing.

3.2.3 Kibana

Visualization:

- Kibana provides a user-friendly interface for visualizing and exploring data stored in Elasticsearch.
- It offers various visualization tools, including charts, graphs, and maps, to help users gain insights into security events and trends.

Data Analysis

- Kibana allows users to analyze security data, create custom dashboards, and perform ad-hoc searches.
- It enables security analysts to identify patterns, investigate incidents, and generate reports to support decision-making and incident response.

Chapter 4
Knowing what we need to set up Wazuh

4.1 Detailed explanation of hardware and software requirements for setting up Wazuh

- Wazuh is a security platform that provides unified XDR and SIEM protection for endpoints and cloud workloads. The solution is composed of a single universal agent and three central components.
- Below you can find a section about the requirements needed to install Wazuh. It will help you learn about the hardware requirements and the supported operating systems for your Wazuh installation.

Requirements

Hardware

- Hardware requirements highly depend on the number of protected endpoints and cloud workloads.
- This number can help estimate how much data will be analyzed and how many security alerts will be stored and indexed.

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

Operating system

- Wazuh central components can be installed on a 64-bit Linux operating system.
- Wazuh recommends any of the following operating system versions:

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

Browser compatibility

- Wazuh dashboard supports the following web browsers
  - **Chrome 95 or later**
  - **Firefox 93 or later**
  - **Safari 13.7 or later**
- Other Chromium-based browsers might also work. Internet Explorer 11 is not supported.

4.2 Step-by-step guide on installing and deploying Wazuh components

- We are doing all in one deployment with the wazuh components including Wazuh indexer, Wazuh Server, Wazuh Dashboard with the service of Elastic-Search, Filebeat and Kibana.

Wazuh indexer

- The Wazuh indexer is 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.
- To Install **Indexer** refer below given link.
- https://documentation.wazuh.com/current/installation-guide/wazuh-indexer/step-by-step.html

Wazuh Server

- The Wazuh server analyzes the data received from the Wazuh agents, triggering alerts when threats or anomalies are detected.
- It is also used to remotely manage the agents' configuration and monitor their status.
- To Install **Server** refer below given link.
- https://documentation.wazuh.com/current/installation-guide/wazuh-server/step-by-step.html

Wazuh dashboard

- This central component is a flexible and intuitive web interface for mining, analyzing, and visualizing security data.
- It provides out-of-the-box dashboards, allowing you to seamlessly navigate through the user interface.
- To Install **Dashboard** refer below given link.
- https://documentation.wazuh.com/current/installation-guide/wazuh-dashboard/step-by-step.html

Elasticsearch, Kibana, Filebeat

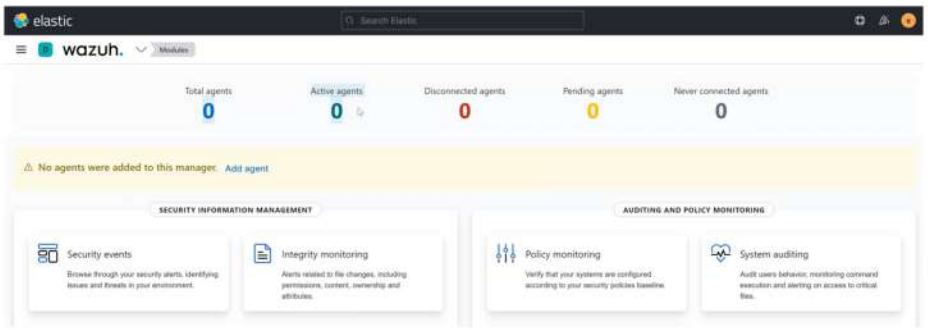
- This installation guide will use the Elastic Stack basic license option, which contains everything included in the open-source version under the Apache 2.0 license, plus additional capabilities such as Elastic Stack Security features, Kibana alerting, and others.
- To Install **ELK-STACK** refer below given link.
- https://documentation.wazuh.com/4.5/deployment-options/elastic-stack/all-in-one-deployment/index.html

Chapter 5
Secure Communication Channels

5.1 Ensuring Secure Data Transmission between Wazuh Agents and Manager

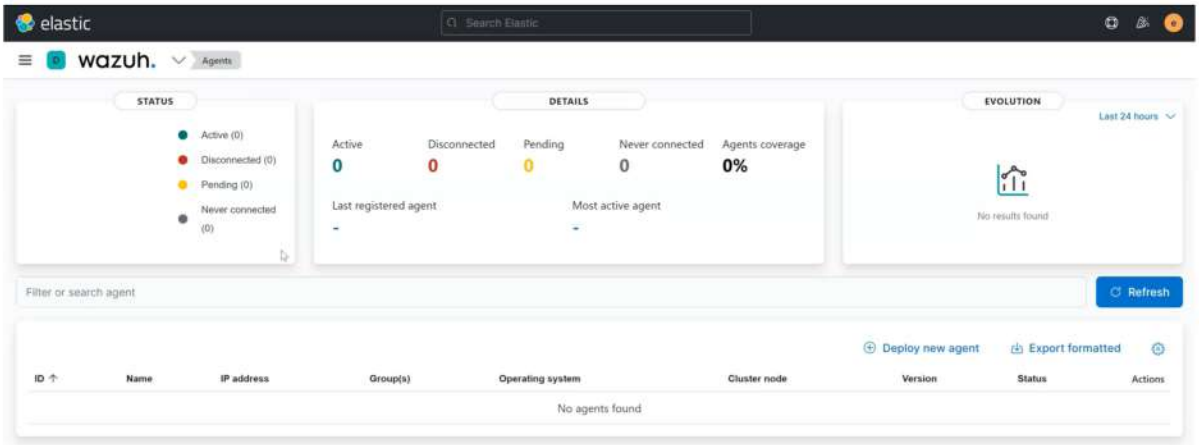
Let's add an Ubuntu agent to the Wazuh Manager and establish secure communication between them

5.1.1 First, let's open the Wazuh dashboard to see the current status. As shown in the image below, you'll notice there are 0 agents and 0 live agents, indicating that no agents are currently added.

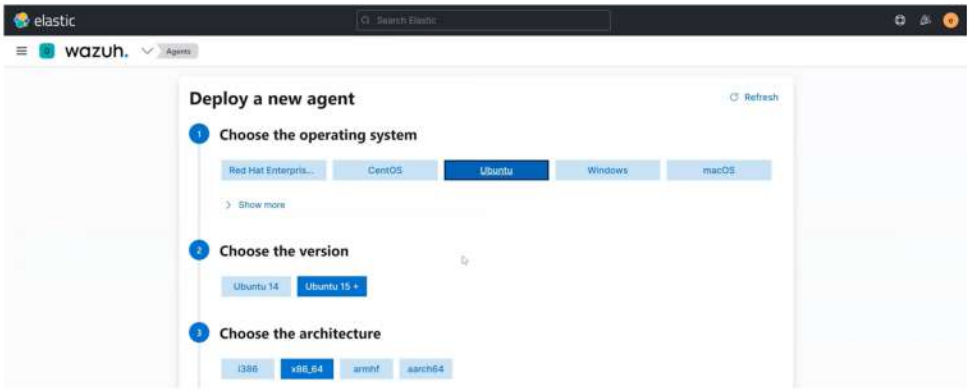


5.1.2 To start adding an agent, click on the "Add Agent" option (Check the image shown as in the step of 5.1.1)

5.1.3 You'll be directed to a new page. Here, click on "Deploy a new agent"



5.1.4 Next, you'll choose the operating system. Since we're adding an Ubuntu agent, select Ubuntu.

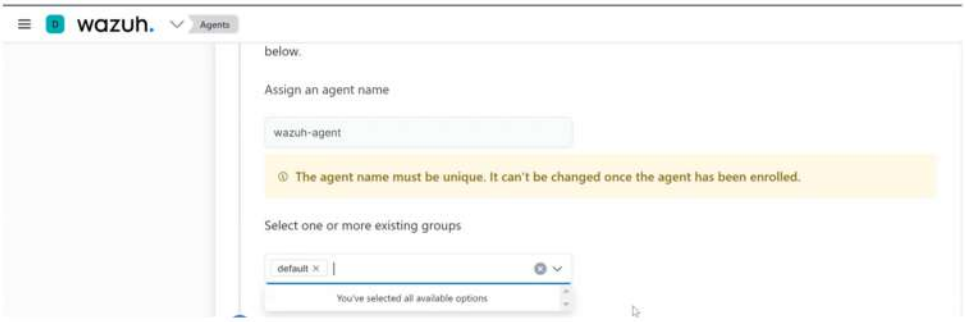


5.1.5 Choose the appropriate architecture for your operating system, and then proceed to the next step.(check the image in the step of 5.1.4)

5.1.6 Now, provide the IP address of your Wazuh server. You can find this by opening your Wazuh server's terminal and typing "ifconfig". Copy the IP address and paste it into the server IP address field in the Wazuh dashboard.

```
root@wazuh-elk:/usr/share/kibana# ifconfig
ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.160.0.2  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::4001:aff:fea0:2  prefixlen 64  scopeid 0x20<link>
        ether 42:01:0a:a0:00:02  txqueuelen 1000  (Ethernet)
        RX packets 79354  bytes 951091946 (951.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 40480  bytes 31159815 (31.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5.1.7 Assign a name to the agent and select the default group



5.1.8 In the Wazuh dashboard, you'll find a link like the one shown below. Copy this link and run it in your Ubuntu Wazuh agent.

```
root@wazuh-agent:~# curl -so wazuh-agent.deb https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.7.3-1_amd64.deb && sudo WAZUH_MANAGER='10.160.0.2' WAZUH
_AGENT_GROUP='default' WAZUH_AGENT_NAME='wazuh-agent' dpkg -i ./wazuh-agent.deb
Selecting previously unselected package wazuh-agent.
(Reading database ... 66095 files and directories currently installed.)
Preparing to unpack ./wazuh-agent.deb ...
Unpacking wazuh-agent (4.7.3-1) ...
Setting up wazuh-agent (4.7.3-1) ...
root@wazuh-agent:~#
```

5.1.9 Now, go to the Wazuh dashboard and copy the three commands provided. Run each command one by one in the Ubuntu Wazuh agent.

7 Start the agent

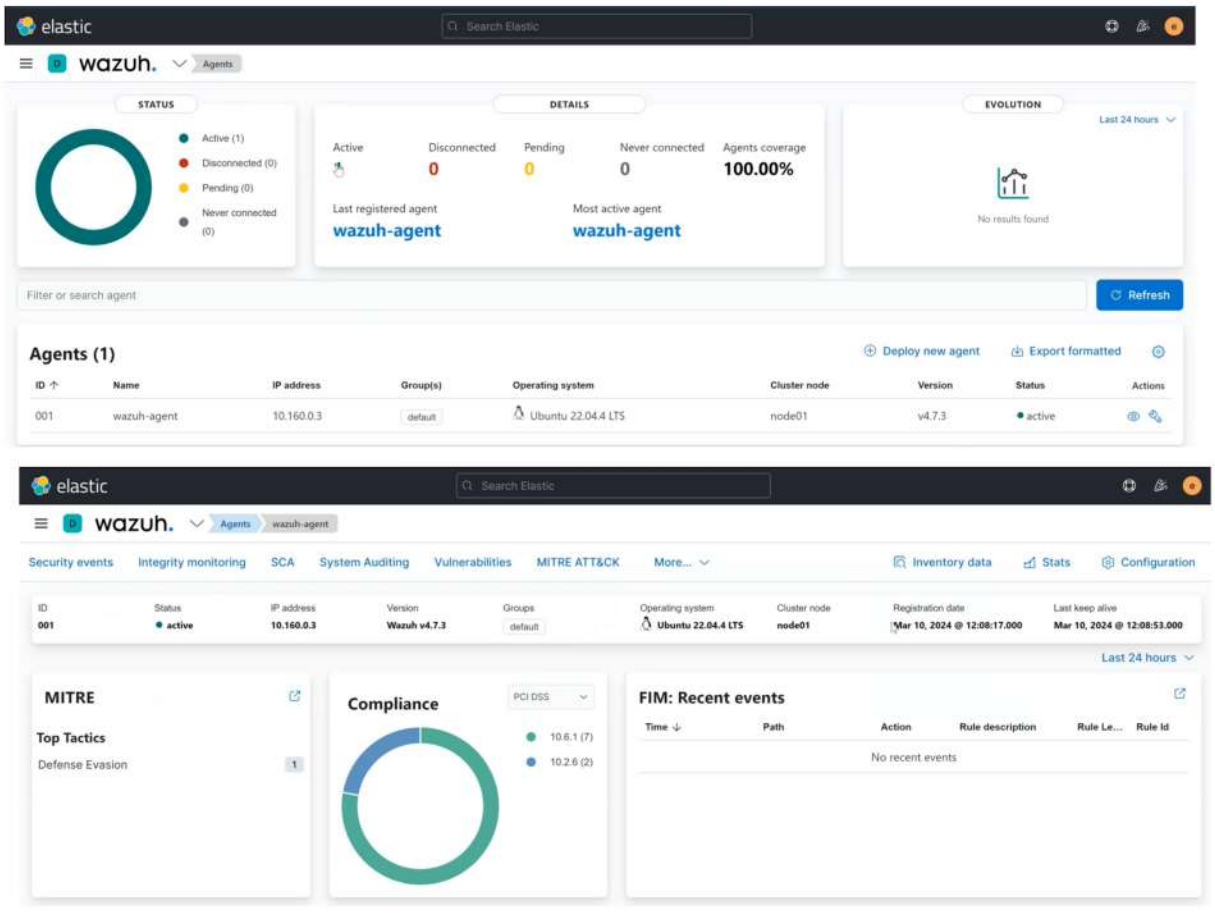
Systemd

```
sudo systemctl daemon-reload
sudo systemctl enable wazuh-agent
sudo systemctl start wazuh-agent
```

```
root@wazuh-agent:~# systemctl daemon-reload
root@wazuh-agent:~#
root@wazuh-agent:~# systemctl enabel wazuh-agent
Unknown command verb enabel.
root@wazuh-agent:~#
root@wazuh-agent:~# systemctl enable wazuh-agent
Created symlink /etc/systemd/system/multi-user.target.wants/wazuh-agent.service → /lib/systemd/system/wazuh-agent.service.
root@wazuh-agent:~#
root@wazuh-agent:~# systemctl start wazuh-agent
root@wazuh-agent:~#
root@wazuh-agent:~# systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
     Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-03-10 06:38:23 UTC; 7s ago
    Process: 4355 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 29 (limit: 9506)
     Memory: 65.0M
        CPU: 2.627s
     CGroup: /system.slice/wazuh-agent.service
             ├─4377 /var/ossec/bin/wazuh-execd
             ├─4388 /var/ossec/bin/wazuh-agentd
             ├─4401 /var/ossec/bin/wazuh-syscheckd
             ├─4414 /var/ossec/bin/wazuh-logcollector
             └─4431 /var/ossec/bin/wazuh-modulesd
```

5.1.10 After running the commands, check the status of the Wazuh agent service to ensure it is active and running properly.(check the image of 5.1.9 step)

5.1.11 Return to the Wazuh dashboard and navigate to the agent page. You will see that our Ubuntu agent is now added and active. You can also verify this by checking its IP address.



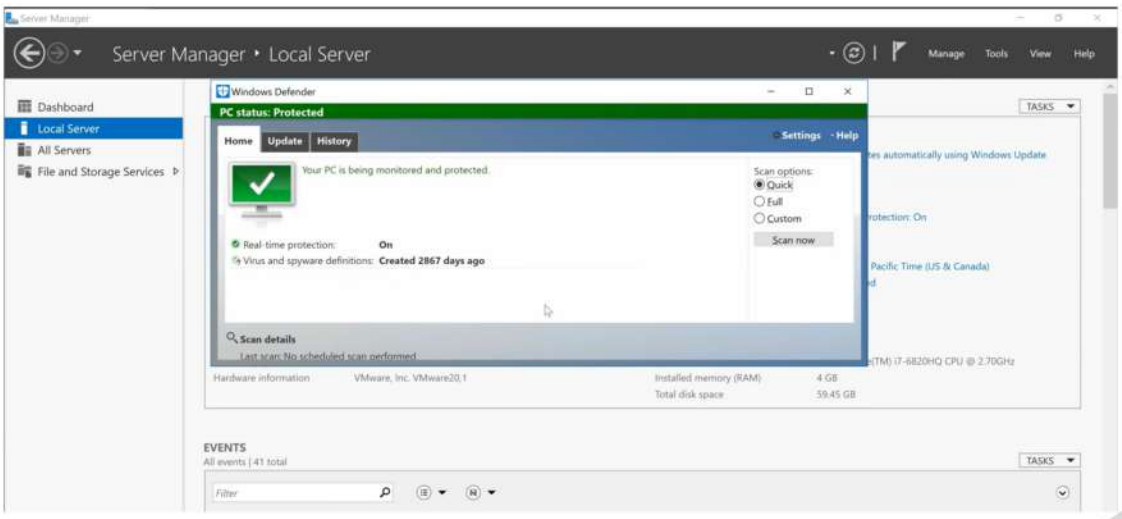
NOTE :- Just same as you can also add other operating system as an agent.In windows you have to run this commands in windows powershell.

Chapter 6
Holistic Threat Monitoring and Response

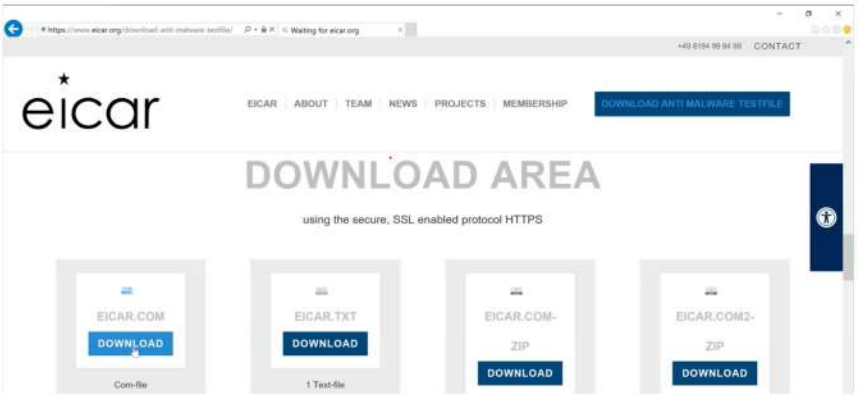
6.1 Seamless Integration of Windows Defender Alerts into Wazuh for Centralized Monitoring

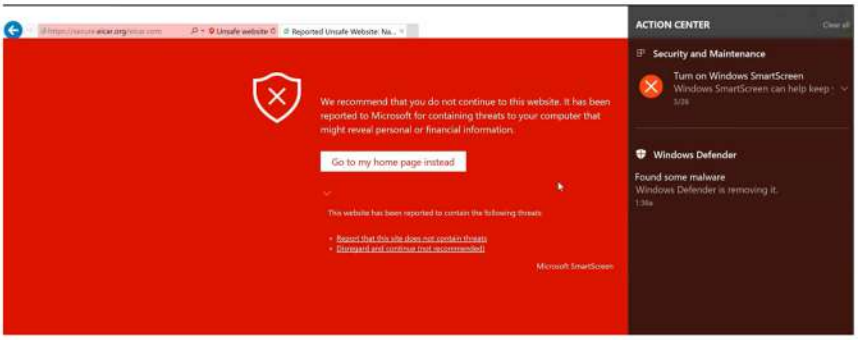
- windows defender which comes pre-installed with all windows operating systems uh and of course is free and in my opinion is a pretty good anti-virus uh solution
- you know it's it still contains a lot of these signature sets that a lot of these commercial anti-virus solutions uh use and
- it also outputs to a windows event log so that we can actually now tell our wazoo agent that's running on all of our windows boxes say hey when this particular event log is seen let's go ahead and send it to wazoo.

- all right so the first thing to do is make sure that windows defender is of course running and has the real-time protection set to enabled to open this gui you can just do it within it's probably with the windows button and search it with windows defender and it will shown to you.

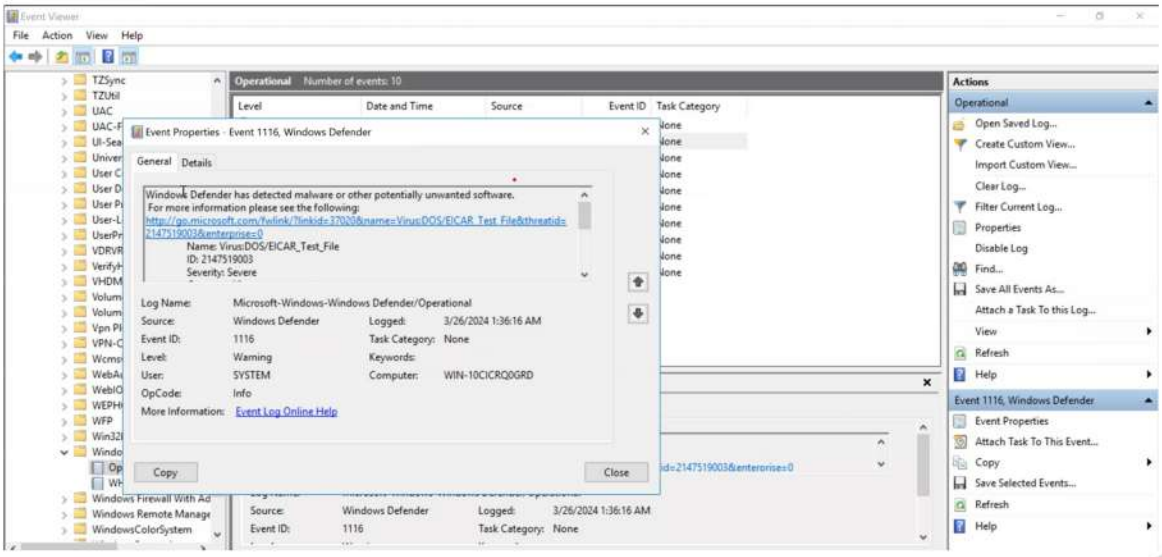


6.1.2 so now that this is running uh now if we try to grab a like a test file so like let's trigger this icar we can see that windows defender has detected and it is in fact actually removing it for us now you know you know where is that log actually output like yeah we got a little notification with a pop-up but wazoo wouldn't be able to read that



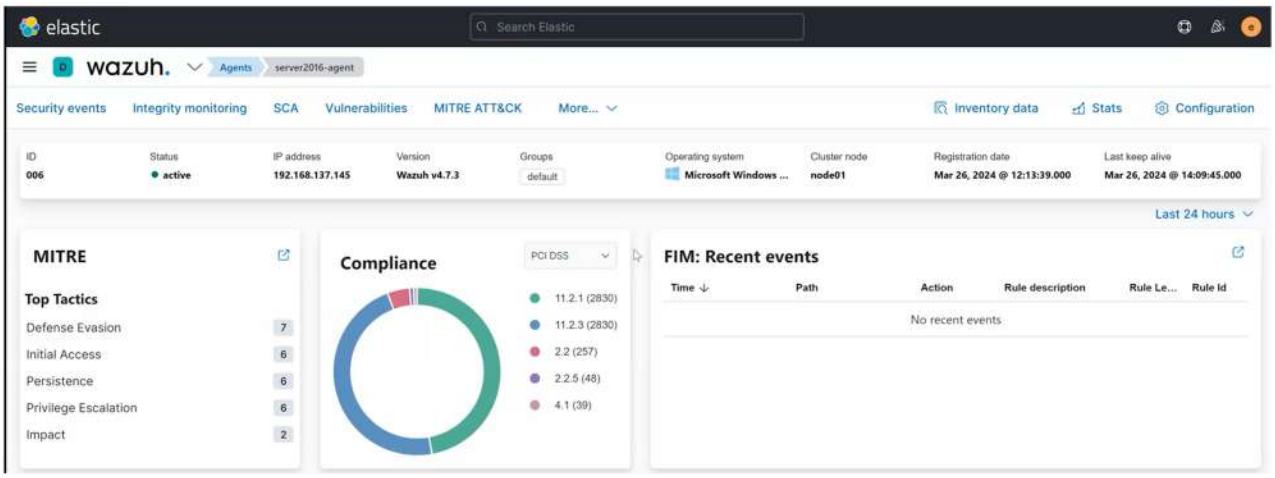


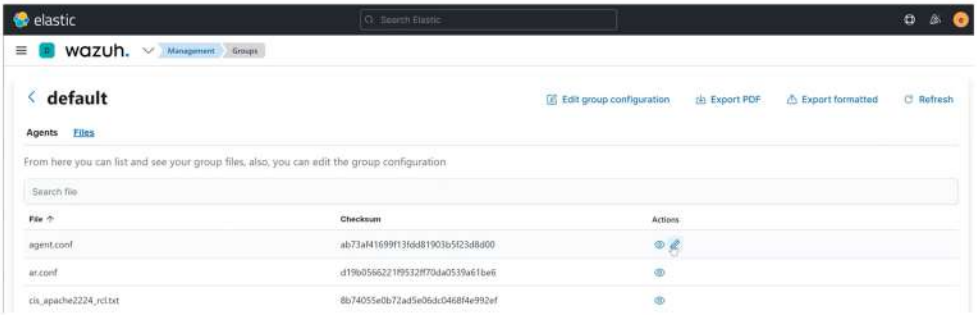
- so by default if we open up our event viewer here we'll be able to drill down to all of our windows defender related logs so if we go into our application and service logs tree here we will then go into microsoft into windows and kind of similar to the sysmon and if we scroll down to our windows defender file here.
- we'll see our operational block so if i go ahead and double click that you'll see that at at we got an alert for our icard test file that we see here and here we see when does the windows defender has detected malware or other potentially unwanted software.



- and so this is where our raw log that windows defender outputs is outputted to what we need to tell our wazoo agent to do is to grab these and forward them to our wazoo manager

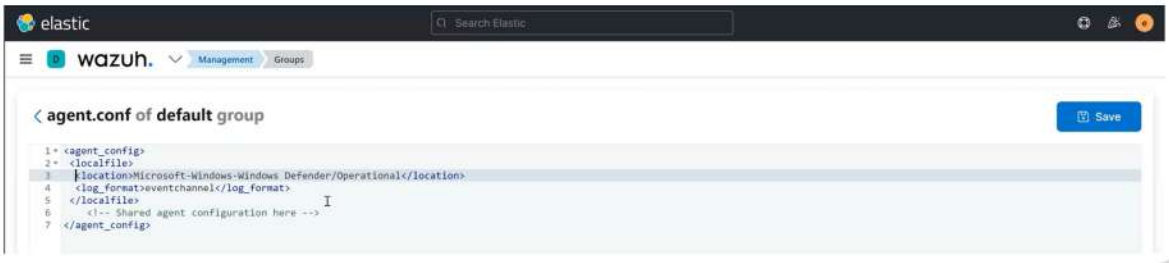
- so if we go into our wazoo plugin here and i'll go under agents and here we see that this this windows agent which is the windows box that i am on is currently within the default group





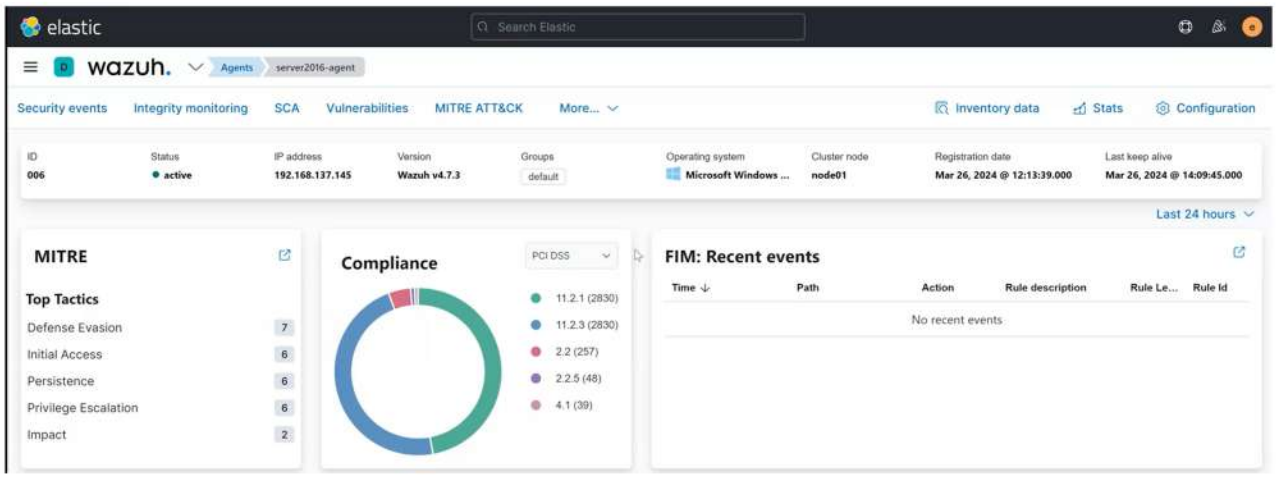
- so we can go ahead and select this group we'll select files and we'll select this agent.conf and we'll be able to edit this

- and once we edit and save this file this will push out to all of the wazoo agents that are within this default group so again what we need to tell our wazoo agent to do is say hey for every event that comes into this particular to this event viewer log which is this microsoft windows windows defender operational i want you to gather that up and send that to our manager so if we go back into here we can just simply add this block here

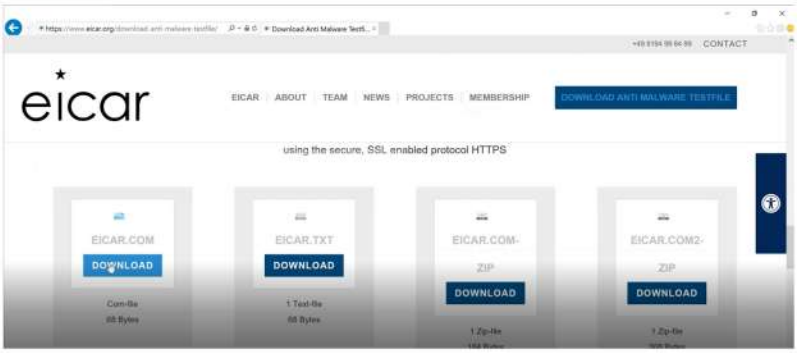


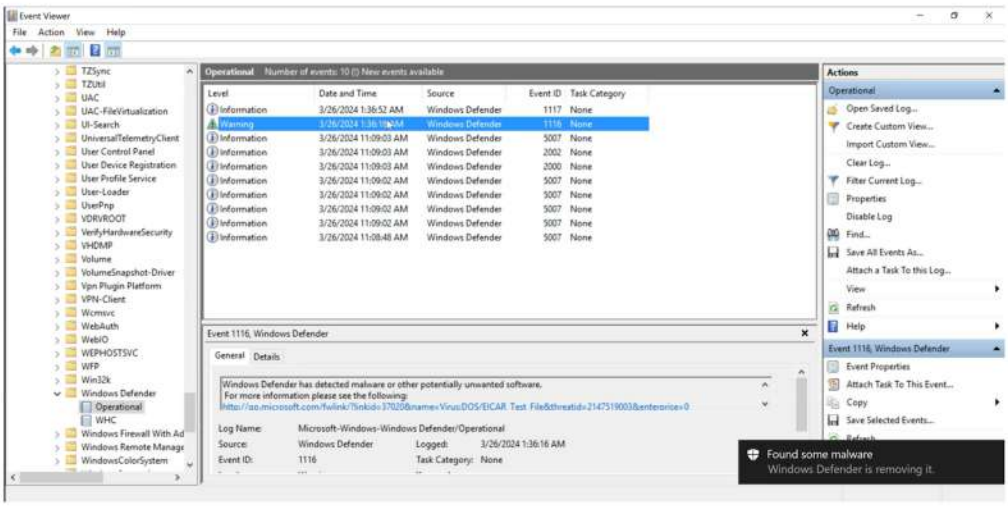
- so once we save this off once that's saved you can always just to make sure that it was saved and saved correctly you can go back into this guy and you should see your changes persisted

- so now our wazoo agent is now going to gather up these logs and send them to our wazoo manager so if we go ahead and jump back on to this agent's particular dashboard



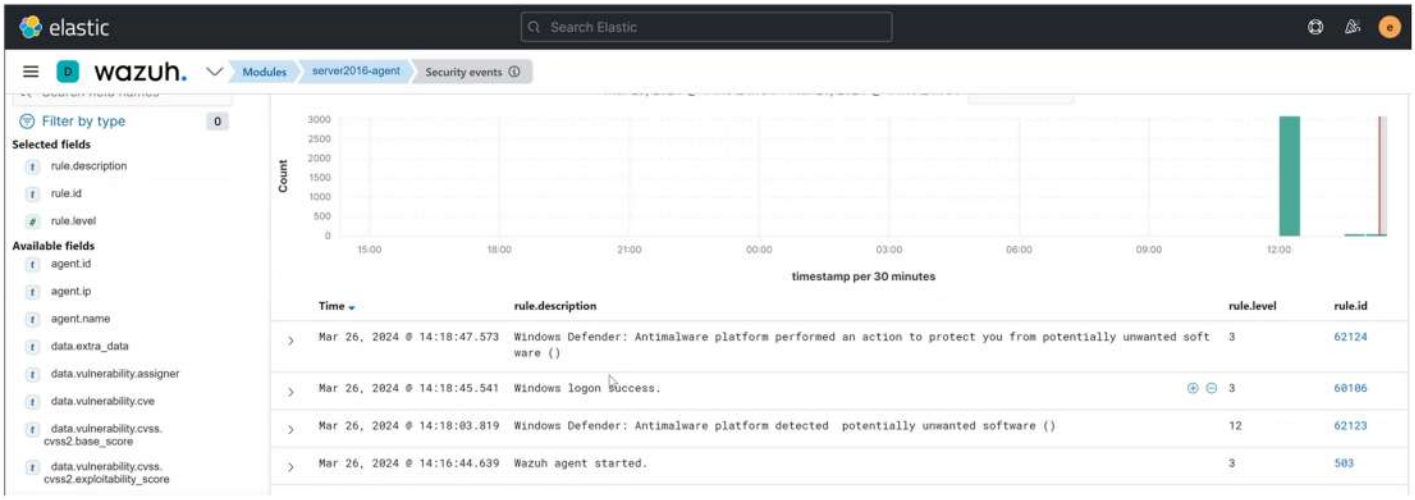
- let's now trigger the same alert so if we go ahead and we'll just grab this guy again so we'll see windows malware has detected it



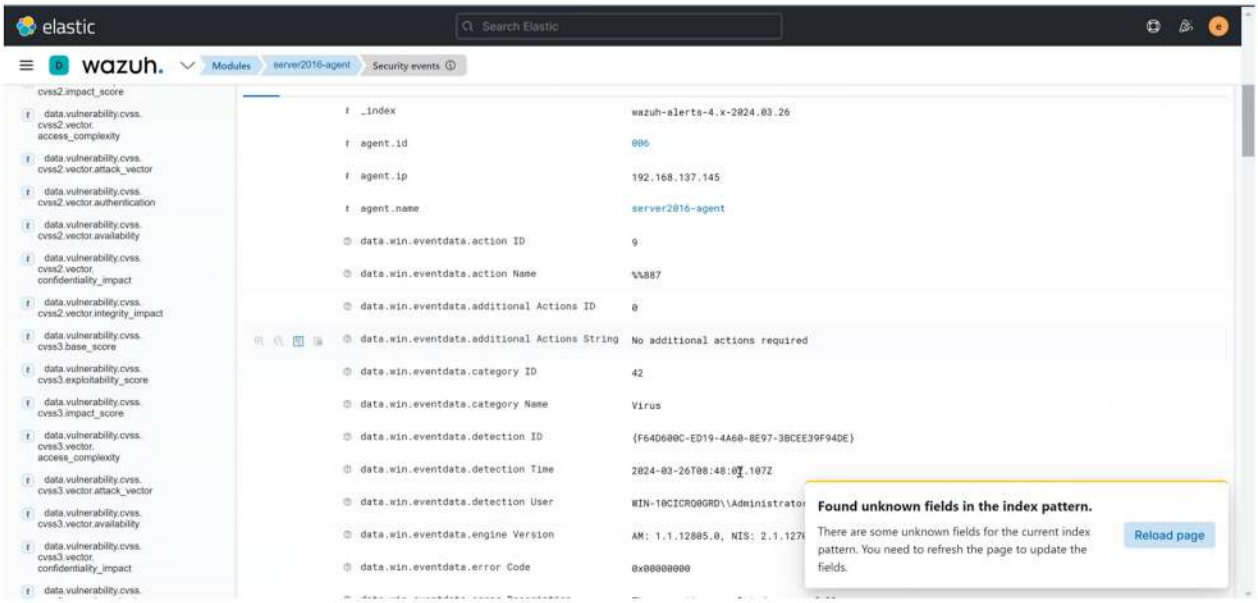


- if we go into our event viewer here and refresh we see our new entry here so that looks good so it's detected our log file

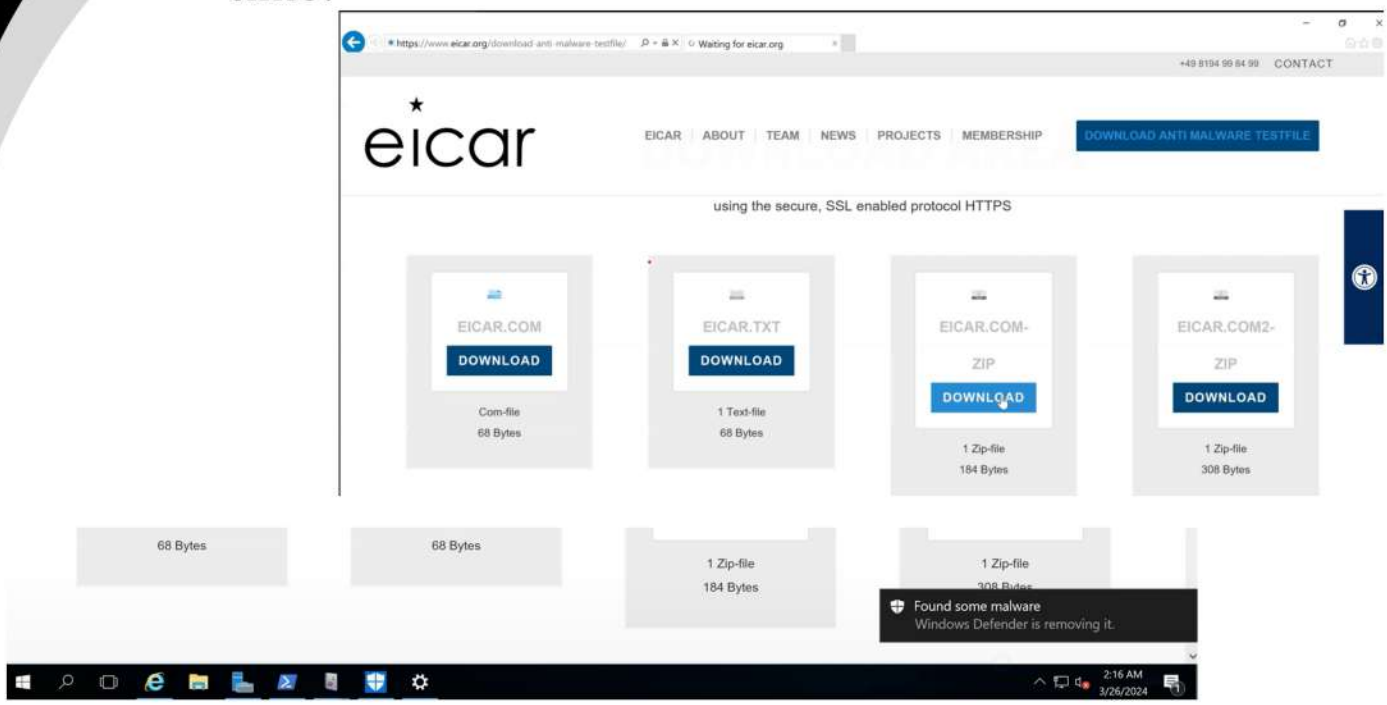
- so we first get our output windows defender has detected a potentially unwanted software so meaning malware and then we can see it performed an action to protect us and that is windows defender actually quarantining the file for us.



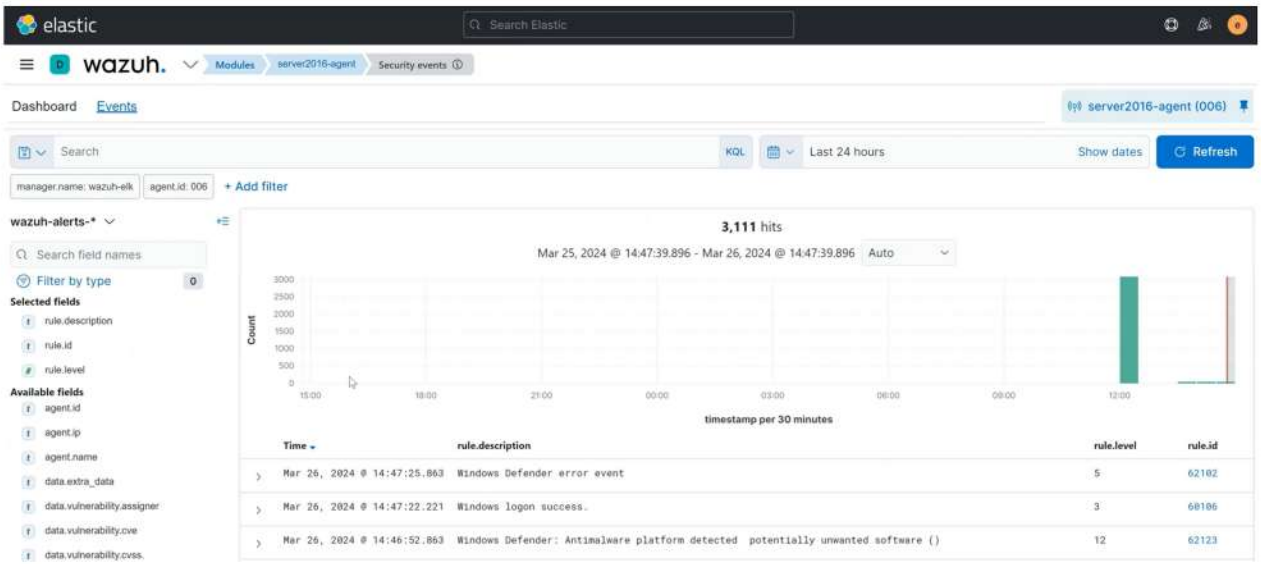
- and if we expand this out we get all the metadata around that particular file that windows defender recognized as being malicious so here you can see it this icarcom tried to file tried to be written into our downloads directory and we see all the metadata around it which is really nice.



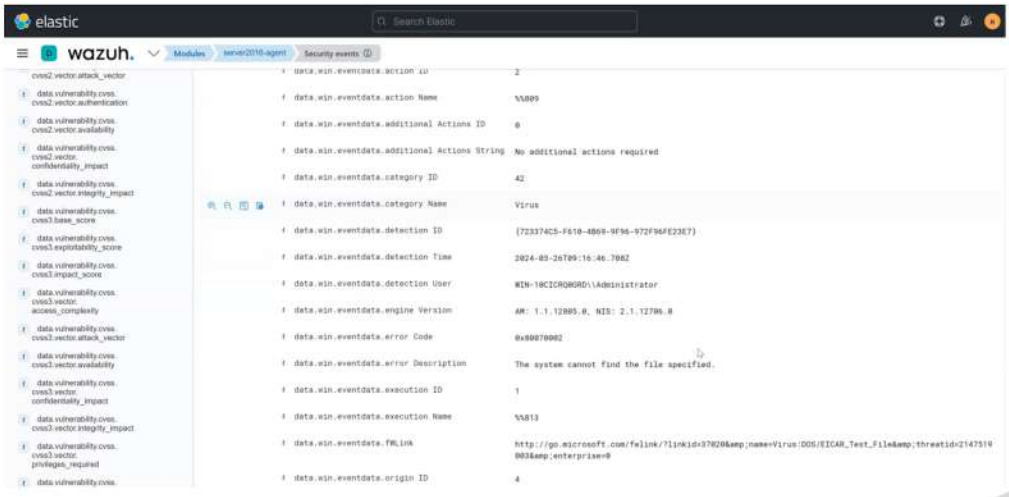
- Let's take another example and just download zip file this time.



- Let's refresh this page and check if alerts trigger or not



- And if we go into details this is what we download virus and where it is located.



Chapter 7
Integrity Monitoring and Assurance

7.1 Definition and importance of integrity monitoring in security

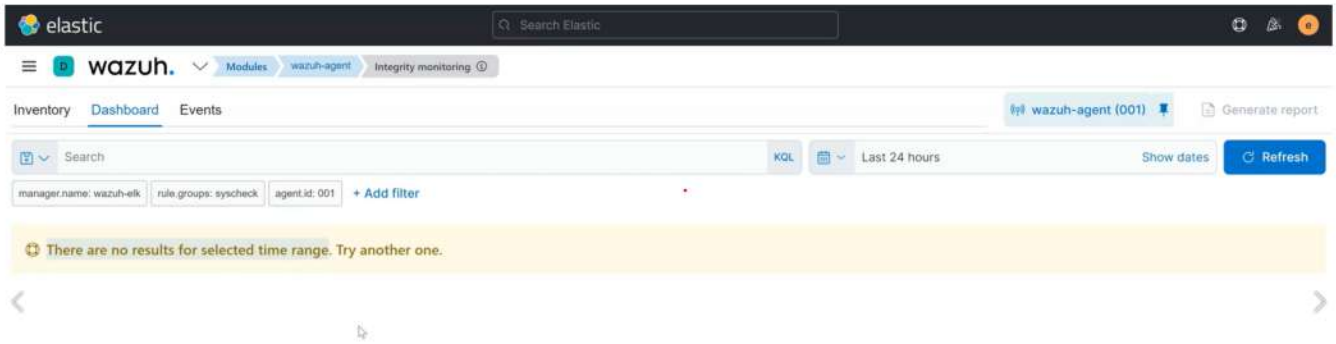
- Integrity monitoring in security, particularly within Wazuh, involves the continuous monitoring of critical files, directories, and system configurations to detect any unauthorized changes or tampering.
- It ensures the integrity and security of the system by alerting administrators to any deviations from the expected state.
- This is crucial for maintaining a secure environment as it helps prevent unauthorized access, data breaches, and the exploitation of vulnerabilities.

The importance of integrity monitoring lies in its ability to

- Identify Unauthorized Changes: Integrity monitoring detects alterations made to files, directories, and configurations that may indicate malicious activity or system compromise.
- Prevent Data Breaches: By promptly alerting administrators to unauthorized changes, integrity monitoring helps prevent data breaches and data loss.
- Enhance Incident Response: Early detection of unauthorized changes allows for swift incident response, minimizing the impact of security incidents.

7.2 Explanation of how Wazuh facilitates integrity monitoring for file systems

- Open the wazuh manager dashboard and go to the Integrity Monitoring. It Is by default disabled you have to make it enabled



- Let's configure some files to enable this file integrity tool just go into the command line of wazuh manager and open the /var/ossec/etc/ossec.conf file to configure and find the file integrity monitoring module and make sure it is enabled

```
<!-- File integrity monitoring -->
<syscheck>
  <disabled>no</disabled>

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Generate alert when new file detected -->
  <alert_new_files>yes</alert_new_files>

  <!-- Don't ignore files that change more than 'frequency' times -->
  <auto_ignore frequency="10" timeframe="3600">no</auto_ignore>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories>/etc,/usr/bin,/usr/sbin</directories>
  <directories>/bin,/sbin,/boot</directories>
```

- We need to edit directory's so wazuh manager can gather data in real time basis so we have to edit like this as shown in the below picture.

```
<!-- File integrity monitoring -->
<syscheck>
  <disabled>no</disabled>

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Generate alert when new file detected -->
  <alert_new_files>yes</alert_new_files>

  <!-- Don't ignore files that change more than 'frequency' times -->
  <auto_ignore frequency="10" timeframe="3600">no</auto_ignore>

  <!-- Directories to check  (perform all possible verifications) -->
  <directories check_all="yes" whodata="yes">/etc,/usr/bin,/usr/sbin</directories>
  <directories check_all="yes" whodata="yes">/bin,/sbin,/boot</directories>

  <!-- Files/directories to ignore -->
  <ignore>/etc/mtab</ignore>
  <ignore>/etc/hosts.deny</ignore>
  <ignore>/etc/mail/statistics</ignore>
```

- We did configuration in our wazuh manager's config file so firstly we have to do restart the wazuh manager service.

```
booksimp385@wazuh-elk:~$ sudo vi /var/ossec/etc/ossec.conf
booksimp385@wazuh-elk:~$ sudo systemctl restart wazuh-manager.service
booksimp385@wazuh-elk:~$
booksimp385@wazuh-elk:~$ sudo systemctl status wazuh-manager.service
● wazuh-manager.service - Wazuh manager
     Loaded: loaded (/lib/systemd/system/wazuh-manager.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-03-10 07:19:21 UTC; 22s ago
    Process: 50355 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 119 (limit: 9506)
     Memory: 321.4M
        CPU: 35.329s
     CGroup: /system.slice/wazuh-manager.service
             ├─50412 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─50413 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─50416 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─50419 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─50463 /var/ossec/bin/wazuh-authd
             ├─50479 /var/ossec/bin/wazuh-db
             ├─50503 /var/ossec/bin/wazuh-execd
```

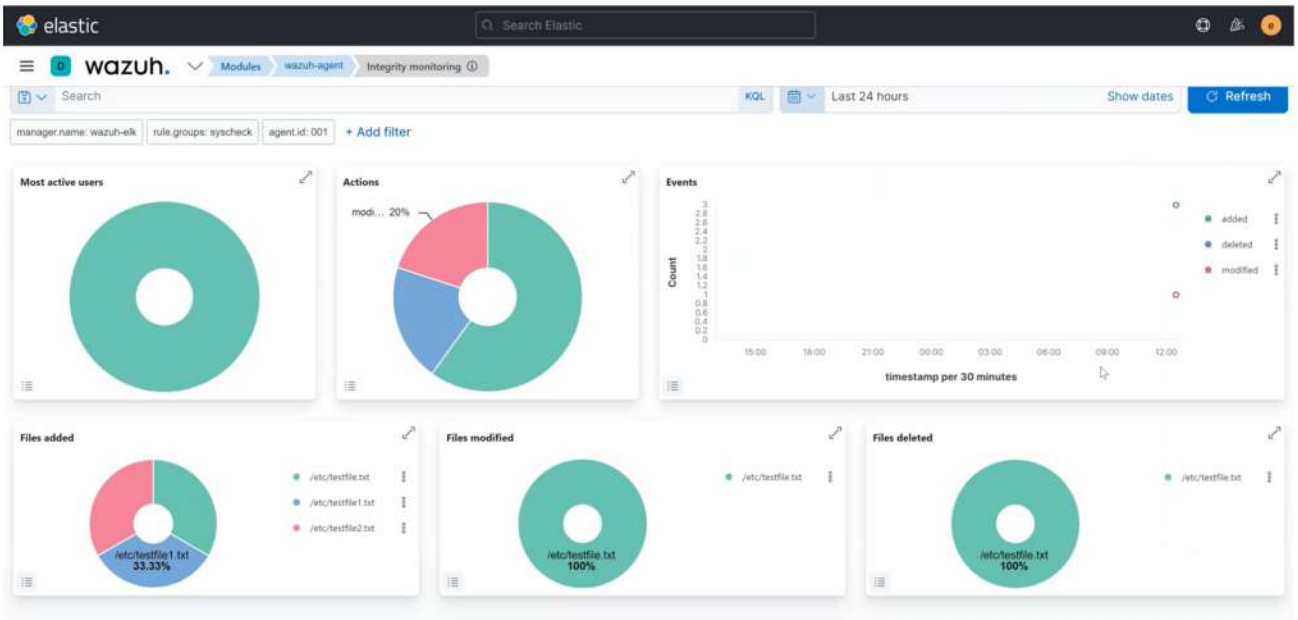
- The same file is also available in the agent side you have to also configure this file as same as we did for the wazuh manager. Configure that file and also restart the wazuh agent service.

```
root@wazuh-agent:/etc# systemctl restart wazuh-agent.service
root@wazuh-agent:/etc# systemctl status wazuh-agent.service
● wazuh-agent.service - Wazuh agent
     Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-03-10 07:27:05 UTC; 9s ago
    Process: 7219 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 33 (limit: 9506)
     Memory: 16.2M
        CPU: 13.142s
     CGroup: /system.slice/wazuh-agent.service
             ├─7242 /var/ossec/bin/wazuh-execd
             ├─7253 /var/ossec/bin/wazuh-agentd
             ├─7267 /var/ossec/bin/wazuh-syscheckd
             ├─7282 /var/ossec/bin/wazuh-logcollector
             └─7298 /var/ossec/bin/wazuh-modulesd
```

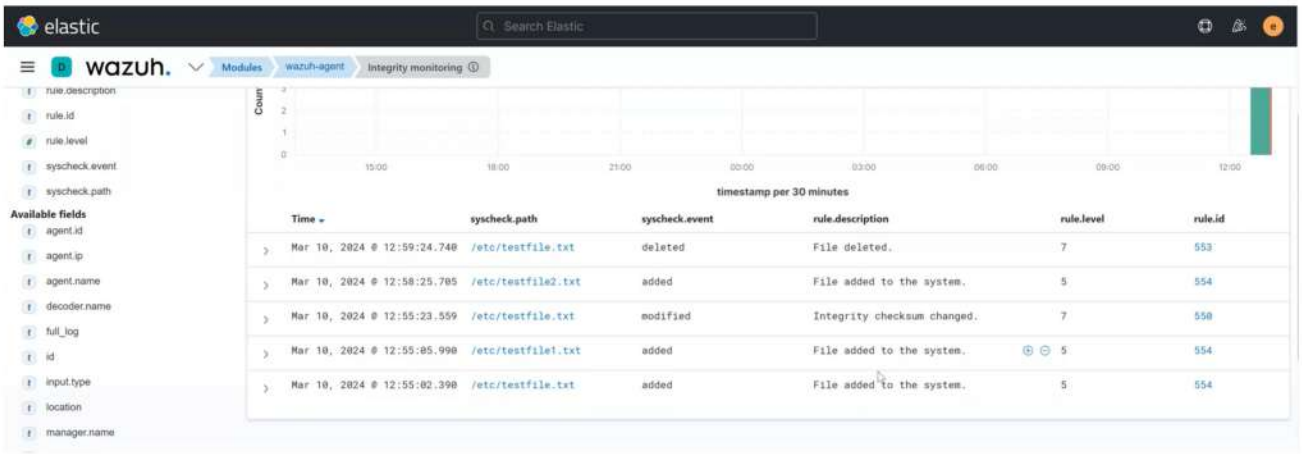
- Let's do some work so that we can trigger some alerts or notifications in our wazuh manager. we are doing like creating some empty files then modifying it and also deleting so let's hope for the best.

```
booksimp385@wazuh-agent:~$ cd /etc
booksimp385@wazuh-agent:/etc$
booksimp385@wazuh-agent:/etc$ sudo touch testfile2.txt
booksimp385@wazuh-agent:/etc$ vi testfile2.txt
booksimp385@wazuh-agent:/etc$
booksimp385@wazuh-agent:/etc$ vi testfile2.txt
booksimp385@wazuh-agent:/etc$
booksimp385@wazuh-agent:/etc$ rm -r testfile.txt
rm: remove write-protected regular file 'testfile.txt'? y
rm: cannot remove 'testfile.txt': Permission denied
booksimp385@wazuh-agent:/etc$
booksimp385@wazuh-agent:/etc$ sudo rm -r testfile.txt
booksimp385@wazuh-agent:/etc$
booksimp385@wazuh-agent:/etc$
```

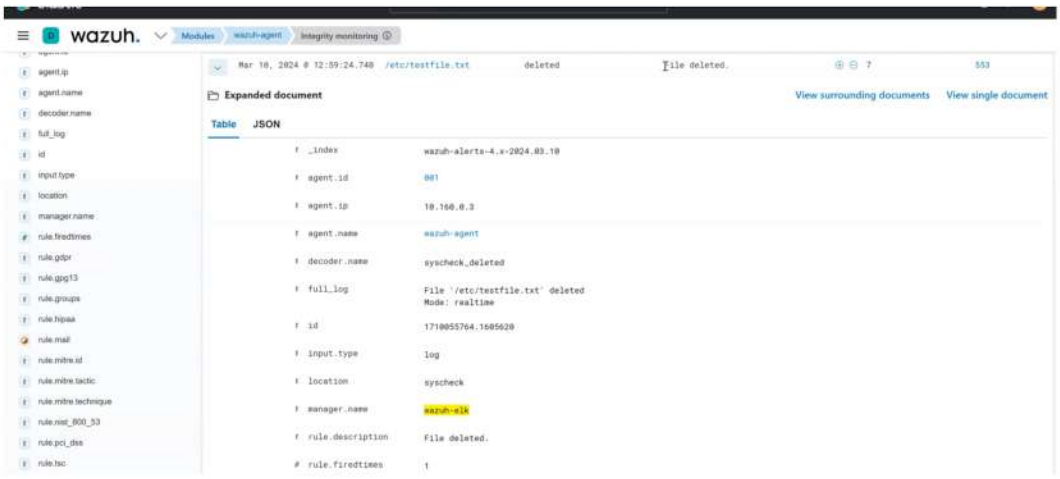
- Let’s check it in the wazuh manager if it is triggered or not.



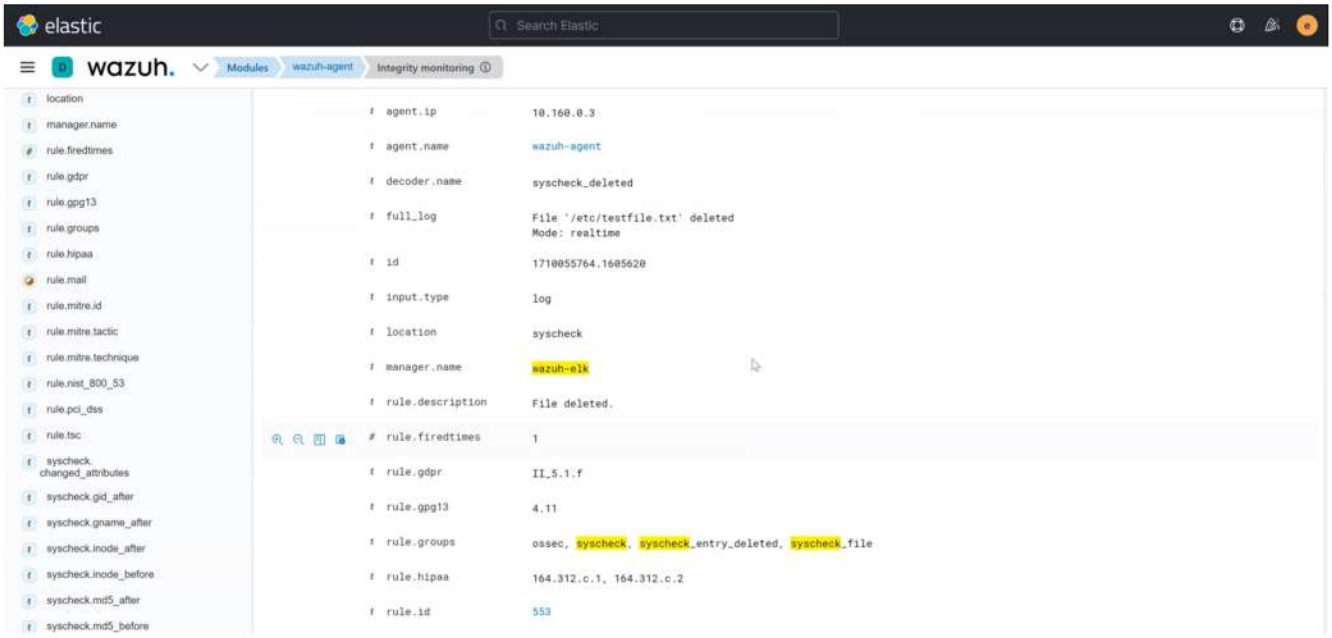
- Soo we can see that there are also three options which we have did right now we added the file modified the file and deleted the file.



- We can check here which file in the real time basis modified, deleted or added to the system.



- We can also explore that triggered event just click on that and scroll down then you can see full details and also logs like from where this file is created and all.



- By following these professional steps to configure file integrity monitoring in Wazuh, organizations can effectively protect their data from theft and unauthorized access.
- FIM provides continuous monitoring of critical files and directories, enabling quick detection and response to any unauthorized changes, thereby enhancing overall data security.

Chapter 8
Comprehensive System Auditing

8.1 Importance of system auditing for compliance and security purposes

8.1.1 Risk Management

- System auditing helps organizations identify and mitigate potential security risks and vulnerabilities.
- By monitoring system activities and configurations, auditing can detect unauthorized access attempts, suspicious behavior, and deviations from established security policies.

8.1.2 Detection of Security Incidents

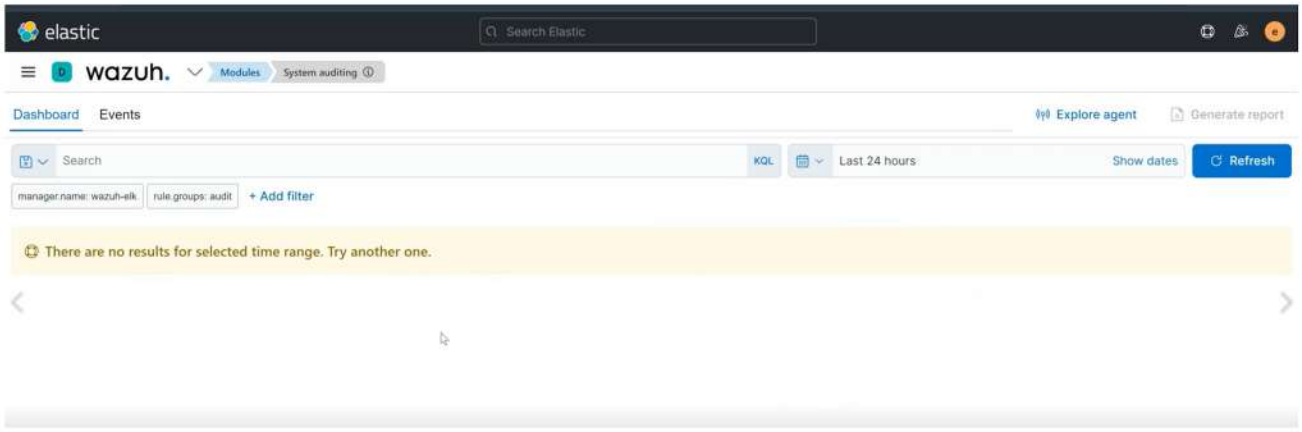
- Auditing provides visibility into system events and activities, allowing organizations to detect security incidents in real-time or through retrospective analysis.
- Suspicious activities, such as unauthorized access, malware infections, or data breaches, can be identified and investigated promptly.

8.1.3 Forensic Analysis:

- Auditing logs can be analyzed to reconstruct the timeline of events, identify the root cause of the incident, and support incident response and remediation efforts.

8.2 how Wazuh aids in system auditing and log management

- As we see in the pervious chapter the FIM Module is not enabled or not configured by default same as here the System auditing module also not enabled we have to configure it manually.



- How we can configure it, Go to the wazuh agent, Firstly we have to install the auditd service on the agent side.

```
booksimp385@wazuh-agent:~$ sudo systemctl status auditd.service
Unit auditd.service could not be found.
booksimp385@wazuh-agent:~$
booksimp385@wazuh-agent:~$  sudo apt-get install -y auditd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libauparse0
Suggested packages:
  audispd-plugins
The following NEW packages will be installed:
  auditd libauparse0
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 270 kB of archives.
After this operation, 876 kB of additional disk space will be used.
Get:1 http://asia-south1.gce.archive.ubuntu.com/ubuntu jammy/main amd64 libauparse0 amd64 1:3.0.7-1build1 [58.0 kB]
Get:2 http://asia-south1.gce.archive.ubuntu.com/ubuntu jammy/main amd64 auditd amd64 1:3.0.7-1build1 [212 kB]
27% [2 auditd 0 B/212 kB 0%]
```

- After installing successfully this service kindly enable and start this service and also don't forget to check the status of the service.

```
booksimp385@wazuh-agent:~$ sudo systemctl start auditd
booksimp385@wazuh-agent:~$ sudo systemctl enable auditd
Synchronizing state of auditd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable auditd
booksimp385@wazuh-agent:~$
booksimp385@wazuh-agent:~$ sudo systemctl status auditd
● auditd.service - Security Auditing Service
     Loaded: loaded (/lib/systemd/system/auditd.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-03-10 12:07:37 UTC; 1min 58s ago
       Docs: man:auditd(8)
             https://github.com/linux-audit/audit-documentation
   Main PID: 2043 (auditd)
      Tasks: 2 (limit: 9506)
     Memory: 504.0K
        CPU: 53ms
     CGroup: /system.slice/auditd.service
             └─2043 /sbin/auditd
```

- Now we have to configure ossec.conf file in the agent to manually enable the system auditing.
- Open that file and scroll down to the end and add one log which is shown in the picture given below.

```
  <localfile>
    <log_format>audit</log_format>
    <location>/var/log/audit/audit.log</location>
  </localfile>

</ossec_config>
-- INSERT --
```

- It should be added to the end of the file. save & exit. and also restart the agent service with

[sudo systemctl restart wazuh-agent]

- Our next step is to check the user i'd, For which user we have to do system audit for that we are gathering user's id.

```
booksimp385@wazuh-agent:~$ sudo systemctl restart wazuh-agent.service
booksimp385@wazuh-agent:~$
booksimp385@wazuh-agent:~$ echo $EUID
1001
booksimp385@wazuh-agent:~$
booksimp385@wazuh-agent:~$
```

- So there is a rule file for auditd service and we have to configure it with adding some rules to gather real time auditing.
- Open the file with **[sudo vi /etc/audit/rules.d/audit.rule]**
- You can see there is already pre-configured file like as shown below in the picture.

```
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 60000

## Set failure mode to syslog
-f 1
```

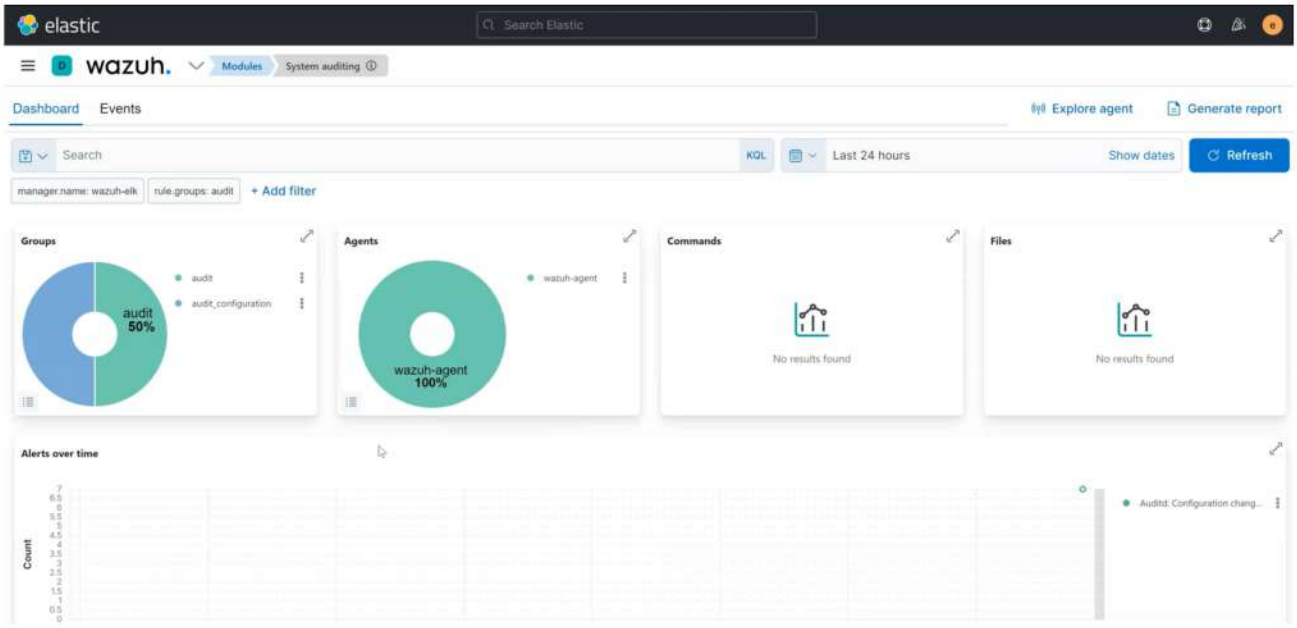
- We have to edit this last two rules in that file kindly replace your user id with yours.

```
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 0

## Set failure mode to syslog
-f 1
-a exit,always -F euid=1001 -F arch=b32 -S execve -k audit-wazuh-c
-a exit,always -F euid=1001 -F arch=b64 -S execve -k audit-wazuh-c
~
```

- This command wil help you to enable this rulewhich have define.

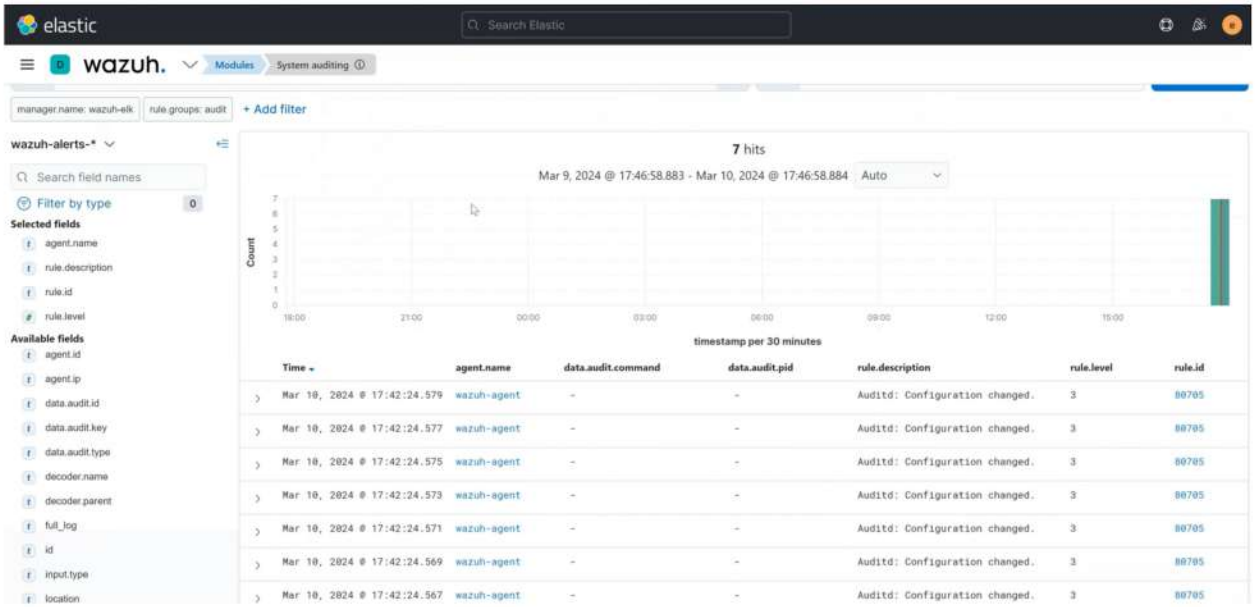
```
booksimp385@wazuh-agent:~$ sudo vi /etc/audit/rules.d/audit.rules
booksimp385@wazuh-agent:~$
booksimp385@wazuh-agent:~$ sudo auditctl -R /etc/audit/rules.d/wazuh.rules
```

- Now go back to the wazuh dashboard and open the system auditing module you can see as shown below picture.

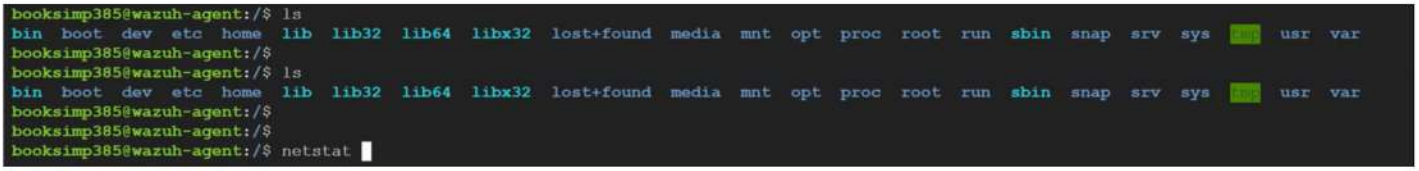


- You can see auditing service is enabled and lastly we modified the audit configuration file and also you can see there is only one agent which is 100% using.

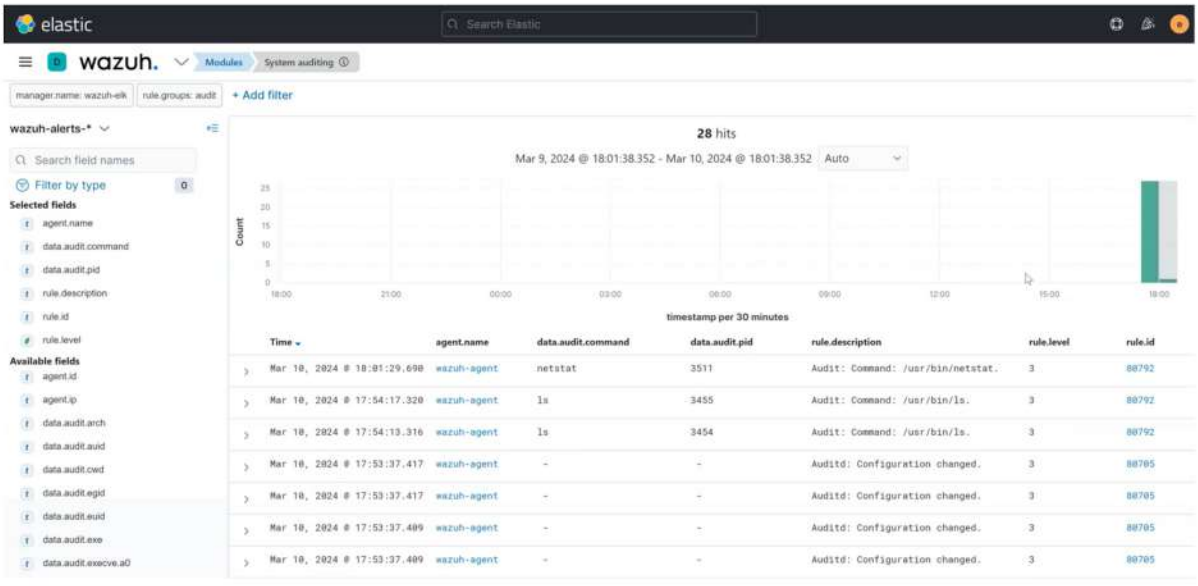
- You can also check the events section.
- there is also events are triggered which we have did with the configuration file.



- Let's run some basic commands and check if we can trigger some events.

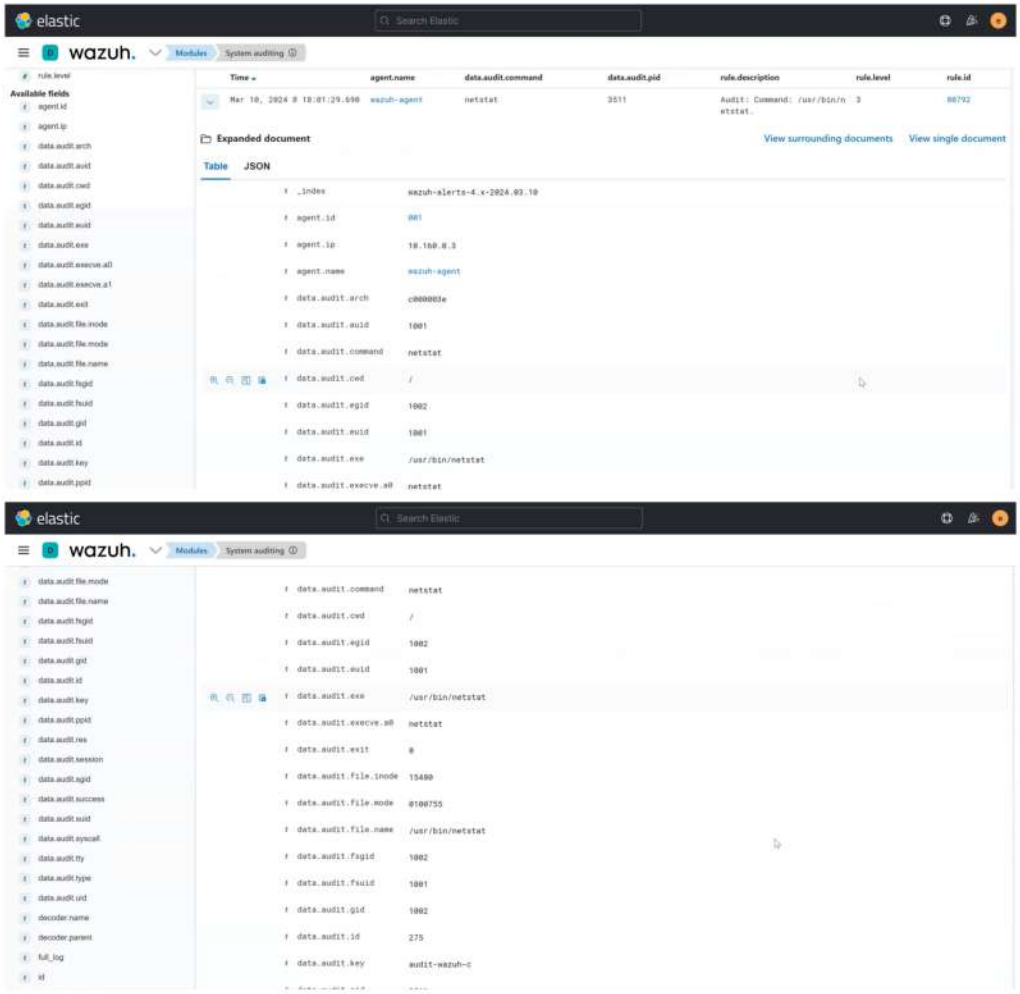


- And here you can see the events are triggered of the commands which we have run on the wazuh agent.



- First three events is which we have triggered two time **LS** command and one time **NETSTAT**.

- You can also exapand this event and check like which type of information showing so also we can check here which agent is running this command which user is running this command from where this command gets trigger and many more.



- In summary, system auditing is indispensable for ensuring compliance with regulations, managing security risks, detecting and responding to security incidents, supporting forensic investigations, enhancing accountability, and demonstrating legal and regulatory compliance.
- It is a fundamental component of effective cybersecurity and governance practices within organizations.

Chapter 9
Enhanced Email Security

9.1 Explanation of the benefits of integrating Wazuh with Outlook for email security

9.1.1 Real-time Threat Detection

- By integrating Wazuh with Outlook, organizations can enhance their email security posture by monitoring incoming and outgoing emails in real-time.
- Wazuh can analyze email headers, attachments, and content for signs of malicious activity, such as phishing attempts, malware, or suspicious links.

9.1.2 Immediate Alerting and Response

- Wazuh can generate immediate alerts upon detecting suspicious emails, allowing security teams to respond promptly to potential threats.
- Alerts can be sent to administrators or analysts via email, providing instant notification of security incidents.

9.1.3 Attachment Analysis

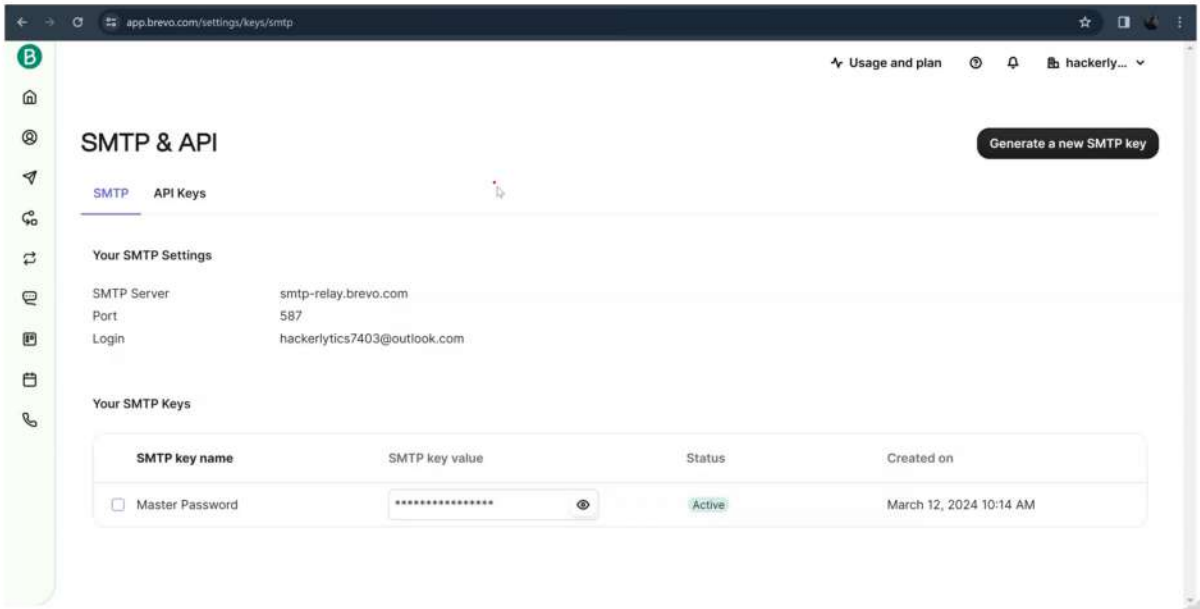
- Wazuh can analyze email attachments for known malware signatures or behavior patterns indicative of malicious intent.
- It can detect and quarantine malicious attachments before they reach end-users, preventing malware infections and data breaches.

9.1.4 Customized Policies and Rules

- Administrators can configure rules based on specific security requirements or threat intelligence feeds, tailoring email security controls to the organization's unique needs.

9.2 Step-by-step guide on integrating Wazuh with Outlook for threat detection in emails

- we're going to look at configuring our wazuh manager to send us email alerts when rules fire over a particular level threshold.
- so what this will allow us to do is to be able to have a email inbox that will alert us when a particular rule severity fires.
- right so now we don't have to just be monitoring our kibana dashboards all the time you know we can set that aside and continue on with some other work and have a email inbox set up to where if a particular rule fires that we want to be made aware about we'll get an email alert in our inbox
- and then that will then direct us of course to going into kibana and gathering some of the metadata on the alert.
- we can use to set up a smtp service and specifically tell our wazoo manager you know what smtp server we want to take advantage of this could be you know an internal smtp server that you have set up within your network or you could take advantage of a cloud-based smtp server.
- so let's go ahead and look into doing that so i first have i'm taking advantage of this send in blue they're just a cloud smtp um solution so i guess free plug for these guys



- so we're going to go ahead and take advantage of post fix to configure these credentials
- grab the postfix software and download that and once that's done we'll move on to the rest of the config all right

```
root@wazuh-elk:~# apt-get update && apt-get install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules
Hit:1 https://artifacts.elastic.co/packages/7.x/apt stable InRelease
Hit:2 https://packages.wazuh.com/4.x/apt stable InRelease
Hit:3 http://asia-south1.gce.archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://asia-south1.gce.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:6 http://asia-south1.gce.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 119 kB in 1s (123 kB/s)
Reading package lists... 67%
```

- Append these lines to /etc/postfix/main.cf to configure Postfix. Create the file if missing.
- relay host will be our send in blue smtp server

```
relayhost = smtp-relay.brevo.com:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
smtp_use_tls = yes
smtpd_relay_restrictions = permit_mynetworks, permit_sasl_authenticated, defer_unauth_destination
~
```

- Set the sender email address and password.
- Replace <USERNAME> and <PASSWORD> with your own data.

```
root@wazuh-elk:~# echo smtp-relay.brevo.com:587 hackerlytics7403@outlook.com:ZsYpvmFPSLBOK7bE > /etc/postfix/sasl_passwd
root@wazuh-elk:~#
root@wazuh-elk:~# cat /etc/postfix/sasl_passwd
smtp-relay.brevo.com:587 hackerlytics7403@outlook.com:ZsYpvmFPSLBOK7bE
root@wazuh-elk:~#
```

- Let's change the permissions of files.

```
root@wazuh-elk:~# postmap /etc/postfix/sasl_passwd
postmap: warning: /etc/postfix/main.cf, line 49: overriding earlier entry: relayhost=
postmap: warning: /etc/postfix/main.cf, line 55: overriding earlier entry: smtpd_relay_restrictions=permit_mynetworks permit_sasl_authenticated defer_unauth_destination
root@wazuh-elk:~#
root@wazuh-elk:~# chmod 400 /etc/postfix/sasl_passwd
root@wazuh-elk:~#
```

- Let's add some security & Secure your password DB file

```
root@wazuh-elk:~# chown root:root /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
root@wazuh-elk:~# chmod 0600 /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
root@wazuh-elk:~#
root@wazuh-elk:~#
```

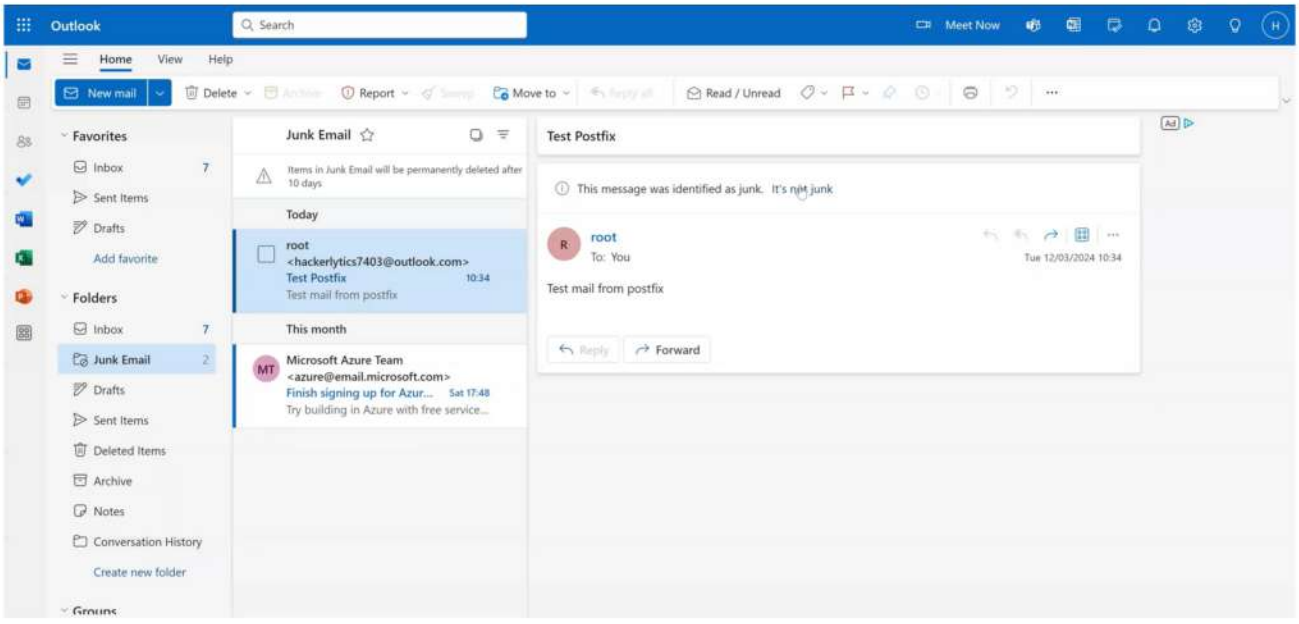
- i will go ahead and restart postfix so let's see if that service is up and running let's go ahead and start this guy okay

```
root@wazuh-elk:~# systemctl restart postfix
root@wazuh-elk:~#
root@wazuh-elk:~# systemctl status  postfix
● postfix.service - Postfix Mail Transport Agent
     Loaded: loaded (/lib/systemd/system/postfix.service; enabled; vendor preset: enabled)
     Active: active (exited) since Tue 2024-03-12 05:01:37 UTC; 24s ago
       Docs: man:postfix(1)
    Process: 7023 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 7023 (code=exited, status=0/SUCCESS)
        CPU: 2ms

Mar 12 05:01:37 wazuh-elk systemd[1]: Starting Postfix Mail Transport Agent...
Mar 12 05:01:37 wazuh-elk systemd[1]: Finished Postfix Mail Transport Agent.
root@wazuh-elk:~#
root@wazuh-elk:~#
```

- Run the following command to test the configuration. Replace you@example.com with your email address. Check, then, that you receive this test email

```
root@wazuh-elk:~# echo "Test mail from postfix" | mail -s "Test Postfix" -r "hackerlytics7403@outlook.com" hackerlytics7403@outlook.com
root@wazuh-elk:~#
```

- As you can see we received the test email.



- Everything is working correctly so for now let's go and tell wazuh manager to send emails. Configure email notifications in the Wazuh server /var/ossec/etc/ossec.conf file as follows.

- Open the ossec config file and find the gloable configuration and edit as shown in picture below. (replace emails and smtp server with yours).

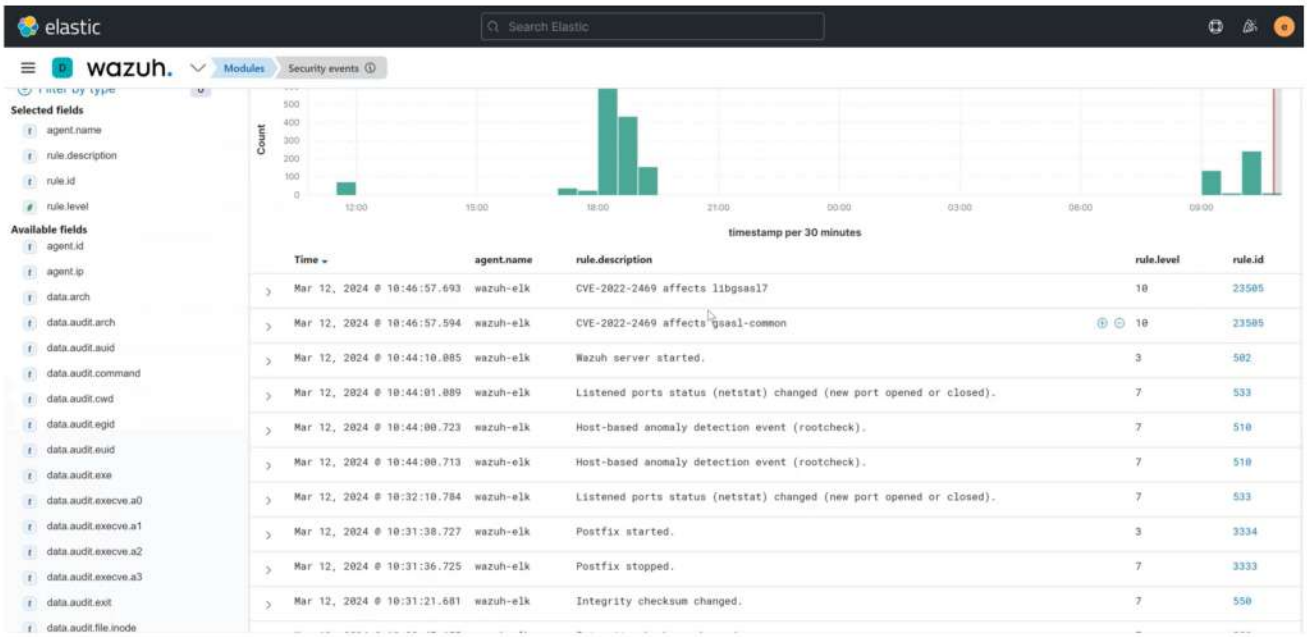
```
<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>no</logall>
    <logall_json>no</logall_json>
    <email_notification>yes</email_notification>
    <smtp_server>localhost</smtp_server>
    <email_from>hackerlytics7403@outlook.com</email_from>
    <email_to>hackerlytics7403@outlook.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
  </global>

  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>3</email_alert_level>
  </alerts>
```

- Let’s go ahead and restart the wazuh manager and see the status.

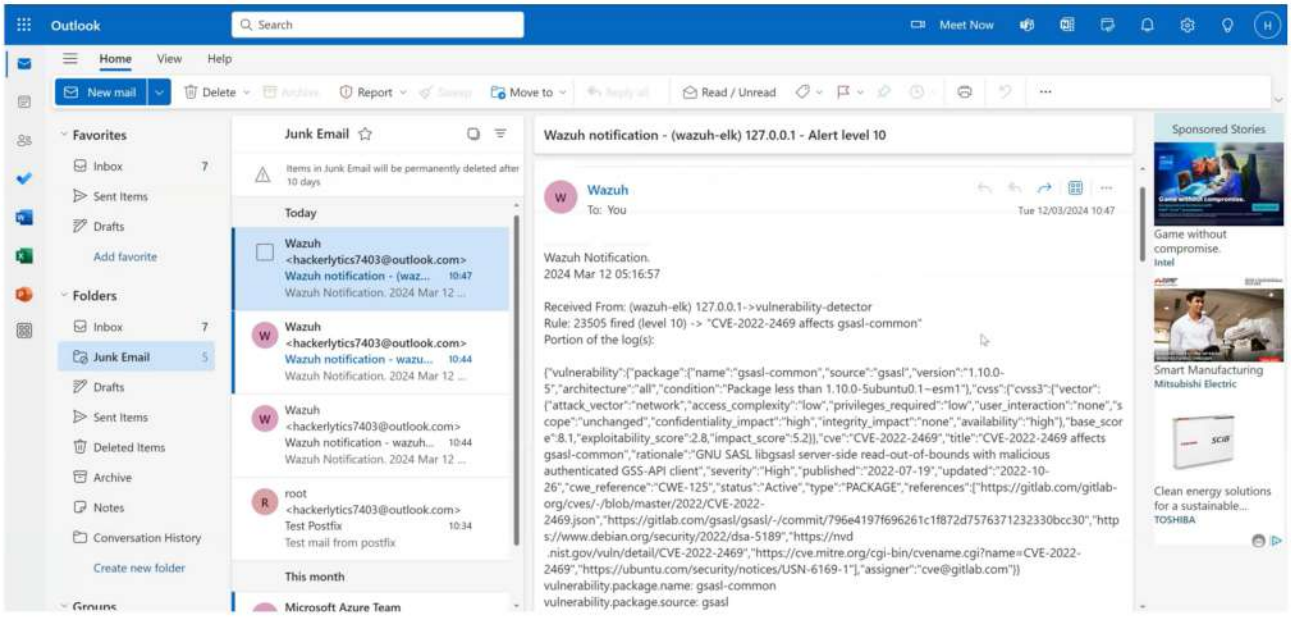
```
root@wazuh-elk:~# vi /var/ossec/etc/ossec.conf
root@wazuh-elk:~# systemctl restart wazuh-manager.service
root@wazuh-elk:~#
root@wazuh-elk:~# systemctl status wazuh-manager.service
● wazuh-manager.service - Wazuh manager
     Loaded: loaded (/lib/systemd/system/wazuh-manager.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2024-03-12 05:14:04 UTC; 8s ago
    Process: 7297 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
      Tasks: 124 (limit: 9506)
     Memory: 637.3M
        CPU: 46.039s
     CGroup: /system.slice/wazuh-manager.service
             ├─7355 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─7356 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─7358 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─7362 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
             ├─7386 /var/ossec/bin/wazuh-integratord
             ├─7407 /var/ossec/bin/wazuh-authd
             ├─7425 /var/ossec/bin/wazuh-db
```

- Now let‘s go to the wazuh dashboard and in the security events and wait for the new events get‘s triggered.

- You can see there is en event which is our vulnerability detector at the rule level of 10 (First event)



- That the same event you can also fine in the your mailbox which is outlook for us. Here you can see in the below given picture.



- By following these steps, you can efficiently manage and monitor alerts through your Outlook mailbox, enhancing your ability to respond promptly to any incidents or important notifications.

Chapter 10
Real-Time Alerting

10.1 Overview of integrating Wazuh with Telegram for real-time alerts

10.1.1 Instant Notifications

- Wazuh can send alerts directly to Telegram, ensuring that security incidents are promptly communicated to relevant personnel.

10.1.2 Centralized Alert Management

- Telegram integration allows for centralized alert management, ensuring that all critical alerts are consolidated in one platform for easy monitoring.

10.1.3 Remote Accessibility

- Telegram alerts can be accessed from any device with the Telegram app installed, providing flexibility for security teams to receive alerts even when they are not in the office.

10.1.4 Customizable Alerts

- Alerts sent to Telegram can be customized based on severity levels or specific types of security events, allowing security teams to prioritize their response accordingly.

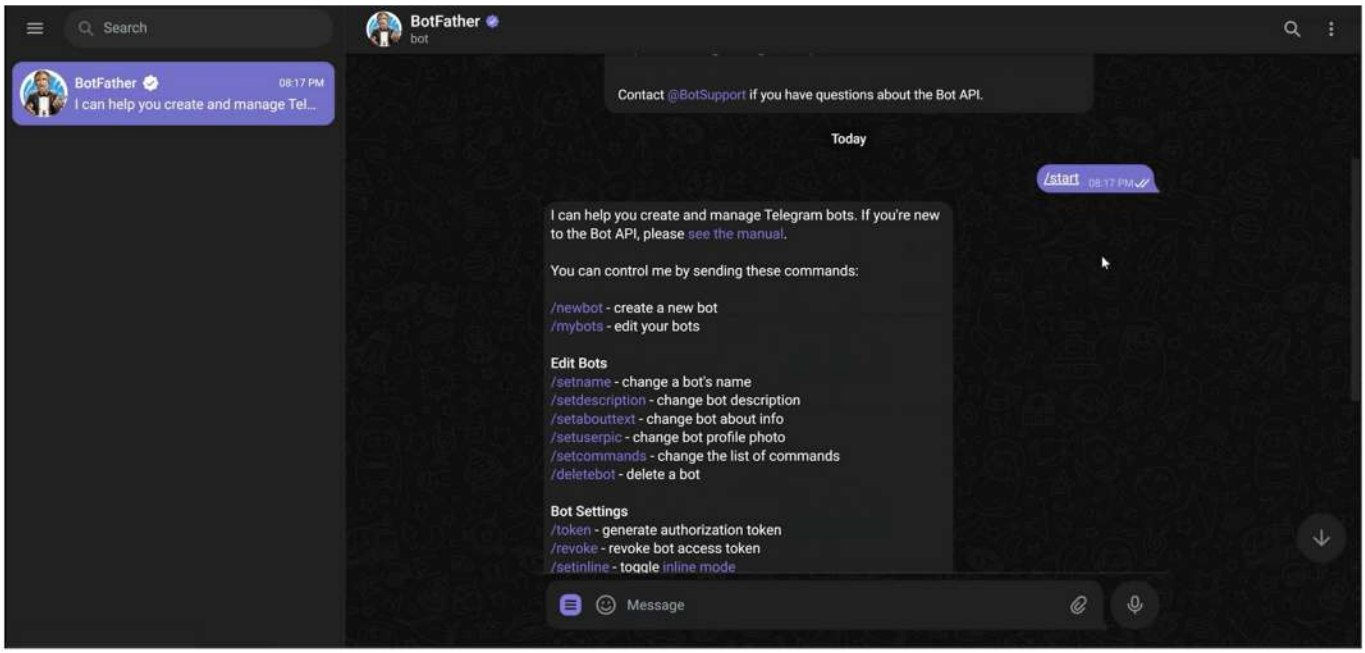
10.1.5 Cost-Effective Solution

- Telegram is a free messaging platform, making it a cost-effective solution for receiving real-time alerts compared to other commercial solutions.

10.2 Demonstration of setting up Telegram integration with Wazuh

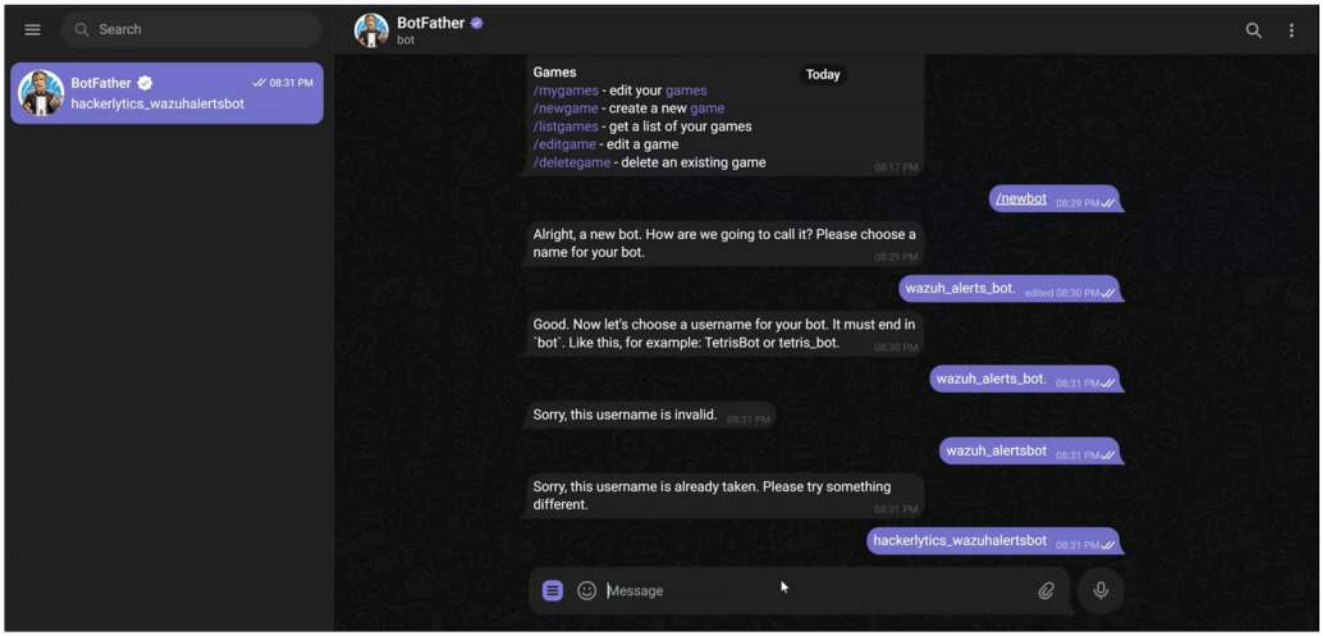
- i want to show you guys how we can integrate wazoo with telegram which is a instant messaging platform and is a nice little integration to make to be able to send alerts telegram group in real time as they come through.

- so you first need to create a telegram user account um but after you've done that you will get your little chat window here and what we're first going to do is start off with a call to the bot father because we'll need to create a new bot that will post our wazoo alerts to our telegram group.
- and to do that telegram has a tool called bot father which is a pretty hilarious name actually that we can that we can interact with to create our new bot and so that's first thing.

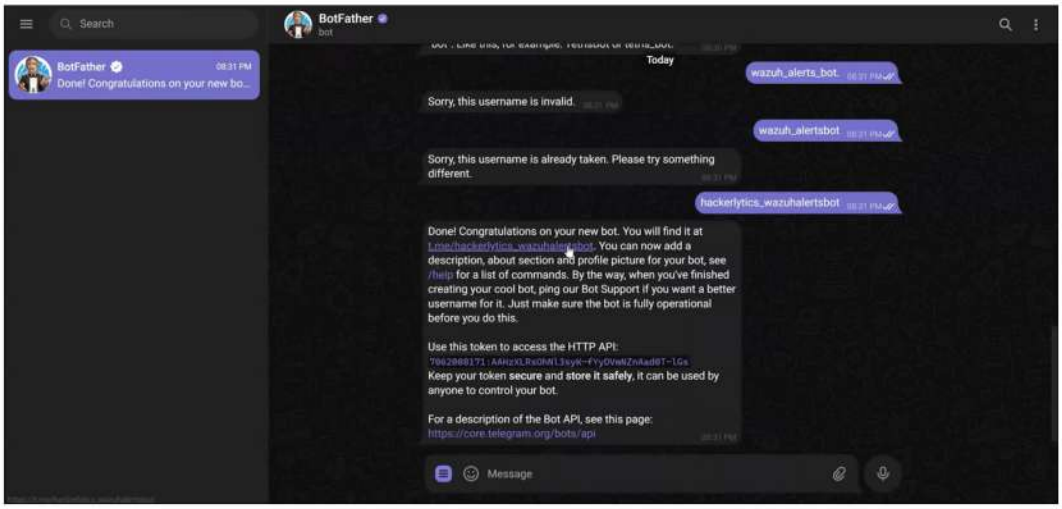


- we're first going to say new bot so we're telling the bot father hey we wanna stand up a new bot and he's gonna ask you okay what's the name you want and i'll say open secure wazoo.

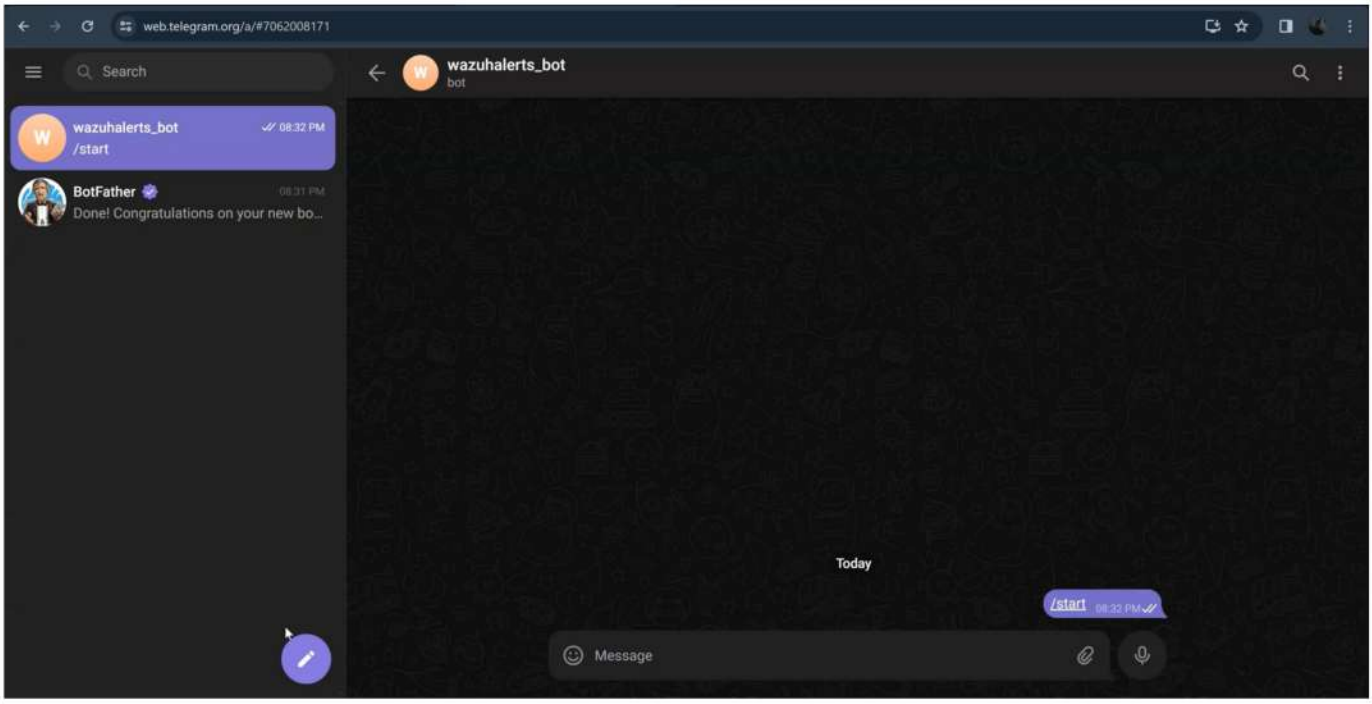
- so now we've created our open secure wazoo bot and now we need a username for the bot uh you do see here that it has to end with the with the word bot so just make sure whatever you come up with that it ends with the bot so here i'll just say open secure wazoo bot



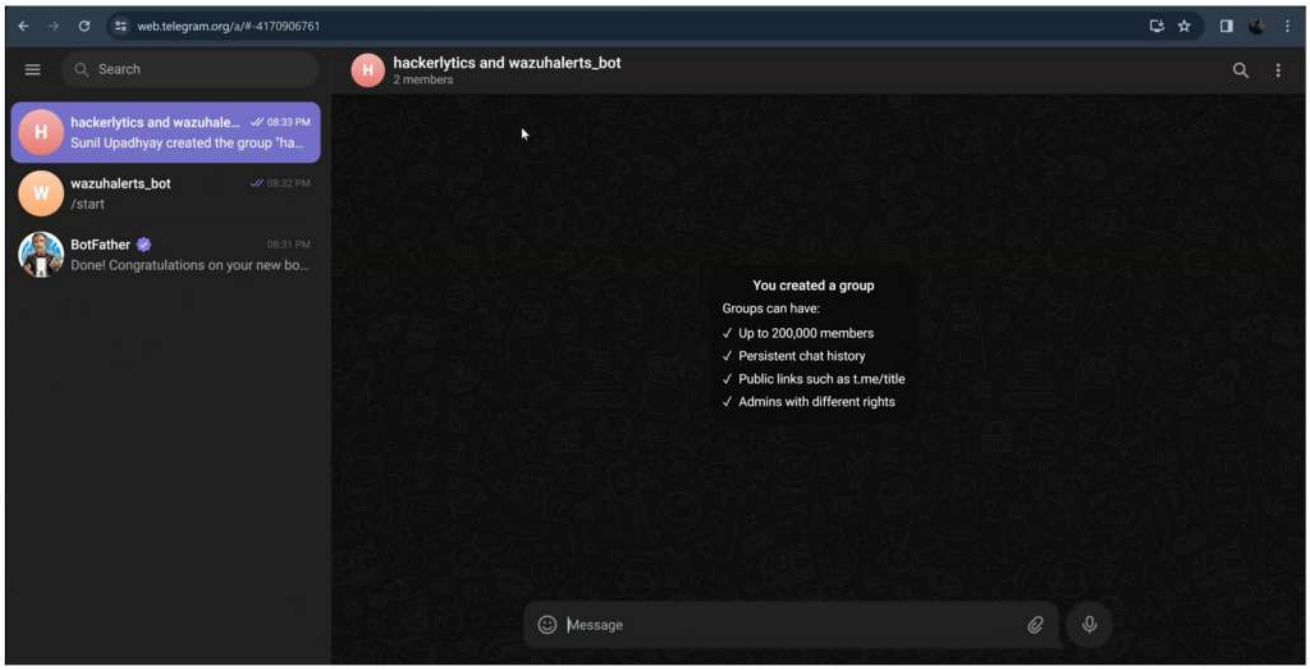
- you notice that we now get our api key which we use to authenticate with our bots so we need to make sure that we copy this value down and store it somewhere because we will now include these within our wazoo scripts



- now let's actually create a group for this bot so we'll select our new bot that's linked here and we need to start this guy so we'll go ahead and start the bot

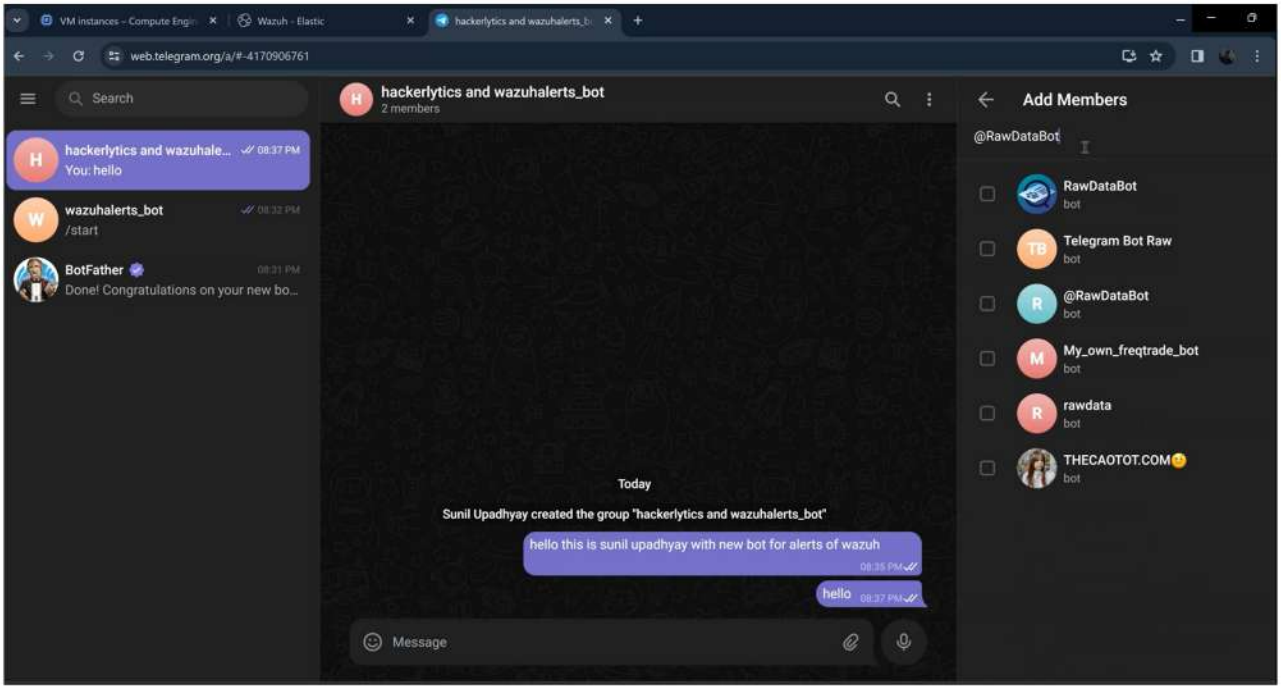


- all right now that we have that created a group and added a member which is our newly created bot.

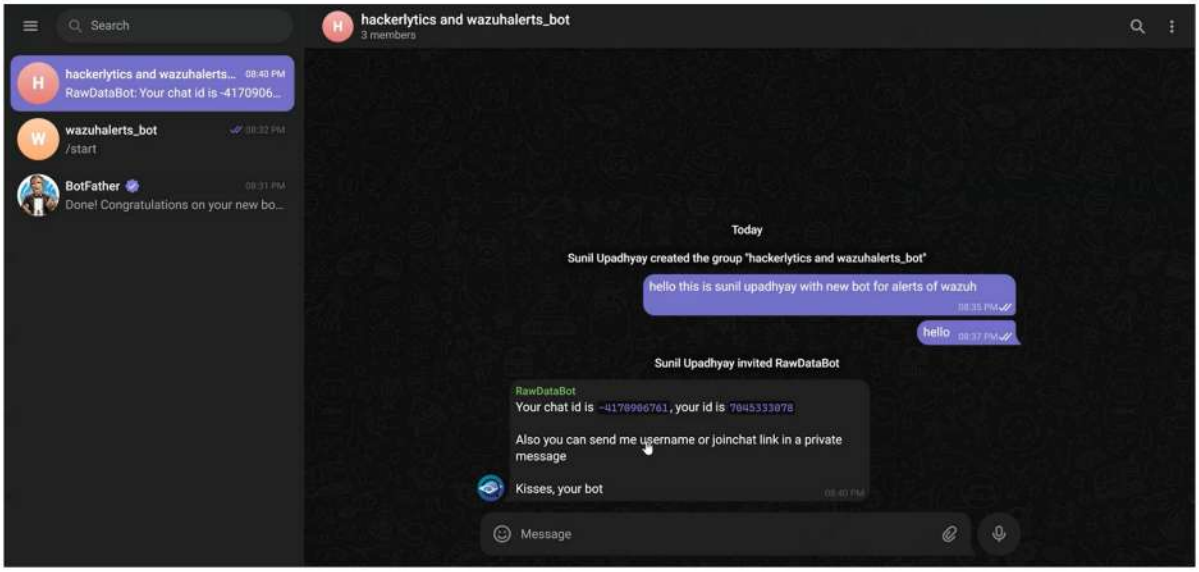


- let's actually first do a we need to put a message into this group so uh this message can be whatever whatever you want there just needs to be

- So let's add one bot to the our newly created group to get the chat i'd.



- So once you add this bot to group it will show you automatically your chat i'd.



- You can also find a git hub repo which i used for integrating telegram with wazuh.

- https://github.com/OpenSecureCo/Demos/blob/main/Telegram%20Integration

- so to do so we just need to jump onto our wazuh manager
- we first need to create our python wrapper which i've just named it custom telegram here you can of course name it whatever you want and copy paste first block from the git hub repo which i have linked in last page.

```
#!/bin/sh

WPYTHON_BIN="framework/python/bin/python3"

SCRIPT_PATH_NAME="$0"

DIR_NAME="$(cd $(dirname ${SCRIPT_PATH_NAME}); pwd -P)"
SCRIPT_NAME="$(basename ${SCRIPT_PATH_NAME})"

case ${DIR_NAME} in
    */active-response/bin | */wodles*)
        if [ -z "${WAZUH_PATH}" ]; then
            WAZUH_PATH="$(cd ${DIR_NAME}/../..; pwd)"
        fi

        PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
    ;;
    */bin)
        if [ -z "${WAZUH_PATH}" ]; then
            WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
        fi

        PYTHON_SCRIPT="${WAZUH_PATH}/framework/scripts/${SCRIPT_NAME}.py"
    ;;
     */integrations)
        if [ -z "${WAZUH_PATH}" ]; then
            WAZUH_PATH="$(cd ${DIR_NAME}/..; pwd)"
        fi

        PYTHON_SCRIPT="${DIR_NAME}/${SCRIPT_NAME}.py"
    ;;
esac


${WAZUH_PATH}/${WPYTHON_BIN} ${PYTHON_SCRIPT} "$@"
```

- we'll now create the actual python script so we'll create our customtelegram.py and then we'll copy the python content down and paste and make some changes to it.
- here you see we do need to make a change here for our chat id so whatever value that we just grabbed via this tool here we need to take this value and post it

```
#!/usr/bin/env python

import sys
import json
import requests
from requests.auth import HTTPBasicAuth

#CHAT_ID="xxxx"
CHAT_ID="-4170906761"

# Read configuration parameters
alert_file = open(sys.argv[1])
hook_url = sys.argv[3]


# Read the alert file
alert_json = json.loads(alert_file.read())
alert_file.close()

# Extract data fields
alert_level = alert_json['rule']['level'] if 'level' in alert_json['rule'] else "N/A"
description = alert_json['rule']['description'] if 'description' in alert_json['rule'] else "N/A"
agent = alert_json['agent']['name'] if 'name' in alert_json['agent'] else "N/A"
# Generate request
msg_data = {}
msg_data['chat_id'] = CHAT_ID
msg_data['text'] = {}
msg_data['text']['description'] =  description
msg_data['text']['alert_level'] = str(alert_level)
msg_data['text']['agent'] =  agent
headers = {'content-type': 'application/json', 'Accept-Charset': 'UTF-8'}


# Send the request
requests.post(hook_url, headers=headers, data=json.dumps(msg_data))

sys.exit(0)
```

- we need to change the permissions of these scripts here so first we'll change the permission to the root owner and wazuh group and then we need to modify their execution permissions.

```
root@wazuh-elk:~# chown root:wazuh /var/ossec/integrations/custom-telegram*
root@wazuh-elk:~#
root@wazuh-elk:~# chmod 750 /var/ossec/integrations/custom-telegram*
root@wazuh-elk:~#
root@wazuh-elk:~# ls -llh /var/ossec/integrations
total 88K
-rwxr-xr-x 1 root wazuh 2.7K Mar 11 12:22 custom-remove-threat
-rwxr-x--- 1 root wazuh  846 Mar 24 15:13 custom-telegram
-rwxr-x--- 1 root wazuh  995 Mar 24 15:16 custom-telegram.py
-rwxr-x--- 1 root wazuh 1.1K Feb 29 13:05 maltiverse
-rwxr-x--- 1 root wazuh  17K Feb 29 13:05 maltiverse.py
-rwxr-x--- 1 root wazuh 1.1K Feb 29 13:05 pagerduty
-rwxr-x--- 1 root wazuh 7.0K Feb 29 13:05 pagerduty.py
-rwxr-x--- 1 root wazuh 1.1K Feb 29 13:05 shuffle
-rwxr-x--- 1 root wazuh 7.6K Feb 29 13:05 shuffle.py
-rwxr-x--- 1 root wazuh 1.1K Feb 29 13:05 slack
-rwxr-x--- 1 root wazuh 7.2K Feb 29 13:05 slack.py
-rwxr-x--- 1 root wazuh 1.1K Feb 29 13:05 virustotal
-rwxr-x--- 1 root wazuh 9.6K Feb 29 13:05 virustotal.py
root@wazuh-elk:~#
```

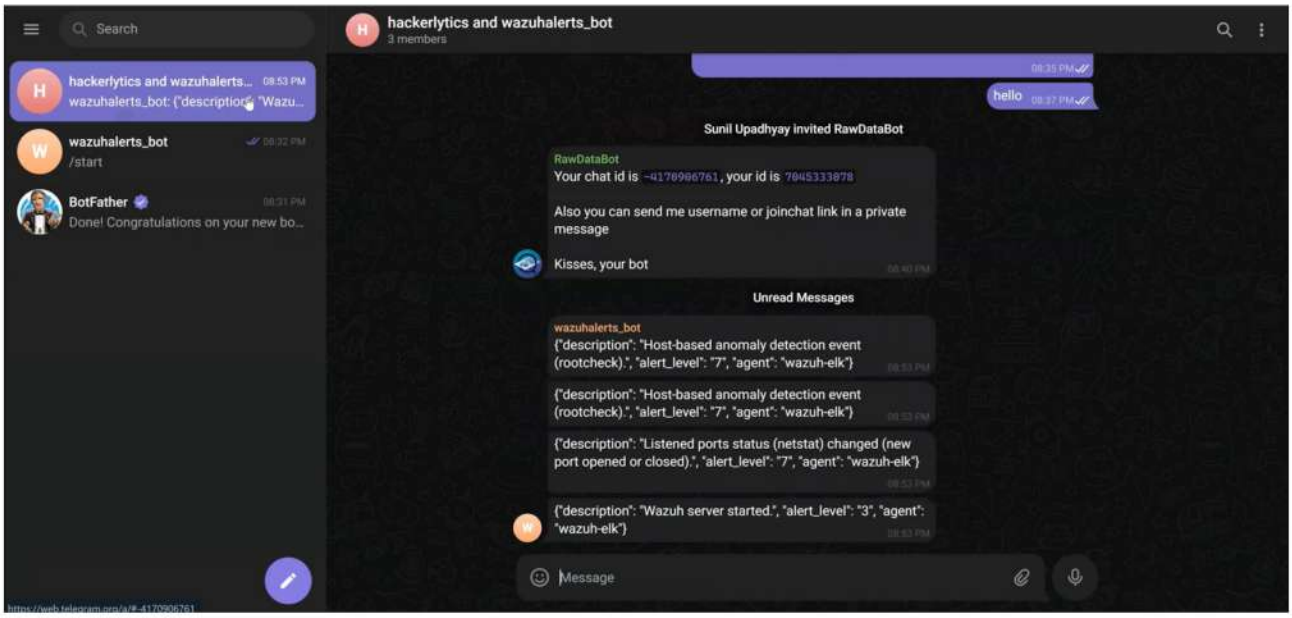
- let's now pop into our ossec.conf here and we need to now add our integration block so i will copy this block here and scroll down to where i have my integrations and let's add our integration block

```
<integration>
  <name>custom-telegram</name>
  <level>3</level>
  <hook_url>https://api.telegram.org/bot7062008171:AAHzXLRsOhNl3syK-fYyDVmNZnAad0T-lGs/sendMessage</hook_url>
  <alert_format>json</alert_format>
</integration>

<!-- Osquery integration -->
```

- Make sure you change your api key with yours.
- Now restart the wazuh manager and check the status .

```
root@wazuh-elk:~# systemctl restart wazuh-manager.service
root@wazuh-elk:~#
root@wazuh-elk:~#
root@wazuh-elk:~#
```

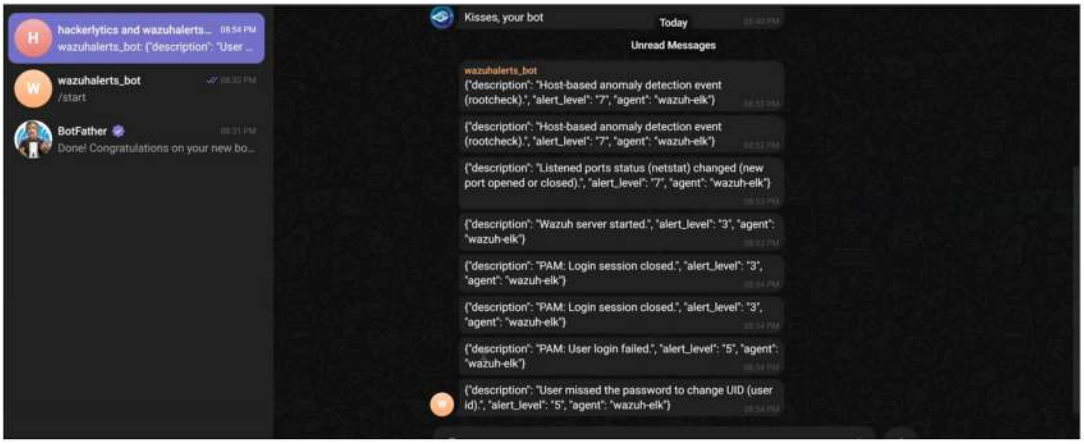
- Let’s trigger some events or wait for few minutes to get some updates on telegram.
- Here you can see now we get an updates and real time events we can see here.



- You can see there is last event which is wazuh server started this events is for just now we restarted the wazuh server.
- We are triggering en event for writing wrong password or authentication failure.

```
booksimp385@wazuh-elk:~$
booksimp385@wazuh-elk:~$ su -
Password:
su: Authentication failure
booksimp385@wazuh-elk:~$
```

- And here you can see we get events of user logout session closed and also user login failed.
- That's how we can integrate telegram with wazuh



Chapter 11
Automated Health Monitoring

11.1 Importance of monitoring Wazuh's health and performance

11.1.1 Early Detection of Issues

- Monitoring allows for early detection of potential issues or failures within the Wazuh environment.
- This includes issues with system resources, services, or configuration errors.

11.1.2 Proactive Maintenance

- By monitoring health and performance metrics, administrators can proactively address potential problems before they escalate into critical issues.
- This helps in maintaining system reliability and uptime.

11.1.3 Capacity Planning

- Monitoring helps in capacity planning by providing insights into resource usage trends over time.
- It allows administrators to anticipate future resource requirements and scale the infrastructure accordingly.

11.1.4 Security Effectiveness

- A healthy and well-performing Wazuh environment ensures the effectiveness of security monitoring and incident response activities.

NO MANAGER = NO ALERTS

Multiple Processes Make up the Wazuh Manager Service

- wazuh-integratord
- wazuh-authd
- wazuh=db
- wazuh-execd
- wazuh-analysid
- wazuh-syscheckd
- wazuh-remoted
- wazuh-logcollector
- wazuh-monitord
- wazuh-modulesd

```
[root@opensearch opt]# ps -aux | grep wazuh
ossec     4061 11.0  2.1 432548 84024 ?        Sl   10:40   0:04 /var/ossec/framework/python/bin/py
ossecm    4084  0.0  0.0  34780  1556 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-integratord
root      4105  0.0  0.0 124936  3324 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-authd
ossec     4119  0.2  0.4 640632 17616 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-db
root      4144  0.0  0.0  34704  1536 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-execd
ossec     4156  0.8  0.5 970440 21688 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-analysisd
root      4166  5.1  0.1 135384  5016 ?        SNl  10:40   0:01 /var/ossec/bin/wazuh-syscheckd
ossecr    4188  0.1  0.0 716088  3596 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-remoted
root      4221  0.2  0.0 411712  2100 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-logcollector
ossec     4231  0.0  0.0  34796  1536 ?        Sl   10:40   0:00 /var/ossec/bin/wazuh-monitord
root      4252  1.2  0.4 1341920 17940 ?       Sl   10:40   0:00 /var/ossec/bin/wazuh-modulesd
```

- no manager equals no alerts so we really need to make sure our managers are always in a healthy state or else we're going to miss out on potential high severity alerts within our environment
- so as part of you guys are probably familiar now with the Wazuh manager that runs as a service however that service is not just one standalone process there are multiple processes that make up the Wazuh manager service as a whole and these are responsible for doing various tasks
- what I want to do is just create a simple bash script that we're going to run on our Wazuh manager on a set interval of seconds so our bash script will do a p grab we'll take advantage of the pgrep command
- so we're going to create a bash script to detect running Wazuh processes we're then going to attempt to have the script to resolve the issue by itself
- we're then going to write the output to our temp health.json file we're going to collect the file with the Wazuh manager

- I am on my Wazuh manager here um and if I do a ps aux and let's grep for Wazuh here you can see all of our running processes.

```
root@wazuh-elk:~# ps -aux | grep wazuh
wazuh       1667  1.3  1.5 841572 123376 ?       Sl   03:07   0:16 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       1668  0.0  0.8 177608 71756 ?        S    03:07   0:00 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       1670  0.4  0.8 251400 72188 ?        S    03:07   0:05 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       1673  0.0  0.7 386952 62624 ?        S    03:07   0:00 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       1699  0.0  0.0  24848  4300 ?        Sl   03:07   0:00 /var/ossec/bin/wazuh-integratord
root        1719  0.2  0.0 180632  6976 ?        Sl   03:07   0:03 /var/ossec/bin/wazuh-authd
wazuh       1735  0.1  0.2 770620 16652 ?        Sl   03:07   0:01 /var/ossec/bin/wazuh-db
root        1759  0.0  0.0  24904  3536 ?        Sl   03:07   0:00 /var/ossec/bin/wazuh-execd
wazuh       1774  0.0  0.0  24980  3992 ?        Sl   03:07   0:00 /var/ossec/bin/wazuh-maild
wazuh       1780  0.2  0.3 1278124 31772 ?       Sl   03:07   0:02 /var/ossec/bin/wazuh-analysisd
root        1805  1.4  0.1 409272 12080 ?        SNl  03:07   0:18 /var/ossec/bin/wazuh-syscheckd
wazuh       1821  0.0  0.0  16748  3832 ?        S    03:07   0:00 /var/ossec/bin/wazuh-remoted
wazuh       1822  0.2  0.1 845252  8768 ?        Sl   03:07   0:03 /var/ossec/bin/wazuh-remoted
root        1855  0.0  0.0 467316  4944 ?        Sl   03:07   0:00 /var/ossec/bin/wazuh-logcollector
wazuh       1877  0.0  0.0  24884  4104 ?        Sl   03:07   0:00 /var/ossec/bin/wazuh-monitord
root        1901 14.6  0.3 491644 28596 ?        Sl   03:07   3:01 /var/ossec/bin/wazuh-modulesd
root        4735  0.0  0.0   7008  2304 pts/1    R+   03:28   0:00 grep --color=auto wazuh
root@wazuh-elk:~#
```

- So now just make a bash file and open it and paste content that is shown below.

```
#!/bin/sh
host=$(ip route get 8.8.8.8 | awk -F"src " 'NR==1{split($2,a," ");print a[1]}')

#Check wazuh-Authd processes
if [ -n "$(pgrep wazuh-authd)" ]; then
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-authd", "healthy":"yes"}'
echo -e "$json" >> /tmp/health.json
else
#Attempt a restart
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-authd", "healthy":"attempting_restart"}'
echo -e "$json" >> /tmp/health.json
service wazuh-manager restart
sleep 5
if [ -n "$(pgrep wazuh-authd)" ]; then
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-authd", "healthy":"yes"}'
echo -e "$json" >> /tmp/health.json
else
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-authd", "healthy":"no"}'
echo -e "$json" >> /tmp/health.json
fi
fi

#Check wazuh-db processes
if [ -n "$(pgrep wazuh-db)" ]; then
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-db", "healthy":"yes"}'
echo -e "$json" >> /tmp/health.json
else
#Attempt a restart
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-db", "healthy":"attempting_restart"}'
echo -e "$json" >> /tmp/health.json
service wazuh-manager restart
sleep 5
if [ -n "$(pgrep wazuh-db)" ]; then
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-db", "healthy":"yes"}'
echo -e "$json" >> /tmp/health.json
else
json='{"host":"'"$host"'", "wazuhprocess":"wazuh-db", "healthy":"no"}'
echo -e "$json" >> /tmp/health.json
fi
fi
```

- To understand properly this script you can refer this video https://youtu.be/qWzq90brTl4?si=cqKTGsZj7IQ-EMSb.

- Our next step is to make script executable to run this script out. and also don't forget to restart the wazuh manager service.

```
root@wazuh-elk:/opt# systemctl restart wazuh-manager.service
root@wazuh-elk:/opt#
root@wazuh-elk:/opt# chmod +x wazuh_healthcheck.sh
root@wazuh-elk:/opt#
```

- Let's remove first pervious content from health.json

```
root@wazuh-elk:/opt# echo > /tmp/health.json
root@wazuh-elk:/opt#
```

- So let's run this script and check the content of the health.json

```
root@wazuh-elk:/opt# ./wazuh_healthcheck.sh
root@wazuh-elk:/opt#
root@wazuh-elk:/opt# cat /tmp/health.json
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-authd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-db", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-execd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-analysisd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-syscheckd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-remoted", "healthy":"attempting_restart"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-remoted", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-logcollector", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-monitord", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-modulesd", "healthy":"yes"}
root@wazuh-elk:/opt#
root@wazuh-elk:/opt#
root@wazuh-elk:/opt#
```

- We should see everything is in health state and also you can see json string which we have added in our script
- we have our host in the output also we have the status of the processes and in the end current state of the process.

- Let's go ahead and kill one process and after that run the script and check the state of that process.

```
root@wazuh-elk:/opt# ps -aux | grep wazuh
wazuh       7182 14.2  1.2 750044 105388 ?       Sl   03:41   0:08 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       7183  0.0  0.7 165708 62416 ?        S    03:41   0:00 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       7186  0.0  0.7 247636 62696 ?        S    03:41   0:00 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       7189  0.0  0.7 395100 62620 ?        S    03:41   0:00 /var/ossec/framework/python/bin/python3 /var/ossec/api/scripts/wazuh-apid.py
wazuh       7214  0.0  0.0  24856  4308 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-integratord
root        7234  0.3  0.0 115092  6976 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-authd
wazuh       7251  0.5  0.1 770616 11588 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-db
root        7275  0.0  0.0  24900  3660 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-execd
wazuh       7289  0.0  0.0  24980  4232 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-maild
wazuh       7296  1.9  0.3 1278140 30108 ?       Sl   03:41   0:01 /var/ossec/bin/wazuh-analysisd
root        7309 29.2  0.1 278204 11572 ?        SNl  03:41   0:16 /var/ossec/bin/wazuh-syscheckd
wazuh       7328  0.0  0.0  16752  3836 ?        S    03:41   0:00 /var/ossec/bin/wazuh-remoted
wazuh       7329  0.2  0.0 714184  8004 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-remoted
root        7362  0.0  0.0 401780  4940 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-logcollector
wazuh       7407  0.0  0.0  24884  3592 ?        Sl   03:41   0:00 /var/ossec/bin/wazuh-monitord
root        7433 66.7  2.0 441980 170020 ?       Sl   03:41   0:34 /var/ossec/bin/wazuh-modulesd
root        9087  0.0  0.0   7008  2304 pts/1    S+   03:42   0:00 grep --color=auto wazuh
root@wazuh-elk:/opt#
root@wazuh-elk:/opt#
root@wazuh-elk:/opt#
root@wazuh-elk:/opt# kill -9 7234
root@wazuh-elk:/opt#
```

- Now let's run that script again and check the state of killed process.

```
root@wazuh-elk:/opt# ./wazuh_healthcheck.sh
root@wazuh-elk:/opt#
root@wazuh-elk:/opt#
root@wazuh-elk:/opt# cat /tmp/health.json

-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-authd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-db", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-execd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-analysisd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-syscheckd", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-remoted", "healthy":"attempting_restart"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-remoted", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-logcollector", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-monitord", "healthy":"yes"}
-e {"host":"10.160.0.2", "wazuhprocess":"wazuh-modulesd", "healthy":"yes"}
```

- Here you can see that process we killed shown as attempting to restart
- This approach ensures that our Wazuh managers are always running in a healthy state, maximizing the effectiveness of our security monitoring. Remember, no manager equals no alerts, so it's crucial to keep them healthy.

Chapter 12
Efficient Resource Utilization Monitoring

12.1 Explanation of how Wazuh monitors system resources automatically

12.1.1 Disk Space Monitoring

- Wazuh constantly monitors disk space usage across your systems. It checks for filesystems nearing capacity, which could potentially lead to system failures or data loss.

12.1.2 CPU Usage Monitoring

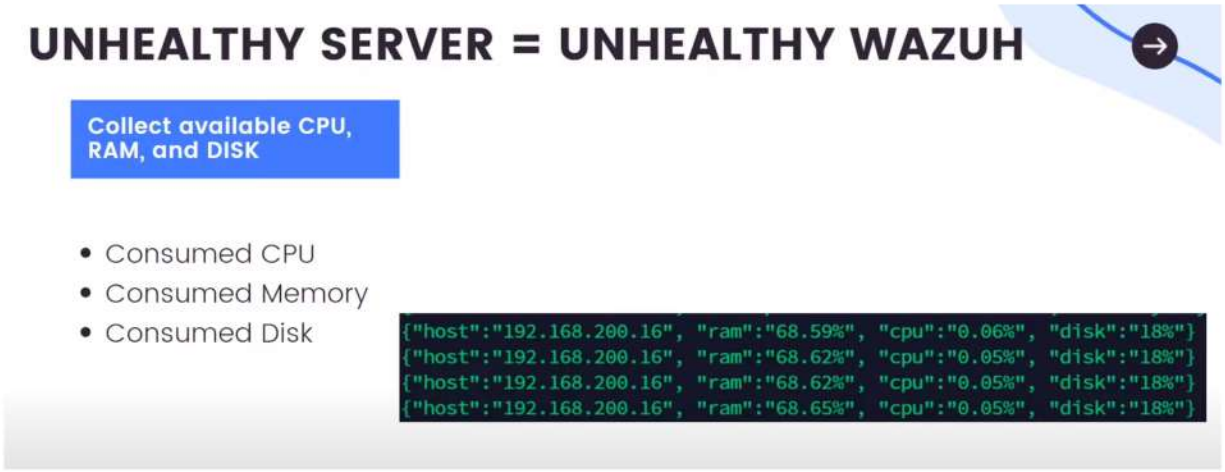
- Wazuh monitors CPU usage to identify processes consuming excessive resources. High CPU usage can slow down the system and impact overall performance.

12.1.3 Memory Utilization Monitoring

- Wazuh tracks memory usage to detect memory leaks or inefficient resource allocation by processes.

12.1.4 Rule-Based Alerts

- Wazuh uses preconfigured rules to detect abnormal behavior in system resources. These rules cover a wide range of scenarios,
- from sudden spikes in CPU usage to low disk space warnings.



- i want to show you guys how we can use wazuh to also detect an unhealthy server itself right
- so the wazuh manager is going to run as a service on a server whether that be a vm or a physical server and we need to make sure that our server is also in a healthy state some points that i want and some resources that i want to cover in this is consume cpu
- so if all of a sudden our cpu is starting to spike how much memory is currently being consumed and how much disk space is left for wazuh to actually write its alerts to uh so these three resources cpu memory and disk are of course crucial to your server.
- all right so i'm on my wazoo manager here uh so here i can just run like some simple commands like df h to get how much disk space is left for the root partition uh here you can see you still have gigs left i could do like a free mh to look at memory that i have available to the box

```
root@wazuh-elk:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G  7.3G   22G  26% /
tmpfs           3.9G   80K  3.9G   1% /dev/shm
tmpfs           1.6G  984K  1.6G   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
efivarfs         56K   24K   27K  48% /sys/firmware/efi/efivars
/dev/sda15      105M  6.1M   99M   6% /boot/efi
tmpfs           793M  4.0K  793M   1% /run/user/1001
root@wazuh-elk:~#
root@wazuh-elk:~#
root@wazuh-elk:~# free -mh
               total        used        free      shared  buff/cache   available
Mem:           7.7Gi       5.2Gi       513Mi       1.0Mi       2.1Gi       2.3Gi
Swap:             0B          0B          0B
root@wazuh-elk:~#
```

- what we want to do is have our script run all this for us

- So i have created a bash script that collect all the resources utilization of our wazuh manager.

```
#!/bin/bash
host=$(ip route get 8.8.8.8 | awk -F"src " 'NR==1{split($2,a," ");print a[1]}')
ram=$(free -m | awk 'NR==2{printf "%.2f%%\t\t", $3*100/$2 }' | sed 's/[ \t]*$//')
disk=$(df -h | awk '$NF=="/"{printf "%s\t\t", $5}' | sed 's/[ \t]*$//')
cpu=$(top -bn1 | grep load | awk '{printf "%.2f%%\t\t\n", $(NF-2)}' | sed 's/[ \t]*$//')
json='{"host":"'"$host"'", "ram":"'"$ram"'", "cpu":"'"$cpu"'", "disk":"'"$disk"'"}'
echo -e "$json" >> /tmp/health.json
~
~
~
```

- So just make a bash shell script file and add this script and save this off

- To understand properly this script you can refer this video https://youtu.be/Dc5VIf7hK9s?si=jds7mfiuLue2j0jU.

- Actually we have to change the permissions we have to give executable permission to our script.

```
root@wazuh-elk:/opt# chmod +x metric.sh
root@wazuh-elk:/opt#
root@wazuh-elk:/opt#
```

- Here we go so we can check our resources utilization from just one command kind of pretty and interesting right.

```
root@wazuh-elk:/opt# ./metric.sh
root@wazuh-elk:/opt# cat /etc/health.json
cat: /etc/health.json: No such file or directory
root@wazuh-elk:/opt#
root@wazuh-elk:/opt#
root@wazuh-elk:/opt# cat /tmp/health.json
{"host":"10.160.0.2", "ram":"66.79%", "cpu":"0.16%", "disk":"26%"}
root@wazuh-elk:/opt#
```

- Overall, Wazuh's automated system resource monitoring provides proactive detection and response to potential issues, helping you maintain a stable and reliable IT infrastructure.

Reference Material

Chapter 4:-

- https://youtu.be/mKijuwrTeRM?si=5WZyMKSD8BhQPlxe
- https://youtu.be/IdHcCIhHwv0?si=jAi2v71c2yN5VF4t

Chapter 6:-

- https://youtu.be/UpcwKai3218?si=IPOkJy0n12KraM9M

Chapter 7:-

- https://youtu.be/q84m_C5rJ5k?si=QJIJkoB41JAFZ_-Z

Chapter 8:-

- https://youtu.be/O7Q1dMuGMNM?si=QoQciJTX3kZeCY_O

Chapter 9:-

- https://youtu.be/fSpprRMqA_I?si=TTo3vd-ui7b-jYmn

Chapter 10:-

- https://youtu.be/2X1wVizsvjw?si=eyiZU_fVPhT4SfTJ

Chapter 11:-

- https://youtu.be/qWzq90brTl4?si=d26qEP83v_tOY4ge

Chapter 12:-

- https://youtu.be/Dc5VIf7hK9s?si=ZRHUpxvNF7SnBvWZ

Conclusion: Fortifying Cybersecurity Resilience with Wazuh

- In conclusion, the implementation of Wazuh has been instrumental in enhancing our cybersecurity resilience.
- By automating health monitoring and promptly resolving issues, we have significantly improved our system's reliability and reduced the risk of potential threats.
- The ability to create custom alert rules ensures that we can proactively address security issues before they escalate. Through this project,
- we have demonstrated the effectiveness of Wazuh in fortifying our defenses and maintaining a robust cybersecurity posture.
- As we continue to evolve our security measures, Wazuh will remain a critical component in safeguarding our digital assets and mitigating risks effectively.