Bangladesh University of Engineering and Technology

Department of Computer Science and Engineering

Academic Year 2023–2024

CSE 406
Computer Security Sessional

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

---

Submitted by:

1905001 — Mohammad Sadat Hossain

1905004 — Asif Azad

1905005 — Md. Ashrafur Rahman Khan

Supervised by:

Abdur Rashid Tushar

Lecturer

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Submission Date: March 13, 2024

Contents

1 Introduction to Wazuh **4**

1.1 What is Wazuh? . . . 4

1.2 Wazuh Components . . . 4

1.2.1 Wazuh Agent . . . 4

1.2.2 Wazuh Manager . . . 5

1.3 Wazuh Architecture . . . 6

2 Installation Prerequisites **7**

2.1 System Requirements . . . 7

2.1.1 Hardware Specifications . . . 7

2.1.2 Operating System Compatibility . . . 7

2.1.3 Web Browser Support . . . 7

2.2 Configuring the Machines . . . 8

2.2.1 Wazuh Server . . . 8

2.2.2 Wazuh Agents . . . 8

3 Installation **10**

3.1 Setting Up the Wazuh Server . . . 10

3.1.1 Quickstart Installation . . . 10

3.1.2 Step-by-step Installation . . . 11

3.2 Registering Agents . . . 11

3.2.1 Linux . . . 12

3.2.2 MacOS . . . 13

3.2.3 Windows . . . 14

4 Wazuh Features and Use-cases **16**

4.1 File Integrity Module . . . 16

4.1.1 How it works . . . 17

4.1.2 Configuration . . . 18

4.1.3 Simulation . . . 19

Detecting Account Manipulation . . . 19

Monitoring Configuration Changes . . . 20

4.1.4 Dashboard Update . . . 21

Detecting Account Manipulation . . . 21

Monitoring Configuration Changes . . . 22

4.2 Malware Detection . . . 23

4.2.1 CDB Lists and Threat Intelligence . . . 23
How it works . . . 23
Configuration . . . 24
Simulation . . . 26
Dashboard Update . . . 26
4.2.2 File Integrity Monitoring and YARA Scanning . . . 28
How it works . . . 28
Configuration . . . 29
Simulation . . . 34
Dashboard Update . . . 36
4.2.3 VirusTotal Integration . . . 37
How it works . . . 37
Configuration . . . 43
Simulation . . . 44
Dashboard Update . . . 45
4.2.4 Windows Defender Logs Collection . . . 47
How it works . . . 48
Configuration . . . 61
Simulation . . . 62
Dashboard Update . . . 63
4.3 Security Configuration Assessment . . . 64
4.3.1 How it works . . . 64
Overview of an SCA Check . . . 65
Scan Results . . . 66
Integrity Mechanism . . . 70
4.3.2 Configuration . . . 70
Enabling and Disabling Policies . . . 71
Sharing Policy Files and Configuration with the Wazuh Agents . . . 72
4.3.3 Simulation . . . 72
Detecting Keyword in a File . . . 73
4.3.4 Dashboard Update . . . 75
Detecting Keyword in a File . . . 75
4.4 Threat Hunting . . . 76
4.4.1 Log Data Analysis . . . 76
4.4.2 Wazuh Archives . . . 77
4.4.3 MITRE ATT&CK Mapping . . . 77
4.4.4 Third-party Integration . . . 79

4.4.5 Rules and Decoders . . . 79
4.5 Log Data Analysis . . . 80
4.5.1 How it works . . . 80
4.5.2 Configuration . . . 82
Using Syslog . . . 82
Using Wazuh Agent . . . 83
4.5.3 Simulation . . . 84
Linux Log Data Analysis using `rsyslog` . . . 84
Windows Log Data Analysis using Wazuh Agent . . . 85
4.5.4 Dashboard Update . . . 86
Linux Log Data Analysis using `rsyslog` . . . 86
Windows Log Data Analysis using Wazuh Agent . . . 87

5 Source Code **88**
5.1 Repositories . . . 88
5.2 Overview of the Core Wazuh Source Code . . . 89
5.3 Compiling the Front-end from Source . . . 90

6 Issues Faced **95**
6.1 Server Crash: macOS Not Supporting Realtime Monitoring . . . 95
6.2 Random Authentication Error on Server . . . 95
6.3 Source Code Compilation Challenges . . . 95

7 Reference **95**

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

1 INTRODUCTION TO WAZUH

1.1 WHAT IS WAZUH?

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

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

1.2 WAZUH COMPONENTS

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

1.2.1 WAZUH AGENT

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

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

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

- Container security
- Cloud security

1.2.2 WAZUH MANAGER

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

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

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

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

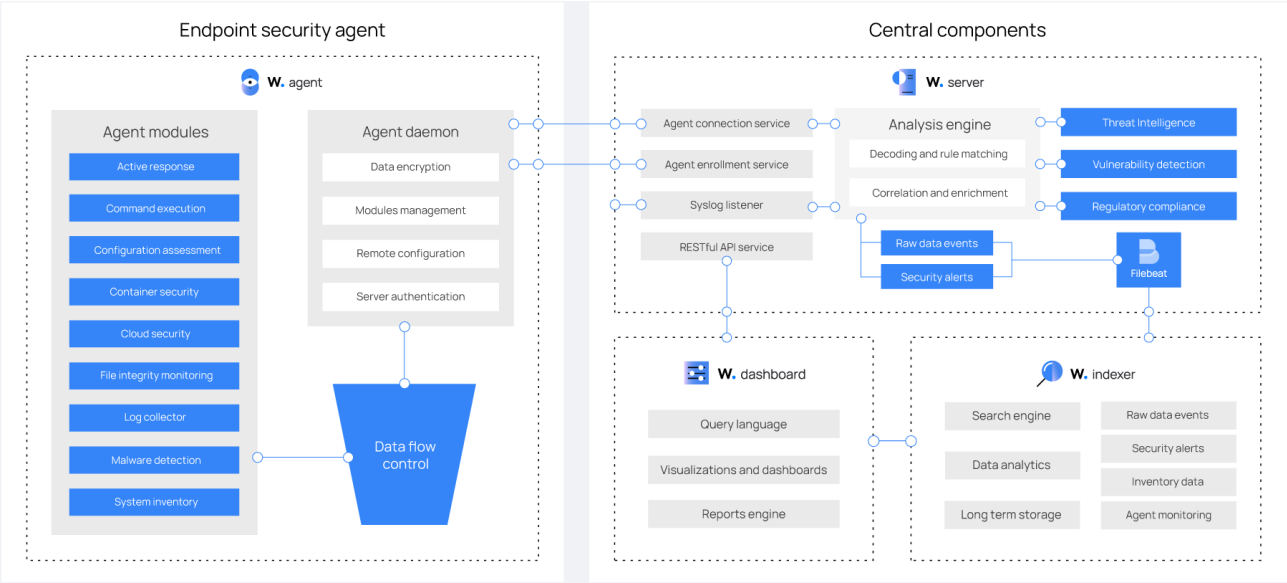


Figure 1: Wazuh Components and Data flow

1.3 WAZUH ARCHITECTURE

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

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

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

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

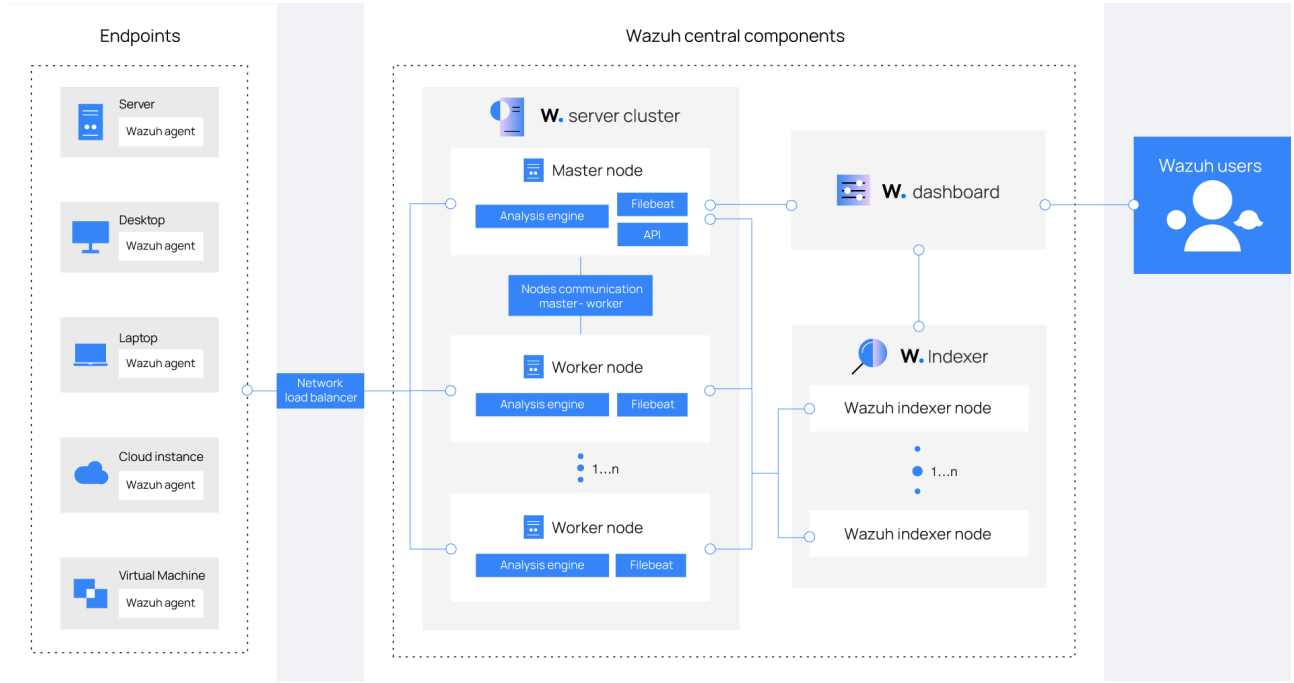


Figure 2: Overview of Wazuh Deployment Architecture

2 INSTALLATION PREREQUISITES

2.1 SYSTEM REQUIREMENTS

2.1.1 HARDWARE SPECIFICATIONS

The scale of hardware requisite directly correlates with the quantity of endpoints and cloud services. For typical use cases, the consolidation of the Wazuh server, indexer, and dashboard within a single host usually suffices, as this is adequate for supervising upto 100 endpoints and maintaining ninety days of accessible alert data. The following table delineates the advisable hardware for such an initial setup:

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

Table 1: Recommended Hardware for Quickstart Deployment

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

2.1.2 OPERATING SYSTEM COMPATIBILITY

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

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

2.1.3 WEB BROWSER SUPPORT

The Wazuh dashboard is compatible with the following browsers:

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

2.2 CONFIGURING THE MACHINES

2.2.1 WAZUH SERVER

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

2.2.2 WAZUH AGENTS

AGENT ID: 001

- **Computer Name:** wazuh-agent-linux-1
- **Operating System:** Ubuntu 22.04.3 LTS
- **Size:** Standard B1s, 1 vCPU, 1GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.6

AGENT ID: 002

- **Computer Name:** wazuh-agent-win
- **Operating System:** Microsoft Windows 11 Pro 10.0.22000.2538
- **Size:** Standard B1s, 1 vCPU, 1GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.4

AGENT ID: 007

- **Computer Name:** seed-vm
- **Operating System:** Ubuntu 20.04.6 LTS
- **Size:** Standard B2s, 2 vCPUs, 4GB RAM

- **Public IP:** N/A
- **Private IP:** 10.0.0.4

AGENT ID: 008

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

AGENT ID: 009

Understandably, macOS integration could not be done on a virtual machine. We used a physical machine for this purpose.

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

AGENT ID: 010

- **Computer Name:** Sadat-Windows
- **Operating System:** Microsoft Windows 11 Pro 10.0.22621.3155
- **Size:** Standard DS2, 2 vCPUs, 7GB RAM
- **Public IP:** N/A
- **Private IP:** 10.1.0.4

3 INSTALLATION

3.1 SETTING UP THE WAZUH SERVER

There are two methods to setup the Wazuh Server:

3.1.1 QUICKSTART INSTALLATION

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

1. Download and run the Wazuh installation assistant.

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

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

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

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

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

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

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

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

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

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

3.1.2 STEP-BY-STEP INSTALLATION

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

3.2 REGISTERING AGENTS

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

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

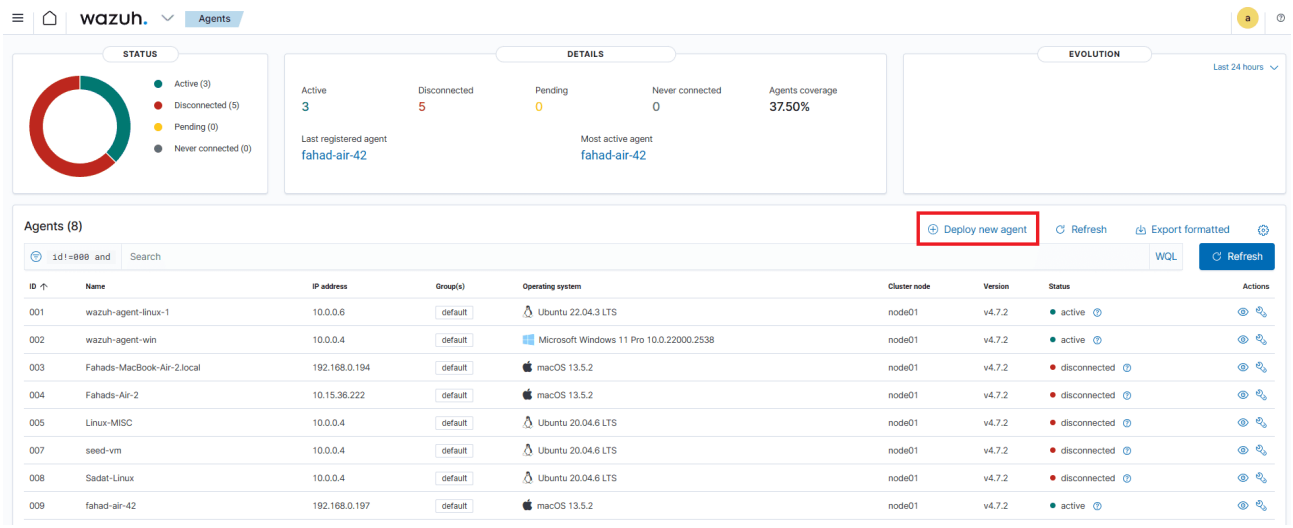


Figure 3: Wazuh Dashboard - Deploy New Agent

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

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

3.2.1 LINUX

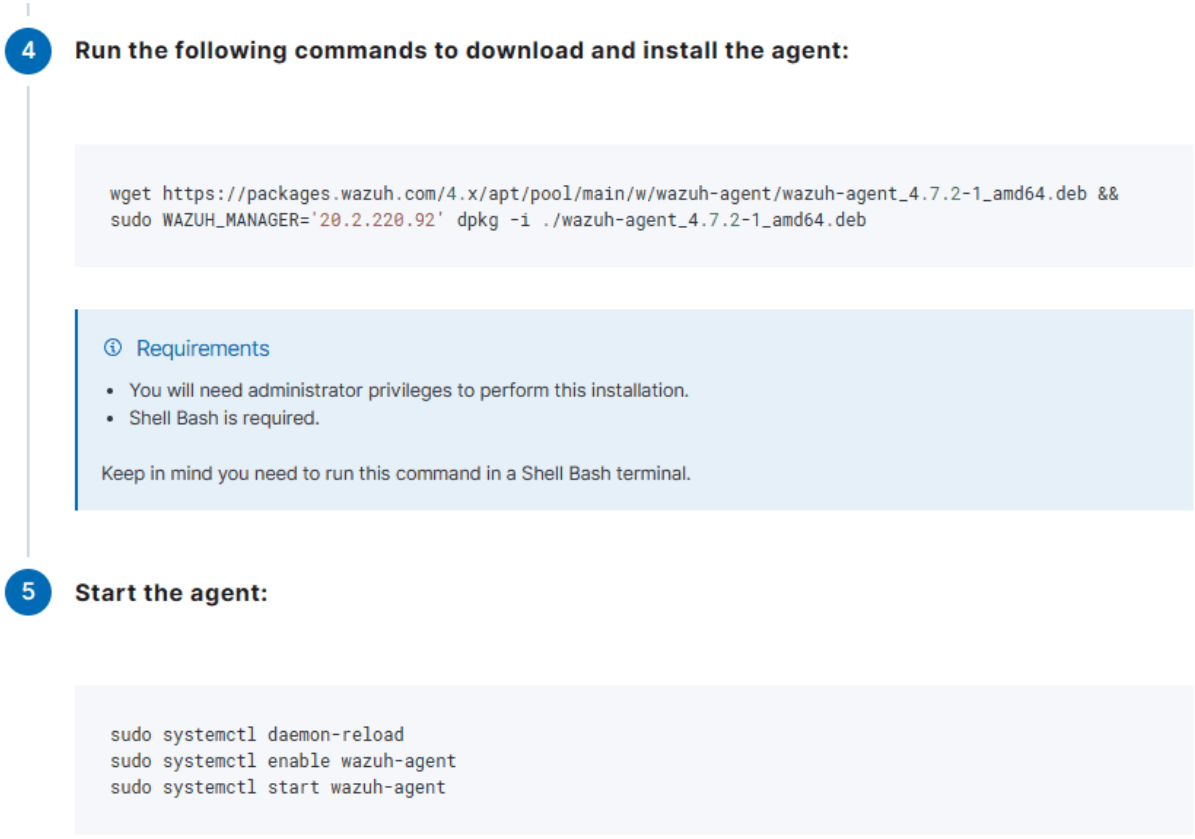


Figure 4: Wazuh Agent Installation Commands for a Linux Machine

The commands in the picture go as follows:

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

3.2.2 MACOS

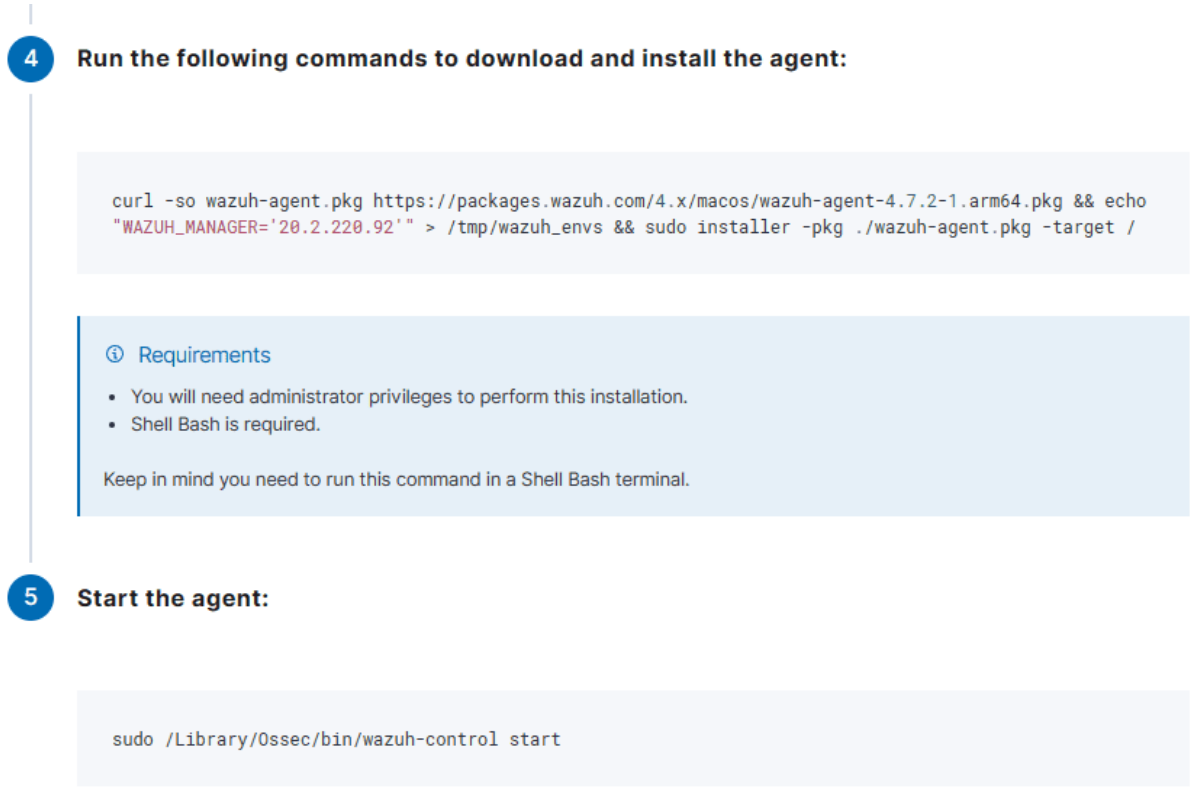


Figure 5: Wazuh Agent Installation Commands for a macOS Machine

The commands are:

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

3.2.3 WINDOWS

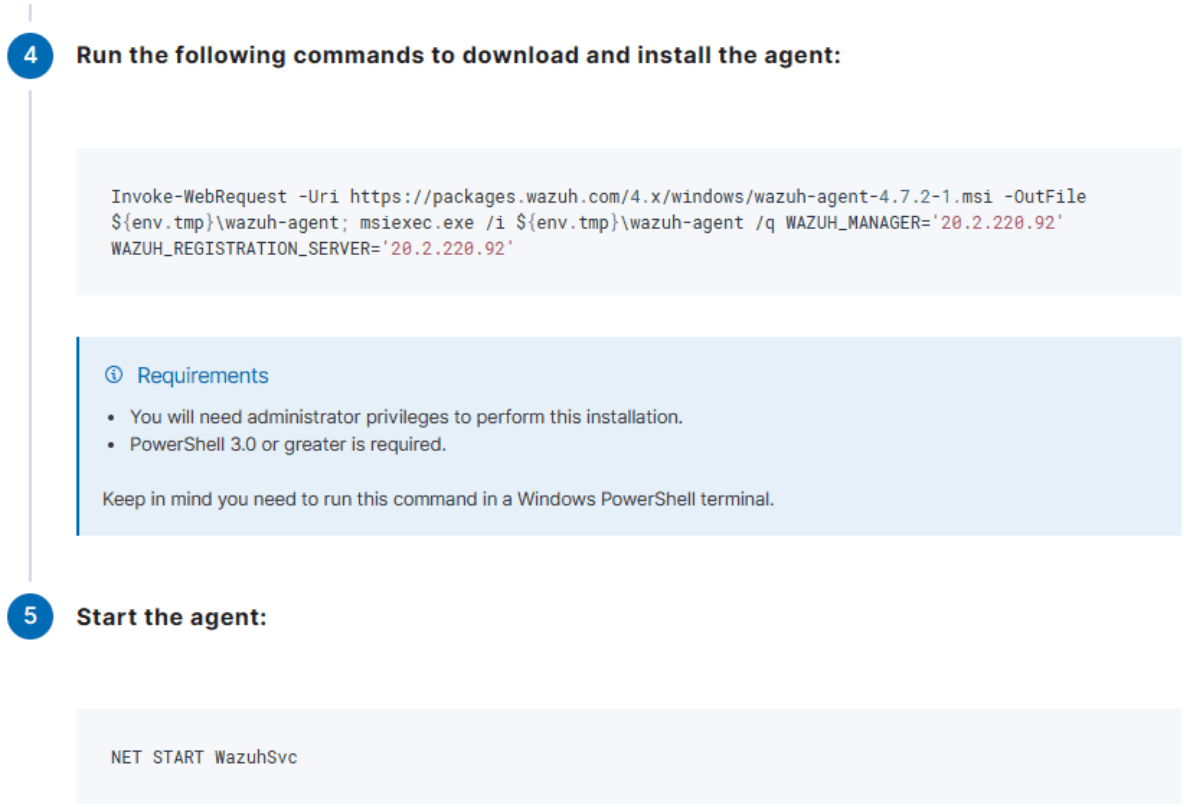


Figure 6: Wazuh Agent Installation Commands for a Windows Machine

The commands are compiled here:

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

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

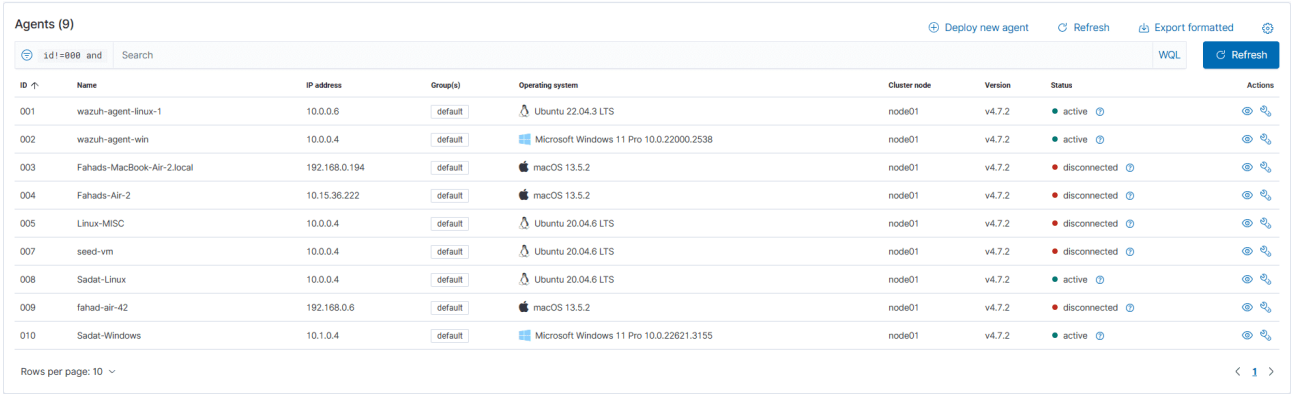


Figure 7: Installed Agents

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

4 WAZUH FEATURES AND USE-CASES

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

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

The first five of these are explored in the subsequent sections.

4.1 FILE INTEGRITY MODULE

File Integrity Monitoring (FIM) is a security process used to monitor the integrity of system and application files. FIM is an important security defense layer for any organization monitoring sensitive assets. It provides protection for sensitive data, application, and device files by monitoring, routinely scanning, and verifying their integrity. It helps organizations detect changes to critical files on their systems which reduces the risk of data being stolen or compromised. This process can save time and money in lost productivity, lost revenue, reputation damage, and legal and regulatory compliance penalties.

Wazuh has a built-in capability for file integrity monitoring. The Wazuh FIM module monitors files and directories and triggers an alert when a user or process creates, modifies, and deletes monitored files. It runs a baseline scan, storing the cryptographic checksum and other attributes of the monitored files. When a user or process changes a file, the module compares

its checksum and attributes to the baseline. It triggers an alert if it finds a mismatch. The FIM module performs real-time and scheduled scans depending on the FIM configuration for agents and manager.

4.1.1 HOW IT WORKS

The FIM module runs periodic scans on specific paths and monitors specific directories for changes in real time. You can set which paths to monitor in the configuration of the Wazuh agents and manager.

FIM stores the files checksums and other attributes in a local FIM database. Upon a scan, the Wazuh agent reports any changes the FIM module finds in the monitored paths to the Wazuh server. The FIM module looks for file modifications by comparing the checksums of a file to its stored checksums and attribute values. It generates an alert if it finds discrepancies.

The Wazuh FIM module uses two databases to collect FIM event data, such as file creation, modification, and deletion data. One is a local SQLite-based database on the monitored endpoint that stores the data in:

- `C:\Program Files (x86)\ossec-agent\queue\fim\db` on Windows.
- `/var/ossec/queue/fim/db` on Linux.
- `/Library/Ossec/queue/fim/db` on macOS.

The other is an agent database on the Wazuh server. The wazuh-db. daemon creates and manages a database for each agent on the Wazuh server. It uses the ID of the agent to identify the database. This service stores the databases at `/var/ossec/queue/db`.

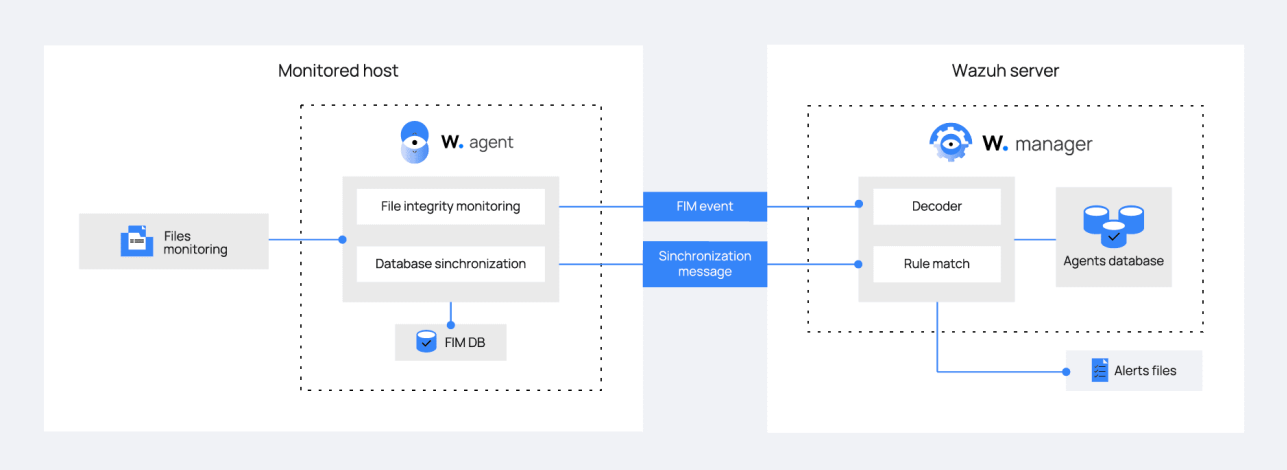


Figure 8: Working Flow of Wazuh File Integrity Module

The FIM module keeps the Wazuh agent and the Wazuh server databases synchronized with each other. It always updates the file inventory in the Wazuh server with the data available to

the Wazuh agent. An up-to-date Wazuh server database allows for servicing FIM-related API queries. The synchronization mechanism only updates the Wazuh server with information from the Wazuh agents such as checksums and file attributes that have changed.

The Wazuh agent and manager have the FIM module enabled and pre-configured by default. However, we recommend that you review the configuration of your endpoints to ensure that you tailor the FIM settings, such as monitored paths, to your environment.

4.1.2 CONFIGURATION

The FIM module runs scans on Windows, Linux, and macOS operating systems. There are both global settings and settings that are specific to the operating system of the endpoint. We discuss these settings and the supported operating systems in the Basic settings section of this guide.

You must specify the directories where the FIM module must monitor the creation, modification, and deletion of files or configure the specific files you need to monitor. You can specify the file or directory to monitor on the Wazuh server and the Wazuh agent configuration files. You can also configure this capability remotely using the centralized configuration file.

You have to set the files and directories to monitor with the directories options. You can include multiple files and directories using comma-separated entries or adding entries on multiple lines. You can configure FIM directories using `*` and `?` wildcards in the same way you would use them in a shell or Command Prompt (cmd) terminal. For example, `C:\Users\*\Downloads`.

Any time the FIM module runs a scan, it triggers alerts if it finds modified files and depending on the changed file attributes. You can view these alerts in the Wazuh dashboard.

Following, you can see how to configure the FIM module to monitor a file and directory. Replace `FILEPATH/OF/MONITORED/FILE` and `FILEPATH/OF/MONITORED/DIRECTORY` with your own filepaths.

- Add the following settings to the Wazuh agent configuration file, replacing the directories values with your own filepaths:
  - Linux: `/var/ossec/etc/ossec.conf`
  - Windows: `C:\Program Files (x86)\ossec-agent\ossec.conf`
  - macOS: `/Library/Ossec/etc/ossec.conf`

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

- Restart the Wazuh agent with administrator privilege to apply any configuration change:
  - Linux: systemctl restart wazuh-agent
  - Windows: Restart-Service -Name wazuh
  - macOS: /Library/Ossec/bin/wazuh-control restart

4.1.3 SIMULATION

We demonstrate the following two use-cases of file integrity module.

DETECTING ACCOUNT MANIPULATION

Account manipulation refers to the creation, modification, or deletion of user accounts or other credentials within an organization's IT infrastructure. Monitoring this activity is critical to the cybersecurity of an organization. Unauthorized account manipulations might grant an attacker access to sensitive systems and data.

To maintain persistence on a victim endpoint, adversaries can alter the SSH `authorized_keys` file to add their public key. This allows them to access the system remotely without needing to authenticate with a password. We simulate this activity by adding a new public key to the `authorized_keys` file.

Ubuntu endpoint

- We edited the `/var/ossec/etc/ossec.conf` configuration file and add `authorized_keys` for monitoring:

```
<syscheck>
    <directories whodata="yes"
    ↪ realtime="yes">/home/*/.ssh/authorized_keys</directories>
</syscheck>
```

- We restarted the Wazuh agent to apply the configuration:

```
systemctl restart wazuh-agent
```

- We generated an SSH keypair for user authentication and saved it as `.ssh/test_key` using the following command:

```
ssh-keygen -f .ssh/test_key
```

- We ran the following command to copy the content of the generated SSH public key `test_key.pub` and added it to the `authorized_keys` file in the target Ubuntu user `.ssh` directory.

```
cat ~/.ssh/test_key.pub | ssh -i
↪ github/cse406/wazuh/wazuh-agent-linux-1_key.pem
↪ asifazad@20.205.141.120 "sudo tee -a
↪ /home/asifazad/.ssh/authorized_keys"
```

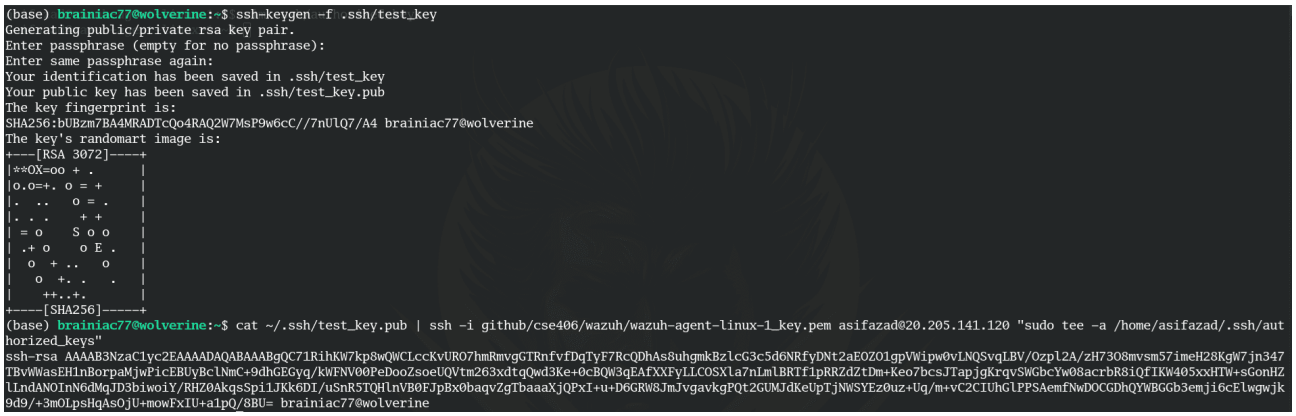


Figure 9: Testing Commands in Terminal

MONITORING CONFIGURATION CHANGES

Monitoring configuration changes helps to establish accountability for changes made to systems and applications. Organizations can identify responsible parties and ensure that changes are properly authorized and documented by maintaining a record of changes and who made them.

We can configure the FIM module to monitor configuration files and report any changes. The Wazuh FIM module uses the `whodata` and `report_changes` attributes to record the following information about such changes:

- The login user that made the changes.
- The time of the changes.
- The process that the user executed.
- The changes made to the file.

Ubuntu endpoint

- We created a file `app.conf` in the `/etc` directory.

```
touch /etc/app.conf
```

- We edited the `/var/ossec/etc/ossec.conf` configuration file and add the configuration below:

```
<syscheck>
    <directories realtime="yes" check_all="yes" report_changes="yes"
    ↪  whodata="yes">/etc/app.conf</directories>
</syscheck>
```

- We restarted the Wazuh agent to apply the configuration changes:

```
systemctl wazuh-agent restart
```

- We modified the `/etc/app.conf` file by using vim with root privilege:

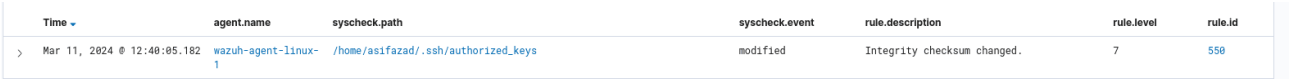
```
vim /etc/app.conf
```

- We added `updated image to V2` to the file and save.

4.1.4 DASHBOARD UPDATE

DETECTING ACCOUNT MANIPULATION

- We navigated to `Modules > Integrity monitoring` on the Wazuh dashboard to view the alert generated when the FIM module detects changes to the `authorized_keys` file.



	Time	agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
›	Mar 11, 2024 @ 12:40:05.182	wazuh-agent-linux-1	/home/asifazad/.ssh/authorized_keys	modified	Integrity checksum changed.	7	550

Figure 10: Alert for Change in `authorized_keys` File

- We extended the alert for detailed information.

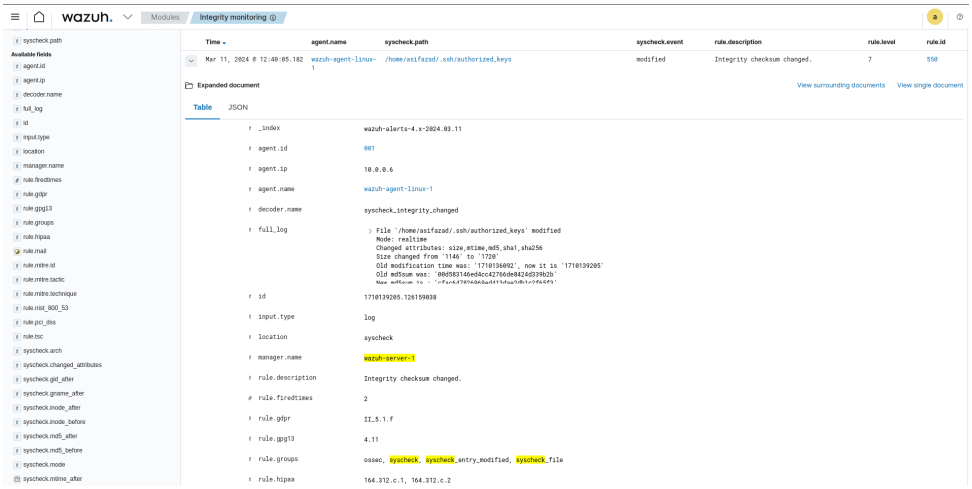


Figure 11: Alert Details for Change in `authorized_keys` File

MONITORING CONFIGURATION CHANGES

- We navigated to `Modules > Integrity` monitoring on the Wazuh dashboard to view the alert generated when the FIM module detects modification of the configuration file.

Time	agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
Mar 11, 2024 @ 12:26:17.085	wazuh-agent-linux-1	/etc/app.conf	modified	Integrity checksum changed.	7	550

Figure 12: Alert for Change in `/etc/app.conf` File

- We expanded the alert to get more information about the event. In this example, the vim text editor modified the configuration file. The logged-in user on the endpoint was ubuntu. The user modified the file using root privilege. The content added to the file is `updated image to V2`.

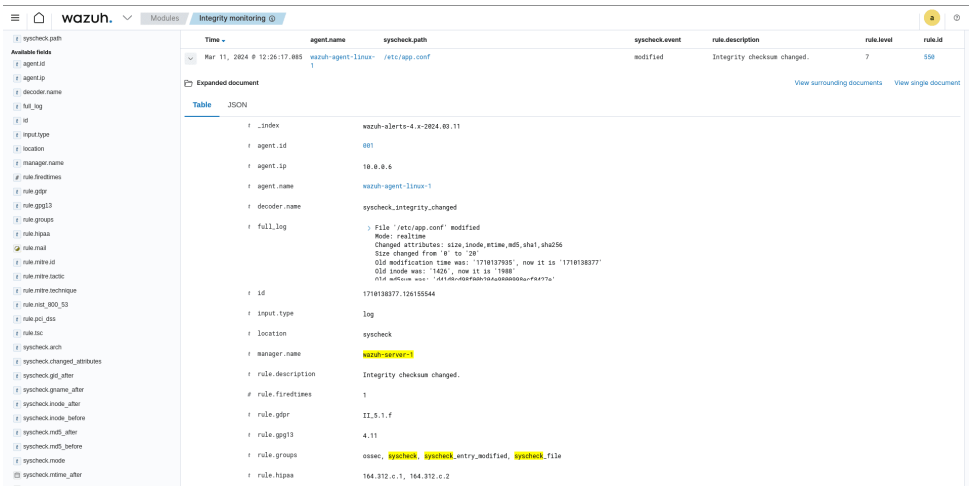


Figure 13: Alert Details for Change in `/etc/app.conf` File

4.2 MALWARE DETECTION

There are multiple ways to adopt malware detection strategies through Wazuh.

- Rootkits behavior detection
- CDB lists and threat intelligence
- VirusTotal integration
- File integrity monitoring and YARA
- ClamAV logs collection
- Windows Defender logs collection
- Custom rules to detect malware IOCs
- Osquery

Among these, we explore CDB lists, VirusTotal integration, YARA scanning and Windows Defender logs collection.

4.2.1 CDB LISTS AND THREAT INTELLIGENCE

HOW IT WORKS

Wazuh utilizes CDB lists to cross-reference field values like IP addresses, file hashes, and others, obtained from decoding security events, facilitating the identification and tracking of malware. This functionality extends to leveraging CDB lists alongside the File Integrity Monitoring (FIM) module for enhanced malware detection. The operational framework is detailed as follows:

1. **File Integrity Monitoring:** The FIM module conducts surveillance over designated directories on endpoints, aiming to spot any occurrences such as the inception or alteration of files. It meticulously records the checksums alongside other relevant attributes of the files it monitors.
2. **Alert Generation:** Upon the creation of an alert by the FIM module, Wazuh's analytical engine proceeds to juxtapose the attributes of the file in question, such as its hash, against the keys housed within a specifically chosen CDB list.
3. **Alert Management:** Should there be a discovery of a match by the analysis engine, it either triggers or suppresses an alert contingent upon the configuration settings established by the user.

This process underscores Wazuh's capability to not only monitor and record file integrity but also to utilize those findings in conjunction with CDB lists for robust malware detection and response strategies.

CONFIGURATION

Wazuh server

1. Create a CDB list `malware-hashes` of known malware hashes and save it to the `/var/ossec/etc/lists` directory on the Wazuh server.

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

2. Add the known malware hashes to the file as `key:value` pairs. In this case, you can use the known MD5 hashes of the Mirai and Xbash malware as shown below.

```
e0ec2cd43f71c80d42cd7b0f17802c73:mirai
55142f1d393c5ba7405239f232a6c059:Xbash
```

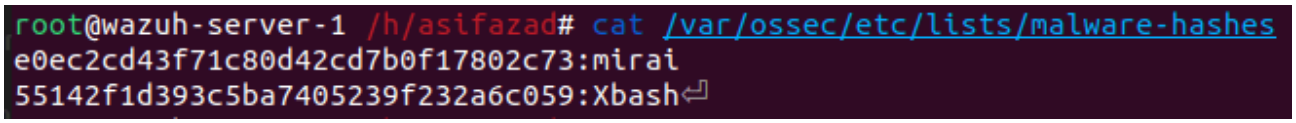


Figure 14: List of Malware Hashes (Terminal)

Alternatively, these configurations can also be updated from the Wazuh dashboard, like the following:

Figure 15: List of Malware Hashes (Dashboard)

3. Add a reference to the CDB list in the Wazuh manager configuration file `/var/ossec/etc/ossec.conf`. This can be done by specifying the path to the list within 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 16: Add Malware List to Ruleset

4. Create a custom rule in the `/var/ossec/etc/rules/local_rules.xml` file on the Wazuh server. The rule generates alerts when the Wazuh analysis engine matches the MD5 hash of a new or modified file to a hash in the CDB list. Rules 554 and 550 must previously match indicating a recently created or modified file.

```
<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 17: Custom Rule added to Server

5. Restart the Wazuh manager to apply changes.

```
systemctl restart wazuh-manager
```

Linux endpoint

1. Configure directory monitoring by adding the `<directories>` block specifying the folders that need to be monitored in the agent configuration file or using the centralized configuration option. We will monitor the `/fim` directory here.

```
root@Sadat999-MISC /h/S/fim# cat /var/ossec/etc/ossec.conf | tail -6
<ossec_config>
  <syscheck>
    <disabled>no</disabled>
    <directories check_all="yes" realtime="yes">/fim</directories>
  </syscheck>
</ossec_config>
```

Figure 18: Adding a Monitored Directory

2. Restart the Wazuh agent to apply the changes:

```
systemctl restart wazuh-agent
```

SIMULATION

To test that everything works correctly, we need to download the Mirai and Xbash malware samples to the directory the FIM module is monitoring.

1. We need to download the malware samples.

```
sudo curl https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai --output
↪  /fim/mirai
sudo curl https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash --output
↪  /fim/Xbash
```

```
root@Sadat999-MISC /# sudo curl https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai --output /fim/mirai
                      sudo curl https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash --output /fim/xbash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 79804  100 79804    0     0  58765      0  0:00:01  0:00:01 --:--:-- 58765
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 9344k  100 9344k    0     0  3285k      0  0:00:02  0:00:02 --:--:-- 3284k
```

Figure 19: Manually Downloading the Malwares

DASHBOARD UPDATE

The alerts can be seen on the Wazuh dashboard. To do this, navigate to the following:

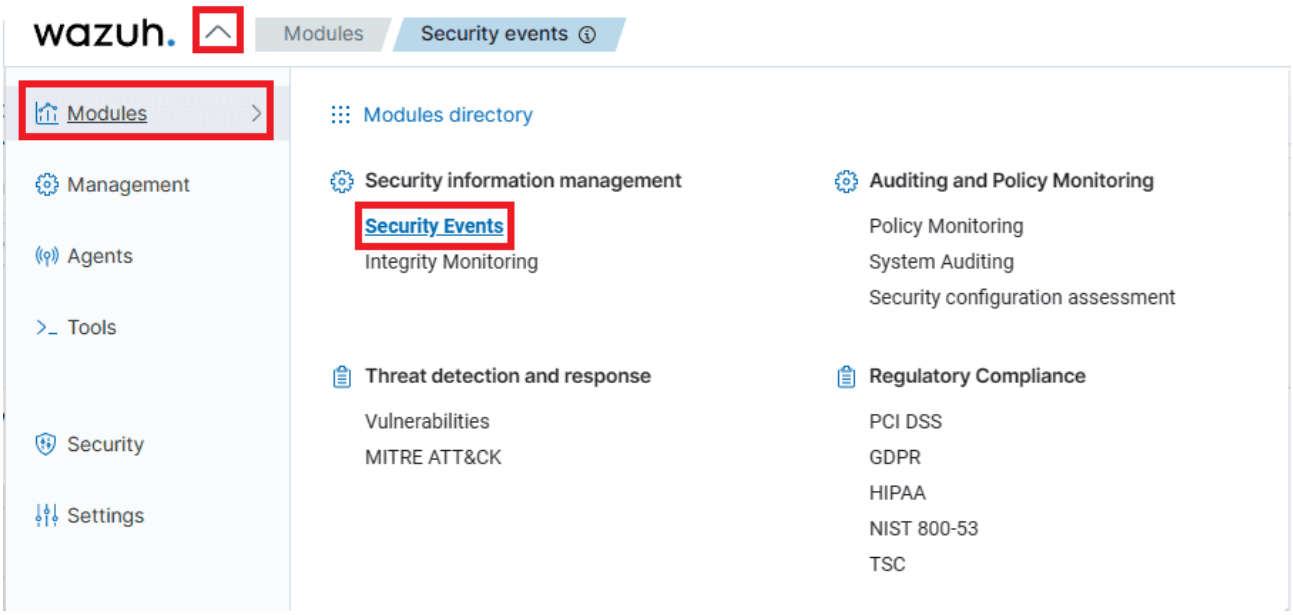


Figure 20: Navigation to Security Events

As per our defined rules, two level 13 alerts should have been generated, for which the number of level 12 or above alerts is now 15, previously this was 13.

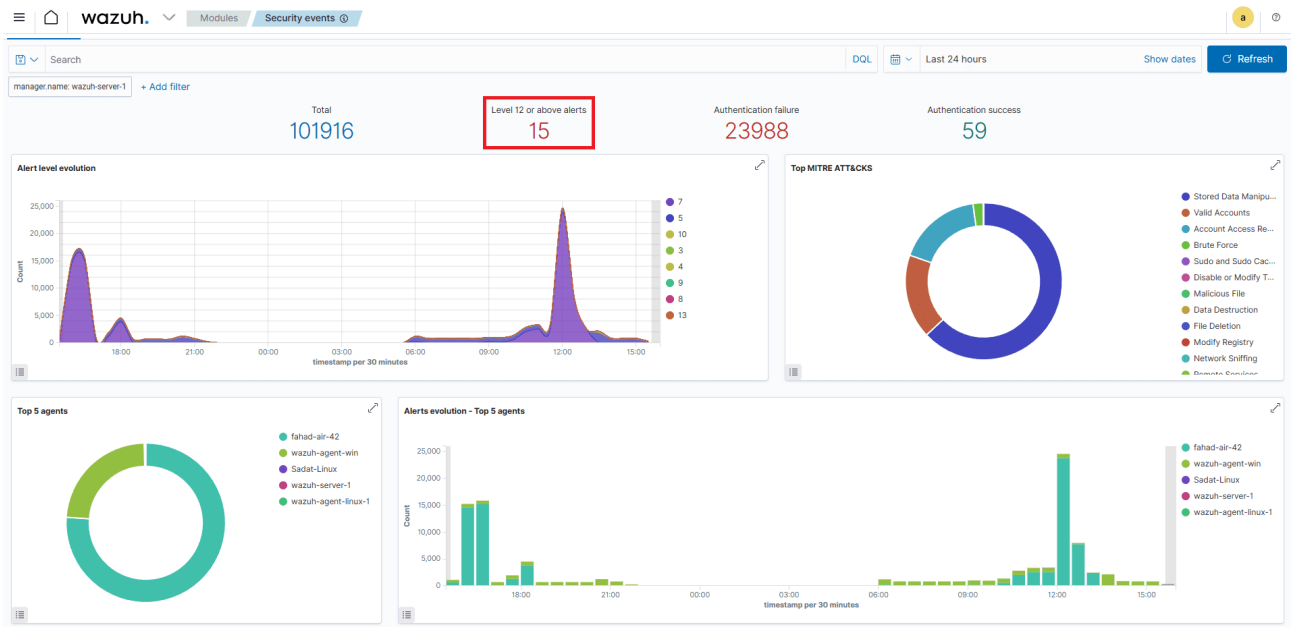


Figure 21: Dashboard after Malware Download

At the bottom, we can see two new alerts have been generated at the latest time because of the two malwares downloaded. We can see further details for them as well upon clicking.

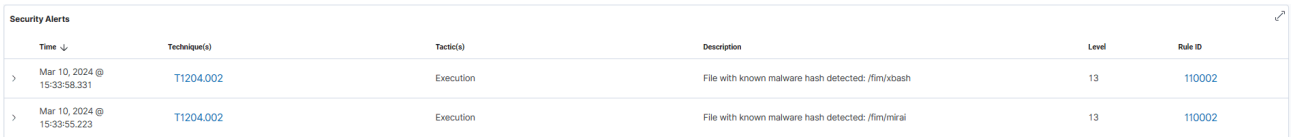


Figure 22: Alerts Generated by CDB Matching

4.2.2 FILE INTEGRITY MONITORING AND YARA SCANNING

HOW IT WORKS

This methodology employed for malware detection unfolds through several phases as follows:

1. The File Integrity Monitoring (FIM) feature of Wazuh scrutinizes directories on endpoints to identify any alterations, including the creation or modification of files.

2. Upon identifying a modification in any monitored directory or file, FIM initiates a YARA scan as part of its active response mechanism. This is executed through the `yara.sh` script, which subsequently examines the implicated file against its YARA rules to ascertain if it contains malware.

3. Should the YARA rules find a match for the file, the ensuing scan data is sent to the Wazuh manager for decoding, analysis, and generation of alerts. It's important to note that these scan outcomes are not immediately interpretable and require the integration of specific decoders into your Wazuh server.

The diagram below illustrates the flow of events between the different components.

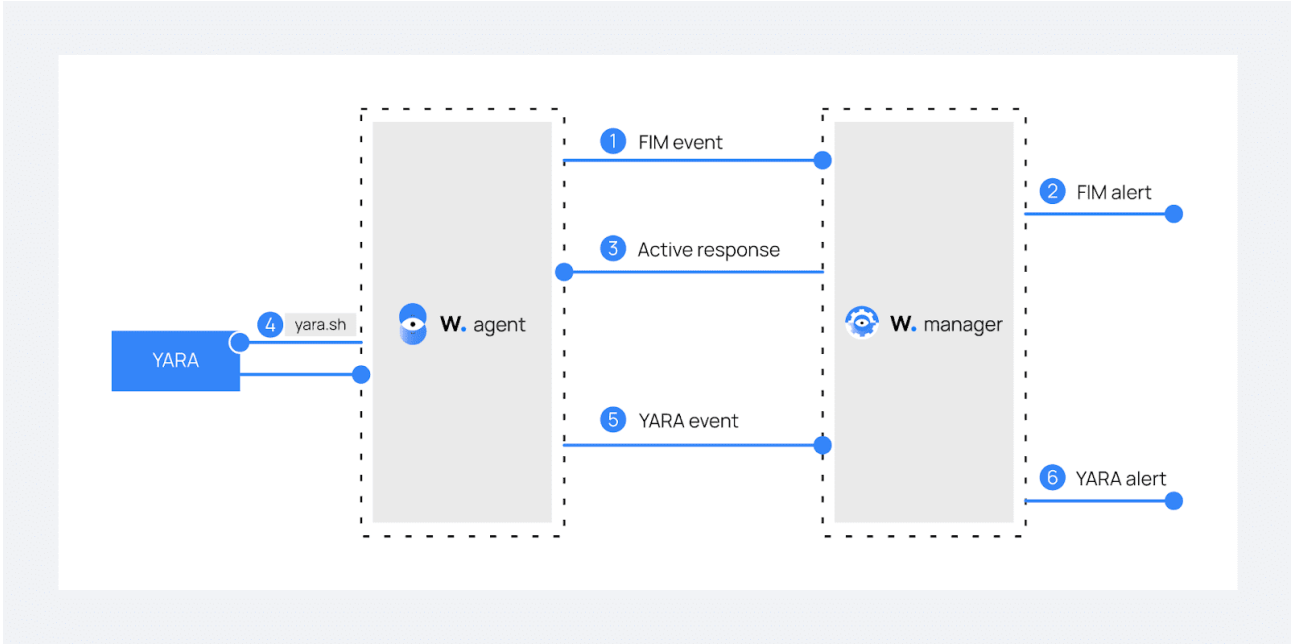


Figure 23: Workflow of Malware Detection through YARA scanning

This YARA scanning procedure, integrated into the active response system, focuses its analysis on either newly created or recently altered files within the monitored directories, thereby ensuring efficient utilization of resources across the endpoints.

CONFIGURATION

Linux endpoint

1. Download, compile, and install YARA:

```
sudo apt update
sudo apt install -y make gcc autoconf libtool libssl-dev pkg-config
sudo curl -LO https://github.com/VirusTotal/yara/archive/v4.2.3.tar.gz
sudo tar -xvzf v4.2.3.tar.gz -C /usr/local/bin/ && rm -f v4.2.3.tar.gz
cd /usr/local/bin/yara-4.2.3/
sudo ./bootstrap.sh && sudo ./configure && sudo make && sudo make
↪  install && sudo make check
```

2. Test that YARA is running properly.

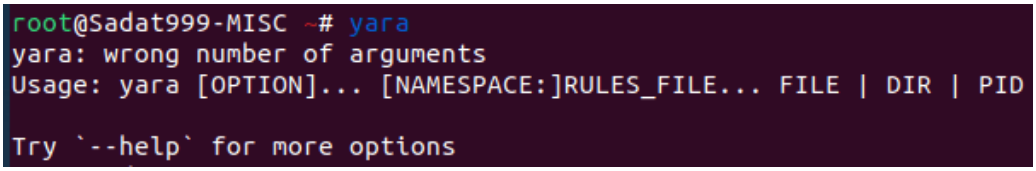


Figure 24: Checking YARA Installation

If it asks for right number of arguments as shown in the image above, then the installation has worked correctly. However, an error might occur saying that shared object file can't be opened. This means that the loader doesn't find the `libyara` library usually located in `/usr/local/lib`. The path `/usr/local/lib` has to be added to the `/etc/ld.so.conf` loader configuration file to solve this.

```
sudo su
echo "/usr/local/lib" >> /etc/ld.so.conf
ldconfig
```

3. Download YARA detection rules:

```
sudo mkdir -p /tmp/yara/rules
sudo curl 'https://valhalla.nextron-systems.com/api/v1/get' \
-H 'Accept:
↪  text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US,en;q=0.5' \
--compressed \
-H 'Referer: https://valhalla.nextron-systems.com/' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'DNT: 1' -H 'Connection: keep-alive' -H 'Upgrade-Insecure-Requests:
↪  1' \
--data 'demo=demo&apikey=1111111111111111111111111111111111111111111111
↪  1111111111111111&format=text'
↪  \
-o /tmp/yara/rules/yara_rules.yar
```

4. Create a `/var/ossec/active-response/bin/yara.sh` file and add the content below:

```
#!/bin/bash
# Wazuh - Yara active response
# Copyright (C) 2015-2022, Wazuh Inc.
#
# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation.


#------------------------ Gather parameters
↪  -------------------------#

# Extra arguments
read INPUT_JSON
YARA_PATH=$(echo $INPUT_JSON | jq -r .parameters.extra_args[1])
YARA_RULES=$(echo $INPUT_JSON | jq -r .parameters.extra_args[3])
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.syscheck.path)
```

```
# Set LOG_FILE path
LOG_FILE="logs/active-responses.log"

size=0
actual_size=$(stat -c %s ${FILENAME})
while [ ${size} -ne ${actual_size} ]; do
    sleep 1
    size=${actual_size}
    actual_size=$(stat -c %s ${FILENAME})
done

#------------------------ Analyze parameters -------------------------#

if [[ ! $YARA_PATH ]] || [[ ! $YARA_RULES ]]
then
    echo "wazuh-yara: ERROR - Yara active response error. Yara path and
    ↪  rules parameters are mandatory." >> ${LOG_FILE}
    exit 1
fi

#-------------------------- Main workflow --------------------------#

# Execute Yara scan on the specified filename
yara_output="$("${YARA_PATH}"/yara -w -r "$YARA_RULES" "$FILENAME")"

if [[ $yara_output != "" ]]
then
    # Iterate every detected rule and append it to the LOG_FILE
    while read -r line; do
        echo "wazuh-yara: INFO - Scan result: $line" >> ${LOG_FILE}
    done <<< "$yara_output"
fi

exit 0;
```

This active response script receives these parameters from the generated FIM alerts:

- The file path contained in the alert that triggered the active response. The value of the file path is held in the `parameters.alert.syscheck.path` key of the JSON alert. The path in this use case is `/root/`.
- `YARA_PATH`: This variable specifies the path to the directory where the YARA executable is located. We installed YARA in the `/usr/local/bin` directory as shown in step 2 above.
- `YARA_RULES`: This variable specifies the path to the file containing the YARA rules used for the scan.

This snippet of the script uses the parameters above to perform a YARA scan and appends the results to a log file called `active-responses.log`. For every line in the output of the YARA scan, the script appends an event to the active response log, `/var/ossec/logs/active-responses.log`.

5. Change the script ownership and permissions with the following commands:

```
sudo chmod 750 /var/ossec/active-response/bin/yara.sh
sudo chown root:wazuh /var/ossec/active-response/bin/yara.sh
```

6. Install the `jq` utility to process the JSON data from the FIM alerts:

```
sudo apt install -y jq
```

7. Add the following within the `<syscheck>` block of the Wazuh agent `/var/ossec/etc/ossec.conf` configuration file to monitor the `/tmp/yara/malware` directory:

```
<directories realtime="yes">/tmp/yara/malware</directories>
```

8. Restart the Wazuh agent to apply the configuration changes:

```
sudo systemctl restart wazuh-agent
```

Wazuh server

1. Add the following rules to the `/var/ossec/etc/rules/local_rules.xml` file.

```
<group name="syscheck,">
  <rule id="100300" level="7">
    <if_sid>550</if_sid>
    <field name="file">/tmp/yara/malware/</field>
    <description>File modified in /tmp/yara/malware/ directory.</description>
  </rule>
  <rule id="100301" level="7">
    <if_sid>554</if_sid>
    <field name="file">/tmp/yara/malware/</field>
    <description>File added to /tmp/yara/malware/ directory.</description>
  </rule>
</group>

<group name="yara,">
  <rule id="108000" level="0">
    <decoded_as>yara_decoder</decoded_as>
    <description>Yara grouping rule</description>
  </rule>
  <rule id="108001" level="14">
    <if_sid>108000</if_sid>
    <match>wazuh-yara: INFO - Scan result: </match>
    <description>File "$(yara_scanned_file)" is a positive match. Yara rule: $(yara_rule)</description>
  </rule>
</group>
```

Figure 25: Custom Rules for YARA Scanning

2. Add the following decoders to the Wazuh server `/var/ossec/etc/decoders/local_decoder.xml` file. This allows extracting the information from YARA scan results.

```
<!--added for YARA scanning-->
<decoder name="yara_decoder">
  <prematch>wazuh-yara:</prematch>
</decoder>

<decoder name="yara_decoder1">
  <parent>yara_decoder</parent>
  <regex>wazuh-yara: (\S+) - Scan result: (\S+) (\S+)</regex>
  <order>log_type, yara_rule, yara_scanned_file</order>
</decoder>
```

Figure 26: Custom Decoders for YARA Scanning

3. Add the following configuration to the Wazuh server `/var/ossec/etc/ossec.conf` configuration file. This configures the active response module to trigger after the rule 100300 and 100301 are fired.

```
<!--YARA scanning-->
<ossec_config>
  <command>
    <name>yara_linux</name>
    <executable>yara.sh</executable>
    <extra_args>-yara_path /usr/local/bin -yara_rules /tmp/yara/rules/yara_rules.yar</extra_args>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <command>yara_linux</command>
    <location>local</location>
    <rules_id>100300,100301</rules_id>
  </active-response>
</ossec_config>
```

Figure 27: Updating the Configuration for Active Response

SIMULATION

1. Create the script `/tmp/yara/malware/malware_downloader.sh` on the monitored endpoint to download malware samples:

```
#!/bin/bash
# Wazuh - Malware Downloader for test purposes
# Copyright (C) 2015-2022, Wazuh Inc.
#
# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation.

function fetch_sample(){

  curl -s -XGET "$1" -o "$2"

}

echo "WARNING: Downloading Malware samples, please use this script with
↪  caution."
read -p "  Do you want to continue? (y/n)" -n 1 -r ANSWER
echo

if [[ $ANSWER =~ ^[Yy]$ ]]
```

```
then
    echo
    # Mirai
    echo "# Mirai: https://en.wikipedia.org/wiki/Mirai_(malware)"
    echo "Downloading malware sample..."
    fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai"
    ↪  "/tmp/yara/malware/mirai" && echo "Done!" || echo "Error while
    ↪  downloading."
    echo
    # Xbash
    echo "# Xbash: https://unit42.paloaltonetworks.com/unit42-xbash-com
    ↪  bines-botnet-ransomware-coinmining-worm-targets-linux-windows/"
    echo "Downloading malware sample..."
    fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash"
    ↪  "/tmp/yara/malware/xbash" && echo "Done!" || echo "Error while
    ↪  downloading."
    echo
    # VPNFilter
    echo "# VPNFilter: https://news.sophos.com/en-us/2018/05/24/vpnfilt
    ↪  er-botnet-a-sophoslabs-analysis/"
    echo "Downloading malware sample..."
    fetch_sample
    ↪  "https://wazuh-demo.s3-us-west-1.amazonaws.com/vpn_filter"
    ↪  "/tmp/yara/malware/vpn_filter" && echo "Done!" || echo "Error
    ↪  while downloading."
    echo
    # Webshell
    echo "# WebShell: https://github.com/SecWiki/WebShell-2/blob/master
    ↪  /Php/Worse%20Linux%20Shell.php"
    echo "Downloading malware sample..."
    fetch_sample
    ↪  "https://wazuh-demo.s3-us-west-1.amazonaws.com/webshell"
    ↪  "/tmp/yara/malware/webshell" && echo "Done!" || echo "Error
    ↪  while downloading."
    echo
fi
```

2. Run the `malware_downloader.sh` script to download malware samples to the `/tmp/yara/malware` directory:

```
sudo bash /tmp/yara/malware/malware_downloader.sh
```

```
root@Sadat999-MISC /# sudo bash /tmp/yara/malware/malware_downloader.sh
WARNING: Downloading Malware samples, please use this script with  caution.
  Do you want to continue? (y/n)y

# Mirai: https://en.wikipedia.org/wiki/Mirai_(malware)
Downloading malware sample...
Done!

# Xbash: https://unit42.paloaltonetworks.com/unit42-xbash-combines-botnet-ransomware-coinmining-worm-targets-linux-windows/
Downloading malware sample...
Done!

# VPNFilter: https://news.sophos.com/en-us/2018/05/24/vpnfilter-botnet-a-sophoslabs-analysis/
Downloading malware sample...
Done!

# WebShell: https://github.com/SecWiki/WebShell-2/blob/master/Php/Worse%20Linux%20Shell.php
Downloading malware sample...
Done!
```

Figure 28: Downloading Four Malwares for YARA Scanning Simulation

DASHBOARD UPDATE

If we navigate like previously shown in 4.2.1, we will see some changes. Number of level 12 or above alerts will go up by quite a bit, because of multiple alert generation for the same malwares. To be precise, they rose by 19.

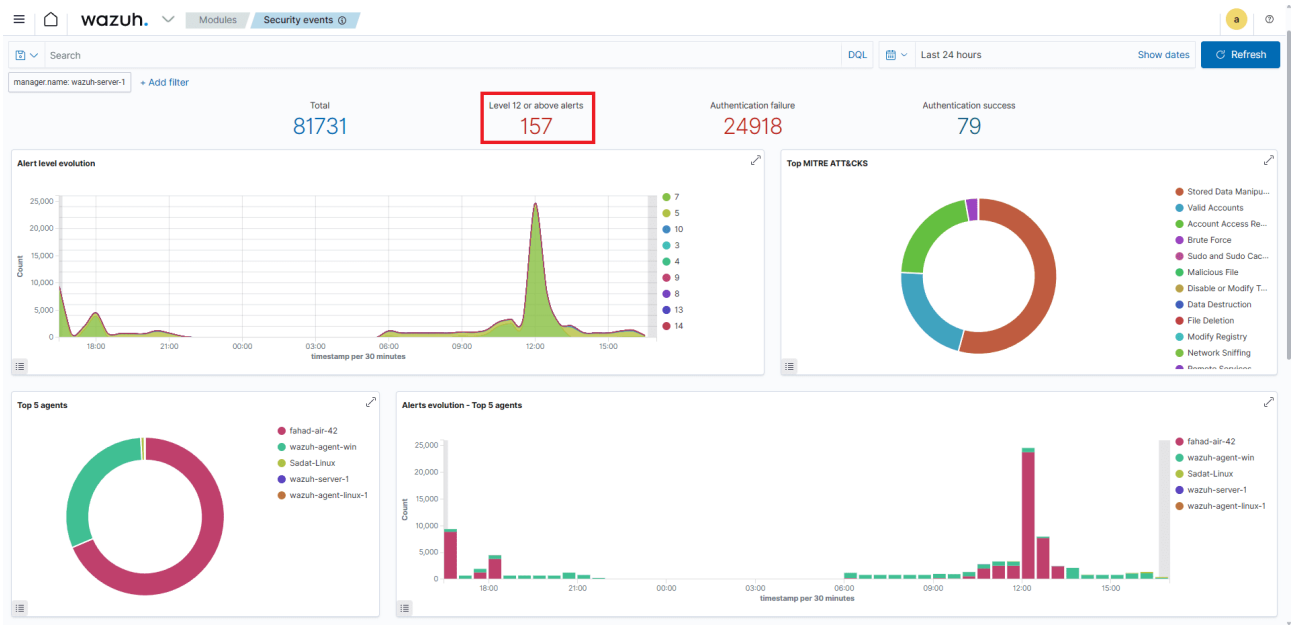
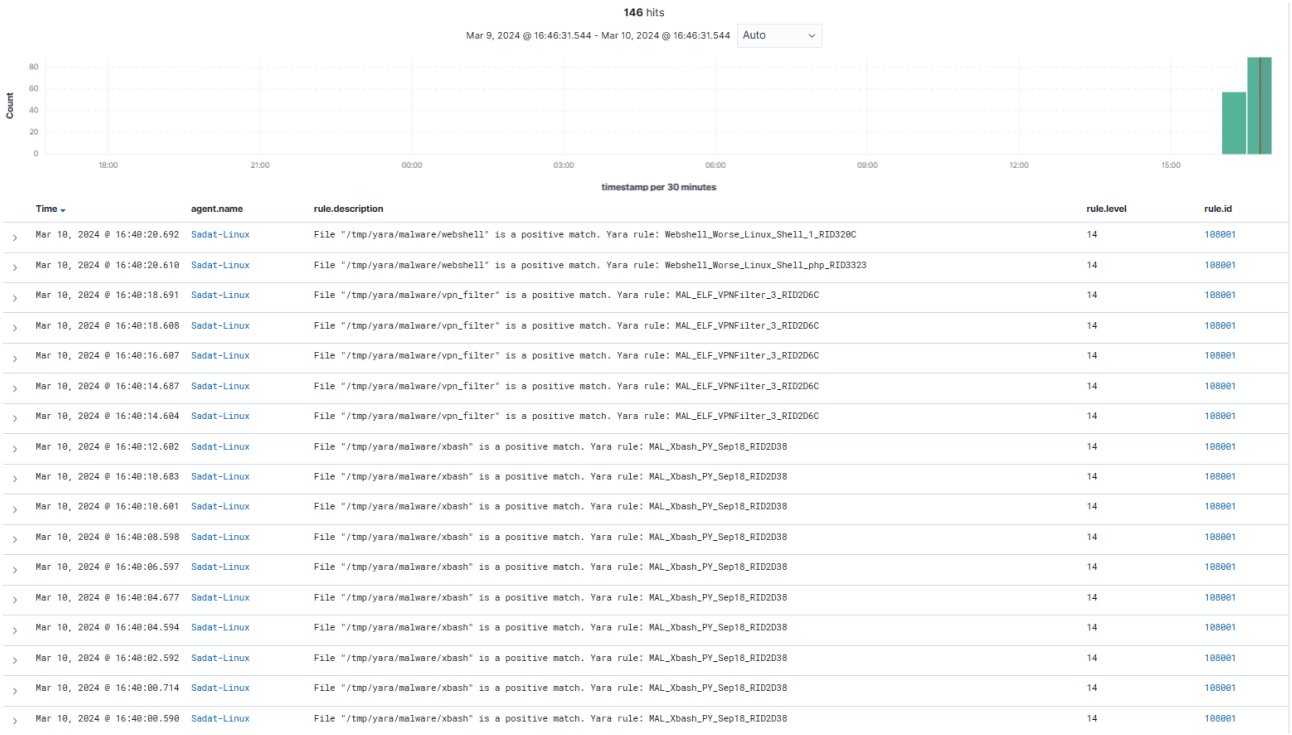


Figure 29: Dashboard after Downloading the Malwares

If we go to the Events tab, we can see the alerts better. To precisely filter out the alerts generated by YARA, we select,

```
rule.groups:yara
```

Then we can see all the generated alerts. Point to be noted here, Wazuh was able to detect all four of the malwares - Mirai, Xbash, VPNFilter and Webshell.



Time	agent.name	rule.description	rule.level	rule.id
Mar 10, 2024 @ 16:40:20.692	Sadat-Linux	File "/tmp/yara/malware/webshell" is a positive match. Yara rule: Webshell_Worse_Linux_Shell_1_RID320C	14	100001
Mar 10, 2024 @ 16:40:20.610	Sadat-Linux	File "/tmp/yara/malware/webshell" is a positive match. Yara rule: Webshell_Worse_Linux_Shell_php_RID3323	14	100001
Mar 10, 2024 @ 16:40:18.691	Sadat-Linux	File "/tmp/yara/malware/vpn_filter" is a positive match. Yara rule: MAL_ELF_VPNFilter_3_RID2D6C	14	100001
Mar 10, 2024 @ 16:40:18.608	Sadat-Linux	File "/tmp/yara/malware/vpn_filter" is a positive match. Yara rule: MAL_ELF_VPNFilter_3_RID2D6C	14	100001
Mar 10, 2024 @ 16:40:16.607	Sadat-Linux	File "/tmp/yara/malware/vpn_filter" is a positive match. Yara rule: MAL_ELF_VPNFilter_3_RID2D6C	14	100001
Mar 10, 2024 @ 16:40:14.687	Sadat-Linux	File "/tmp/yara/malware/vpn_filter" is a positive match. Yara rule: MAL_ELF_VPNFilter_3_RID2D6C	14	100001
Mar 10, 2024 @ 16:40:14.604	Sadat-Linux	File "/tmp/yara/malware/vpn_filter" is a positive match. Yara rule: MAL_ELF_VPNFilter_3_RID2D6C	14	100001
Mar 10, 2024 @ 16:40:12.602	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:10.683	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:10.601	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:08.598	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:06.597	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:04.677	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:04.594	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:02.592	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:00.714	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001
Mar 10, 2024 @ 16:40:00.590	Sadat-Linux	File "/tmp/yara/malware/xbash" is a positive match. Yara rule: MAL_Xbash_PY_Sep18_RID2D38	14	100001

Figure 30: Generated Alerts after Downloading Four Malwares

4.2.3 VIRUSTOTAL INTEGRATION

VirusTotal is an online service that analyzes files and URLs to detect viruses, worms, trojans, and other malicious content using antivirus engines and website scanners. Since VirusTotal stores all the analyses it performs, users can search for file hashes. VirusTotal also provides an API that allows access to the information generated by VirusTotal without needing to utilize the HTML website interface.

HOW IT WORKS

This integration leverages the VirusTotal API to identify malicious content in files and directories monitored by the File Integrity Monitoring (FIM) feature of Wazuh. The workflow is outlined as follows:

1. The FIM module in Wazuh monitors for any additions, changes, or deletions in the monitored directories, generating alerts for any detected modifications.

2. Upon detecting a modification, and if the VirusTotal integration is enabled, Wazuh triggers this integration based on the FIM alert. This involves extracting the file's hash and initiating a VirusTotal scan.

3. The integration executes an HTTP POST request to the VirusTotal database via the VirusTotal API, submitting the file hash for comparison against the VirusTotal database records.

4. Upon receiving a JSON response from VirusTotal, the integration triggers one of the following types of Wazuh alerts based on the response content:

   - **Error: Check credentials**


```
{
   "timestamp":"2022-11-17T19:17:43.637+0200",
   "rule":{
      "level":3,
      "description":"VirusTotal: Error: Check credentials",
      "id":"87102",
      "firedtimes":3,
      "mail":false,
      "groups":[
         "virustotal"
      ],
      "gdpr":[
         "IV_35.7.d",
         "IV_32.2"
      ]
   },
   "agent":{
      "id":"000",
      "name":"localhost.localdomain"
   },
   "manager":{
      "name":"localhost.localdomain"
   },
   "id":"1668705463.51155",
   "decoder":{
```

```
      "name":"json"
   },
   "data":{
      "virustotal":{
         "error":"403",
         "description":"Error: Check credentials"
      },
      "integration":"virustotal"
   },
   "location":"virustotal"
}
```


- **Error: Public API request rate limit reached**


```
{
   "timestamp":"2022-11-17T19:22:13.236+0200",
   "rule":{
      "level":3,
      "description":"VirusTotal: Error: Public API request rate
      ↪  limit reached",
      "id":"87101",
      "firedtimes":2,
      "mail":false,
      "groups":[
         "virustotal"
      ]
   },
   "agent":{
      "id":"000",
      "name":"localhost.localdomain"
   },
   "manager":{
      "name":"localhost.localdomain"
   },
   "id":"1668705733.90632",
   "decoder":{
```

```
      "name":"json"
   },
   "data":{
      "virustotal":{
         "error":"204",
         "description":"Error: Public API request rate limit
          ↪  reached"
      },
      "integration":"virustotal"
   },
   "location":"virustotal"
}
```


- **Alert: No positives found**


```
{
   "timestamp":"2022-11-17T19:22:07.974+0200",
   "rule":{
      "level":3,
      "description":"VirusTotal: Alert -
       ↪  /media/user/software/suspicious-file10.exe \
      - No positives found",
      "id":"87104",
      "firedtimes":4,
      "mail":false,
      "groups":[
         "virustotal"
      ]
   },
   "agent":{
      "id":"010",
      "name":"Ubuntu",
      "ip":"10.0.2.15"
   },
   "manager":{
      "name":"localhost.localdomain"
```

```
   },
   "id":"1668705727.84464",
   "decoder":{
      "name":"json"
   },
   "data":{
      "virustotal":{
         "found":"1",
         "malicious":"0",
         "source":{
            "alert_id":"1668705721.82254",
            "file":"/media/user/software/suspicious-file10.exe",
            "md5":"d41d8cd98f00b204e9800998ecf8427e",
            "sha1":"da39a3ee5e6b4b0d3255bfef95601890afd80709"
         },
         "sha1":"da39a3ee5e6b4b0d3255bfef95601890afd80709",
         "scan_date":"2022-11-17 17:19:48",
         "positives":"0",
         "total":"60",
         "permalink":"https://www.virustotal.com/gui/file/e3b0c4429
         ↪ 8fc1c149afbf\
         4c8996fb92427ae41e4649b934ca495991b7852b855/detection/f-e3
         ↪ b0c44298fc1c\
         149afbf4c8996fb92427ae41e4649b934ca495991b7852b855-1668705
         ↪ 588"
      },
      "integration":"virustotal"
   },
   "location":"virustotal"
}
```


- **Alert: X engines detected this file** Here, X represents the number of antivirus engines that flagged the file.


```
{
   "timestamp":"2022-11-17T19:30:25.085+0200",
```

```
"rule":{
   "level":12,
   "description":"VirusTotal: Alert -
   ↪  /media/user/software/eicar.com - 66 engines detected this
   ↪  file",
   "id":"87105",
   "mitre":{
      "id":[
         "T1203"
      ],
      "tactic":[
         "Execution"
      ],
      "technique":[
         "Exploitation for Client Execution"
      ]
   },
   "firedtimes":1,
   "mail":true,
   "groups":[
      "virustotal"
   ],
   "pci_dss":[
      "10.6.1",
      "11.4"
   ],
   "gdpr":[
      "IV_35.7.d"
   ]
},
"agent":{
   "id":"010",
   "name":"Ubuntu",
   "ip":"10.0.2.15"
},
"manager":{
```

```
      "name":"localhost.localdomain"
   },
   "id":"1668706225.104492",
   "decoder":{
      "name":"json"
   },
   "data":{
      "virustotal":{
         "found":"1",
         "malicious":"1",
         "source":{
            "alert_id":"1668706222.103798",
            "file":"/media/user/software/eicar.com",
            "md5":"44d88612fea8a8f36de82e1278abb02f",
            "sha1":"3395856ce81f2b7382dee72602f798b642f14140"
         },
         "sha1":"3395856ce81f2b7382dee72602f798b642f14140",
         "scan_date":"2022-11-17 17:15:04",
         "positives":"66",
         "total":"68",
         "permalink":"https://www.virustotal.com/gui/file/275a021bb\
         fb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f\
         /detection/f-275a021bbfb6489e54d471899f7db9d1663fc695ec2fe
          ↪  2a2c4538aabf651fd0f-1668705304"
      },
      "integration":"virustotal"
   },
   "location":"virustotal"
}
```


CONFIGURATION

Linux endpoint

1. Add the following to the `<syscheck>` section of the configuration file. We reuse the same folder `/fim` as previously used in 4.2.1. If that configuration is already done, the following no more needs to be added.

```
<syscheck>
  <directories check_all="yes" realtime="yes">/fim</directories>
</syscheck>
```

2. Restart the Wazuh agent.

```
systemctl restart wazuh-agent
```

Wazuh server

1. Add the following to the `/var/ossec/etc/ossec.conf` file on the Wazuh server:

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

Figure 31: Configuration for VirusTotal Integration

SIMULATION

1. Download a malicious file on the endpoint in the monitored folder.

```
sudo curl -Lo /fim/suspicious-file.exe
↪  https://secure.eicar.org/eicar.com
```

```
root@Sadat999-MISC /h/Sadat999# sudo curl -Lo /fim/suspicious-file.exe https://secure.eicar.org/eicar.com
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    68  100    68    0     0    155      0 --:--:-- --:--:-- --:--:--   154
```

Figure 32: Downloading Malware for VirusTotal Checking

DASHBOARD UPDATE

We will again navigate to "Security events" tab as shown previously in 4.2.1. There we will see a new level 12 alert has been generated because of the malware download (the count was previously 179).

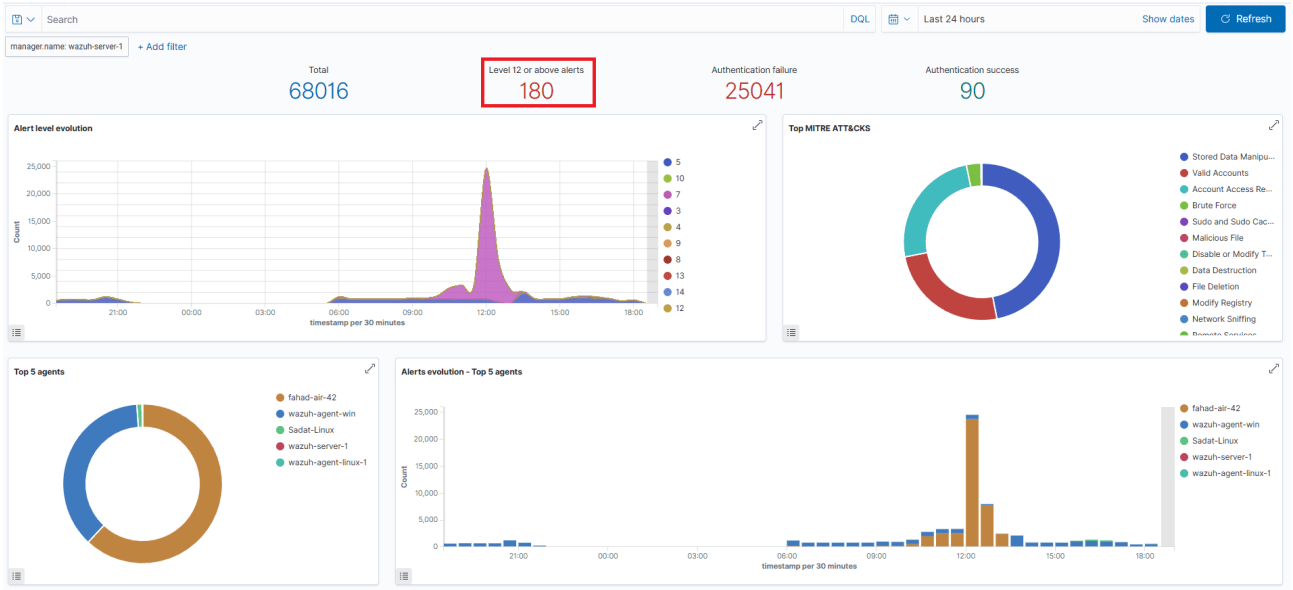


Figure 33: Dashboard after Malware Download

The generated alert goes as follows:

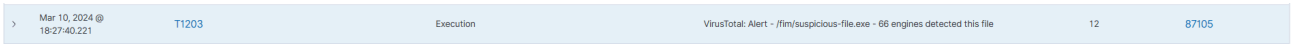


Figure 34: Alert Generated by VirusTotal API Check

If we examine the JSON body of the alert, we can see that 66 engines or anti-virus softwares, out of 68, flagged our downloaded file as a malware.


```
{
  "agent": {
    "ip": "10.0.0.4",
    "name": "Sadat-Linux",
    "id": "008"
  },
  "manager": {
    "name": "wazuh-server-1"
  },
```

```
"data": {
  "integration": "virustotal",
  "virustotal": {
    "sha1": "3395856ce81f2b7382dee72602f798b642f14140",
    "malicious": "1",
    "total": "68",
    "found": "1",
    "positives": "66",
    "source": {
      "sha1": "3395856ce81f2b7382dee72602f798b642f14140",
      "file": "/fim/suspicious-file.exe",
      "alert_id": "1710073657.200258194",
      "md5": "44d88612fea8a8f36de82e1278abb02f"
    },
    "permalink":
    ↪  "https://www.virustotal.com/gui/file/275a021bbfb6489e54d471899f7db
    ↪  9d1663fc695ec2fe2a2c4538aabf651fd0f/detection/f-275a021bbfb6489e54
    ↪  d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0f-1710073285",
    "scan_date": "2024-03-10 12:21:25"
  }
},
"rule": {
  "firedtimes": 1,
  "mail": true,
  "level": 12,
  "pci_dss": [
    "10.6.1",
    "11.4"
  ],
  "description": "VirusTotal: Alert - /fim/suspicious-file.exe - 66 engines
  ↪  detected this file",
  "groups": [
    "virustotal"
  ],
  "mitre": {
    "technique": [
```

```
        "Exploitation for Client Execution"
      ],
      "id": [
        "T1203"
      ],
      "tactic": [
        "Execution"
      ]
    },
    "id": "87105",
    "gdpr": [
      "IV_35.7.d"
    ]
  },
  "decoder": {
    "name": "json"
  },
  "input": {
    "type": "log"
  },
  "@timestamp": "2024-03-10T12:27:40.221Z",
  "location": "virustotal",
  "id": "1710073660.200290701",
  "timestamp": "2024-03-10T12:27:40.221+0000",
  "_id": "6JBVKI4B8KVEhOwaUUG9"
}
```


4.2.4 WINDOWS DEFENDER LOGS COLLECTION

Windows Defender is the anti-malware component of the Microsoft Windows operating system. Wazuh agents installed on Windows endpoints can be configured to collect Windows Defender logs. This provides visibility on malware infections detected by Windows Defender on Windows endpoints. These logs can provide information about:

- The status of the Windows Defender service.
- Results of Windows Defender scans that the users run on these endpoints.

HOW IT WORKS

Wazuh has out-of-the-box decoders for Microsoft Windows logs including Windows Defender. Rules are also included specifically for Windows Defender, which can be found at `/var/ossec/ruleset/rules/0600-win-wdefender_rules.xml` on the Wazuh server. Below are examples of Windows Defender alerts, which are triggered by user and malware activity.

- **Windows Defender detects malware**


```
{
   "timestamp":"2023-01-05T11:44:58.557+0200",
   "rule":{
      "level":12,
      "description":"Windows Defender: Antimalware platform detected
      ↪  potentially unwanted software ()",
      "id":"62123",
      "firedtimes":2,
      "mail":true,
      "groups":[
         "windows",
         "windows_defender"
      ],
      "pci_dss":[
         "5.1",
         "5.2",
         "10.6.1",
         "11.4"
      ],
      "gpg13":[
         "4.2"
      ],
      "gdpr":[
         "IV_35.7.d"
      ],
      "hipaa":[
         "164.312.b"
      ],
      "nist_800_53":[
```

```
            "SI.3",
            "AU.6",
            "SI.4"
        ],
        "tsc":[
            "A1.2",
            "CC7.2",
            "CC7.3",
            "CC6.1",
            "CC6.8"
        ]
    },
    "agent":{
        "id":"012",
        "name":"Windows_11",
        "ip":"10.0.2.15"
    },
    "manager":{
        "name":"localhost.localdomain"
    },
    "id":"1672911898.1113167",
    "decoder":{
        "name":"windows_eventchannel"
    },
    "data":{
        "win":{
            "system":{
                "providerName":"Microsoft-Windows-Windows Defender",
                "providerGuid":"{11cd958a-c507-4ef3-b3f2-5fd9dfbd2c78}",
                "eventID":"1116",
                "version":"0",
                "level":"3",
                "task":"0",
                "opcode":"0",
                "keywords":"0x8000000000000000",
                "systemTime":"2023-01-05T09:44:55.1124563Z",
```

```
            "eventRecordID":"525",
            "processID":"2600",
            "threadID":"432",
            "channel":"Microsoft-Windows-Windows Defender/Operational",
            "computer":"Windows-11",
            "severityValue":"WARNING",
            "message":"\"Microsoft Defender Antivirus has detected
            ↪  malware or other potentially unwanted software.\r\n For
            ↪  more information please see the following:\r\nhttps://g
            ↪  o.microsoft.com/fwlink/?linkid=37020&name=Virus:DOS/EIC
            ↪  AR_Test_File&threatid=2147519003&enterprise=0\r\n
            ↪  \tName: Virus:DOS/EICAR_Test_File\r\n \tID:
            ↪  2147519003\r\n \tSeverity: Severe\r\n \tCategory:
            ↪  Virus\r\n \tPath: file:_C:\\Users\\win11\\AppData\\Loca
            ↪  l\\Temp\\36f9c971-77e5-4f5e-bbef-f7162522dee1.tmp;
            ↪  webfile:_C:\\Users\\win11\\AppData\\Local\\Temp\\36f9c9
            ↪  71-77e5-4f5e-bbef-f7162522dee1.tmp|https://secure.eicar
            ↪  .org/eicar.com.txt|pid:8412,ProcessStart:13317385493924
            ↪  0064\r\n \tDetection Origin: Internet\r\n \tDetection
            ↪  Type: Concrete\r\n \tDetection Source: Downloads and
            ↪  attachments\r\n \tUser: Windows-11\\win11\r\n \tProcess
            ↪  Name: Unknown\r\n \tSecurity intelligence Version: AV:
            ↪  1.381.1755.0, \
           AS: 1.381.1755.0, NIS: 1.381.1755.0\r\n \tEngine Version:
            ↪  AM: 1.1.19900.2, NIS: 1.1.19900.2\""
         },
         "eventdata":{
            "product Name":"Microsoft Defender Antivirus",
            "product Version":"4.18.2211.5",
            "detection ID":"{53737EEC-A8A6-45E0-9155-4566B8133573}",
            "detection Time":"2023-01-05T09:44:55.064Z",
            "threat ID":"2147519003",
            "threat Name":"Virus:DOS/EICAR_Test_File",
            "severity ID":"5",
            "severity Name":"Severe",
            "category ID":"42",
```

```
            "category Name":"Virus",
            "fWLink":"https://go.microsoft.com/fwlink/?linkid=37020&amp
            ↪  ;name=Virus:DOS/EICAR_Test_File&amp;threatid=2147519003
            ↪  &amp;enterprise=0",
            "status Code":"1",
            "state":"1",
            "source ID":"4",
            "source Name":"Downloads and attachments",
            "process Name":"Unknown",
            "detection User":"Windows-11\\\\win11",
            "path":"file:_C:\\\\Users\\\\win11\\\\AppData\\\\Local\\\\T
            ↪  emp\\\\36f9c971-77e5-4f5e-bbef-f7162522dee1.tmp;
            ↪  webfile:_C:\\\\Users\\\\win11\\\\AppData\\\\Local\\\\Te
            ↪  mp\\\\36f9c971-77e5-4f5e-bbef-f7162522dee1.tmp|https://
            ↪  secure.eicar.org/eicar.com.txt|pid:8412,ProcessStart:13
            ↪  3173854939240064",
            "origin ID":"4",
            "origin Name":"Internet",
            "execution ID":"0",
            "execution Name":"Unknown",
            "type ID":"0",
            "type Name":"Concrete",
            "pre Execution Status":"0",
            "action ID":"9",
            "action Name":"Not Applicable",
            "error Code":"0x00000000",
            "error Description":"The operation completed successfully.",
            "post Clean Status":"0",
            "additional Actions ID":"0",
            "additional Actions String":"No additional actions
            ↪  required",
            "security intelligence Version":"AV: 1.381.1755.0, AS:
            ↪  1.381.1755.0, NIS: 1.381.1755.0",
            "engine Version":"AM: 1.1.19900.2, NIS: 1.1.19900.2"
        }
    }
```

```
   },
   "location":"EventChannel"
}
```


- **Windows Defender responds to detected malware**


```
{
   "timestamp":"2023-01-05T11:45:06.032+0200",
   "rule":{
      "level":3,
      "description":"Windows Defender: Antimalware platform performed an
      ↪  action to protect you from potentially unwanted software ()",
      "id":"62124",
      "firedtimes":2,
      "mail":false,
      "groups":[
         "windows",
         "windows_defender"
      ],
      "pci_dss":[
         "5.1",
         "5.2",
         "10.6.1",
         "11.4"
      ],
      "gpg13":[
         "4.2"
      ],
      "gdpr":[
         "IV_35.7.d"
      ],
      "hipaa":[
         "164.312.b"
      ],
      "nist_800_53":[
         "SI.3",
```

```
            "AU.6",
            "SI.4"
        ],
        "tsc":[
            "A1.2",
            "CC7.2",
            "CC7.3",
            "CC6.1",
            "CC6.8"
        ]
    },
    "agent":{
        "id":"012",
        "name":"Windows_11",
        "ip":"10.0.2.15"
    },
    "manager":{
        "name":"localhost.localdomain"
    },
    "id":"1672911906.1119694",
    "decoder":{
        "name":"windows_eventchannel"
    },
    "data":{
        "win":{
            "system":{
                "providerName":"Microsoft-Windows-Windows Defender",
                "providerGuid":"{11cd958a-c507-4ef3-b3f2-5fd9dfbd2c78}",
                "eventID":"1117",
                "version":"0",
                "level":"4",
                "task":"0",
                "opcode":"0",
                "keywords":"0x8000000000000000",
                "systemTime":"2023-01-05T09:45:02.6103899Z",
                "eventRecordID":"526",
```

```
            "processID":"2600",
            "threadID":"432",
            "channel":"Microsoft-Windows-Windows Defender/Operational",
            "computer":"Windows-11",
            "severityValue":"INFORMATION",
            "message": "Microsoft Defender Antivirus has taken action to
             ↪  protect this machine from malware or other potentially
             ↪  unwanted software. For more information please see the
             ↪  following: \
            https://go.microsoft.com/fwlink/?linkid=37020&name=Virus:DO
             ↪  S/EICAR_Test_File&threatid=2147519003&enterprise=0
             ↪  \tName: Virus:DOS/EICAR_Test_File \
            \tID: 2147519003 \
            \tSeverity: Severe \
            \tCategory: Virus \
            \tPath: file:_C:\\Users\\win11\\AppData\\Local\\Temp\\36f9c
             ↪  971-77e5-4f5e-bbef-f7162522dee1.tmp;
             ↪  \
            webfile:_C:\\Users\\win11\\AppData\\Local\\Temp\\36f9c971-7
             ↪  7e5-4f5e-bbef-f7162522dee1.tmp|https://secure.eicar.org
             ↪  /eicar.com.txt|pid:8412,ProcessStart:133173854939240064
             ↪  \
            \tDetection Origin: Internet \
            \tDetection Type: Concrete \
            \tDetection Source: Downloads and attachments \
            \tUser: NT AUTHORITY\\SYSTEM \
            \tProcess Name: Unknown \
            \tAction: Quarantine \
            \tAction Status: No additional actions required \
            \tError Code: 0x00000000 \
            \tError description: The operation completed successfully. \
            \tSecurity intelligence Version: AV: 1.381.1755.0, AS:
             ↪  1.381.1755.0, NIS: 1.381.1755.0 \
            \tEngine Version: AM: 1.1.19900.2, NIS: 1.1.19900.2"
        },
        "eventdata":{
```

```
"product Name":"Microsoft Defender Antivirus",
"product Version":"4.18.2211.5",
"detection ID":"{53737EEC-A8A6-45E0-9155-4566B8133573}",
"detection Time":"2023-01-05T09:44:55.064Z",
"threat ID":"2147519003",
"threat Name":"Virus:DOS/EICAR_Test_File",
"severity ID":"5",
"severity Name":"Severe",
"category ID":"42",
"category Name":"Virus",
"fWLink":"https://go.microsoft.com/fwlink/?linkid=37020&amp
↪  ;name=Virus:DOS/EICAR_Test_File&amp;threatid=2147519003
↪  &amp;enterprise=0",
"status Code":"4",
"state":"2",
"source ID":"4",
"source Name":"Downloads and attachments",
"process Name":"Unknown",
"detection User":"Windows-11\\\\win11",
"origin ID":"4",
"origin Name":"Internet",
"execution ID":"0",
"execution Name":"Unknown",
"path":"file:_C:\\\\Users\\\\win11\\\\AppData\\\\Local\\\\T
↪  emp\\\\36f9c971-77e5-4f5e-bbef-f7162522dee1.tmp;
↪  webfile:_C:\\\\Users\\\\win11\\\\AppData\\\\Local\\\\Te
↪  mp\\\\36f9c971-77e5-4f5e-bbef-f7162522dee1.tmp|https://
↪  secure.eicar.org/eicar.com.txt|pid:8412,ProcessStart:13
↪  3173854939240064",
"type ID":"0",
"type Name":"Concrete",
"pre Execution Status":"0",
"action ID":"2",
"action Name":"Quarantine",
"error Code":"0x00000000",
"error Description":"The operation completed successfully.",
```

```
            "post Clean Status":"0",
            "additional Actions ID":"0",
            "additional Actions String":"No additional actions
            ↪  required",
            "remediation User":"NT AUTHORITY\\\\SYSTEM",
            "security intelligence Version":"AV: 1.381.1755.0, AS:
            ↪  1.381.1755.0, NIS: 1.381.1755.0",
            "engine Version":"AM: 1.1.19900.2, NIS: 1.1.19900.2"
         }
      }
   },
   "location":"EventChannel"
}
```


- **Windows Defender protection is disabled**


```
{
   "timestamp":"2023-01-05T16:26:55.513+0200",
   "rule":{
      "level":5,
      "description":"Windows Defender: Antivirus real-time protection is
      ↪  disabled",
      "id":"62152",
      "firedtimes":1,
      "mail":false,
      "groups":[
         "windows",
         "windows_defender"
      ],
      "pci_dss":[
         "5.1",
         "10.2.6",
         "10.6.1"
      ],
      "gpg13":[
         "4.14",
```

```
            "10.1"
        ],
        "gdpr":[
            "IV_35.7.d"
        ],
        "hipaa":[
            "164.312.b"
        ],
        "nist_800_53":[
            "SI.3",
            "AU.14",
            "AU.5",
            "AU.6"
        ],
        "tsc":[
            "A1.2",
            "CC6.8",
            "CC7.2",
            "CC7.3"
        ]
    },
    "agent":{
        "id":"012",
        "name":"Windows_11",
        "ip":"10.0.2.15"
    },
    "manager":{
        "name":"localhost.localdomain"
    },
    "id":"1672928815.1914866",
    "decoder":{
        "name":"windows_eventchannel"
    },
    "data":{
        "win":{
            "system":{
```

```
            "providerName":"Microsoft-Windows-Windows Defender",
            "providerGuid":"{11cd958a-c507-4ef3-b3f2-5fd9dfbd2c78}",
            "eventID":"5001",
            "version":"0",
            "level":"4",
            "task":"0",
            "opcode":"0",
            "keywords":"0x8000000000000000",
            "systemTime":"2023-01-05T14:33:13.3093446Z",
            "eventRecordID":"540",
            "processID":"2600",
            "threadID":"7152",
            "channel":"Microsoft-Windows-Windows Defender/Operational",
            "computer":"Windows-11",
            "severityValue":"INFORMATION",
            "message":"\"Microsoft Defender Antivirus Real-time
             ↪  Protection scanning for malware and other potentially
             ↪  unwanted software was disabled.\""
         },
         "eventdata":{
            "product Name":"Microsoft Defender Antivirus",
            "product Version":"4.18.2211.5"
         }
      }
   },
   "location":"EventChannel"
}
```


- **Windows Defender updates its signature database**


```
{
   "timestamp":"2023-01-05T12:55:10.920+0200",
   "rule":{
      "level":3,
      "description":"Windows Defender: Antimalware definitions updated
       ↪  successfully",
```

```
      "id":"62130",
      "firedtimes":2,
      "mail":false,
      "groups":[
         "windows",
         "windows_defender"
      ],
      "pci_dss":[
         "5.1",
         "10.6.1",
         "5.2"
      ],
      "gdpr":[
         "IV_35.7.d",
         "IV_35.7.d"
      ],
      "gpg13":[
         "4.4",
         "4.14"
      ],
      "hipaa":[
         "164.312.b"
      ],
      "nist_800_53":[
         "SI.3",
         "AU.6"
      ],
      "tsc":[
         "A1.2",
         "CC7.2",
         "CC7.3"
      ]
   },
   "agent":{
      "id":"011",
      "name":"ONEBOT-1",
```

```
    "ip":"10.5.0.2"
},
"manager":{
    "name":"localhost.localdomain"
},
"id":"1672916110.1441972",
"decoder":{
    "name":"windows_eventchannel"
},
"data":{
    "win":{
        "system":{
            "providerName":"Microsoft-Windows-Windows Defender",
            "providerGuid":"{11cd958a-c507-4ef3-b3f2-5fd9dfbd2c78}",
            "eventID":"2000",
            "version":"0",
            "level":"4",
            "task":"0",
            "opcode":"0",
            "keywords":"0x8000000000000000",
            "systemTime":"2023-01-05T10:55:07.4095656Z",
            "eventRecordID":"649",
            "processID":"6716",
            "threadID":"7528",
            "channel":"Microsoft-Windows-Windows Defender/Operational",
            "computer":"ONEBOT-1",
            "severityValue":"INFORMATION",
            "message":"\"Microsoft Defender Antivirus security
            ↪  intelligence version updated.\r\n \tCurrent security
            ↪  intelligence Version: 1.381.1755.0\r\n \tPrevious
            ↪  security intelligence Version: 1.381.1746.0\r\n
            ↪  \tSecurity intelligence Type: AntiSpyware\r\n \tUpdate
            ↪  Type: Delta\r\n \tUser: NT AUTHORITY\\SYSTEM\r\n
            ↪  \tCurrent Engine Version: 1.1.19900.2\r\n \tPrevious
            ↪  Engine Version: 1.1.19900.2\""
        },
```

```
        "eventdata":{
            "product Name":"Microsoft Defender Antivirus",
            "product Version":"4.18.2211.5",
            "current security intelligence Version":"1.381.1755.0",
            "previous security intelligence Version":"1.381.1746.0",
            "domain":"NT AUTHORITY",
            "user":"SYSTEM",
            "sID":"S-1-5-18",
            "security intelligence Type Index":"2",
            "security intelligence Type":"AntiSpyware",
            "update Type Index":"2",
            "update Type":"Delta",
            "current Engine Version":"1.1.19900.2",
            "previous Engine Version":"1.1.19900.2"
        }
    }
  },
  "location":"EventChannel"
}
```


CONFIGURATION

1. To collect Windows Defender logs, either the centralized configuration can be updated, or locally the agent `C:\Program Files (x86)\ossec-agent\ossec.conf` file. Centralized configuration allows the instructions to be shared with a group of agents. We first navigate to the local configuration file.

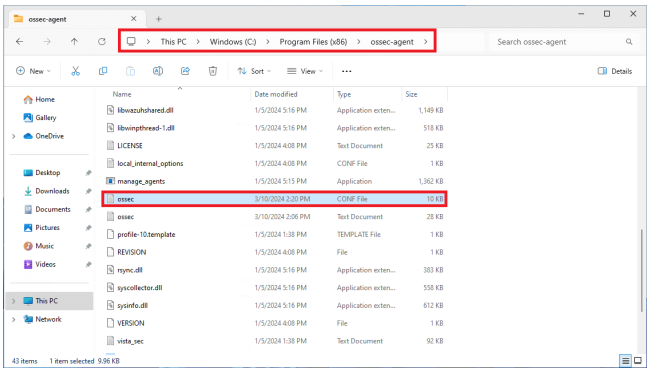


Figure 35: Navigation for Windows Local Configuration File

We then add the following block to the `ossec.conf` file.

```
<localfile>
  <location>Microsoft-Windows-Windows Defender/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>
```

2. As always, we need to restart the Wazuh agent.

```
NET STOP WazuhSvc
NET START WazuhSvc
```

SIMULATION

1. We first run a quick scan to assign Windows Defender some work.

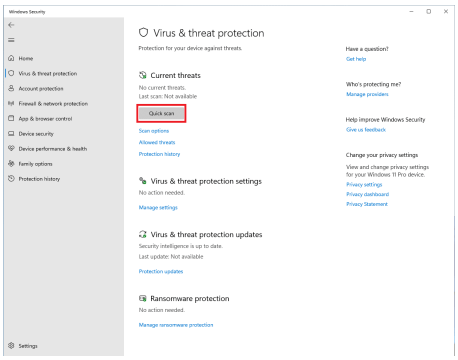


Figure 36: Initiating a Quick Scan on Windows Defender

2. After the scan finished, we disable the Real-time protection to trigger an alert.

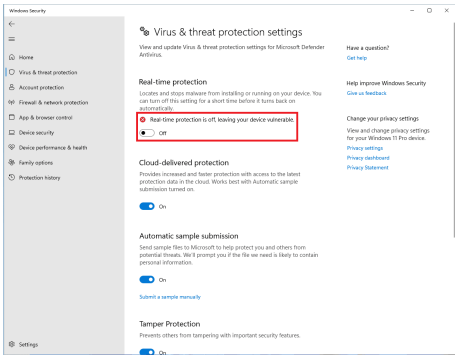


Figure 37: Disabling Realtime Protection

3. Lastly, we turn the defender back on and run the following command to download the malware "eicar".

```
Invoke-WebRequest -Uri https://secure.eicar.org/eicar_com.zip -OutFile
↪  eicar.zip
```

Expectedly, it gets instantly deleted by Windows Defender, but we are more curious to see if anything shows up on Wazuh dashboard.

DASHBOARD UPDATE

Number of level 12 or above alerts go up by 2 because of the malware download.

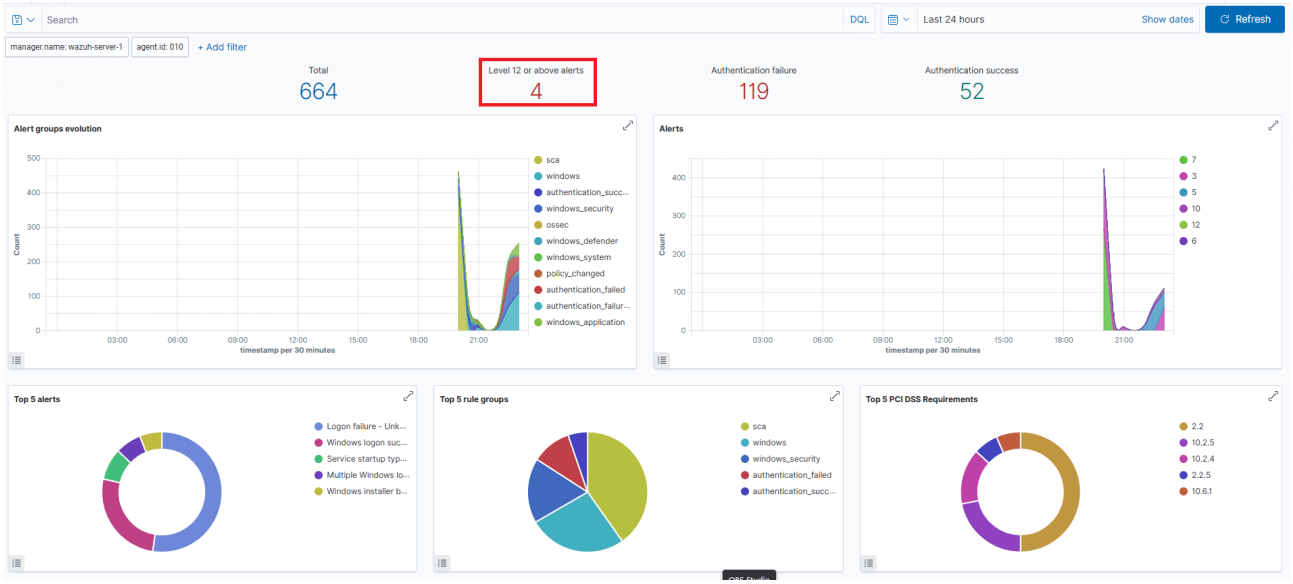


Figure 38: Dashboard after Security Events on Windows Client

We can also see all the alerts generated because of our triggering events. The alerts are generated, as previously described in Defender Disabled, Malware Detected and Malware Response.

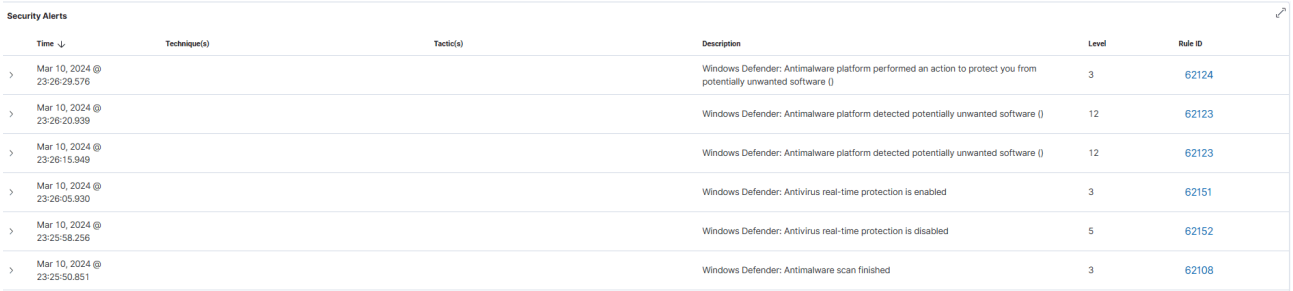


Figure 39: Generated Alerts from Windows Defender

4.3 SECURITY CONFIGURATION ASSESSMENT

Security Configuration Assessment (SCA) is the process of verifying that all systems conform to a set of predefined rules regarding configuration settings and approved application usage. One of the most certain ways to secure endpoints is by reducing their vulnerability surface. This process is commonly known as hardening. Configuration assessment is an effective way to identify weaknesses in your endpoints and patch them to reduce your attack surface.

The Wazuh SCA module performs scans to detect misconfigurations and exposures on monitored endpoints and recommend remediation actions. Those scans assess the configuration of the endpoints using policy files that contain rules to be tested against the actual configuration of the endpoint. SCA policies can check for the existence of files, directories, registry keys and values, running processes, and recursively test for the existence of files inside directories.

For example, the SCA module could assess whether it is necessary to change password-related configuration, remove unnecessary software, disable unnecessary services, or audit the TCP/IP stack configuration.

Policies for the SCA module are written in YAML. This format was chosen because it is human-readable and easy to understand. You can easily write your own SCA policies or extend existing ones to fit your needs. Furthermore, Wazuh is distributed with a set of out-of-the-box policies mostly based on the CIS benchmarks, a well-established standard for endpoint hardening.

4.3.1 HOW IT WORKS

Each Wazuh agent has its own local database where it stores the current state of each SCA check. The Wazuh server maintains an SCA database for all agents that are enrolled to it. Wazuh agents only send the differences detected between scans to the Wazuh server. If there has been no change, only a summary of the SCA scan is sent, thus avoiding unnecessary network traffic while keeping the SCA database on the Wazuh server up to date. The Wazuh server then uses those updates to issue alerts that are shown in the Wazuh dashboard.

Integrity and alerting flow are depicted in the sequence diagram below.

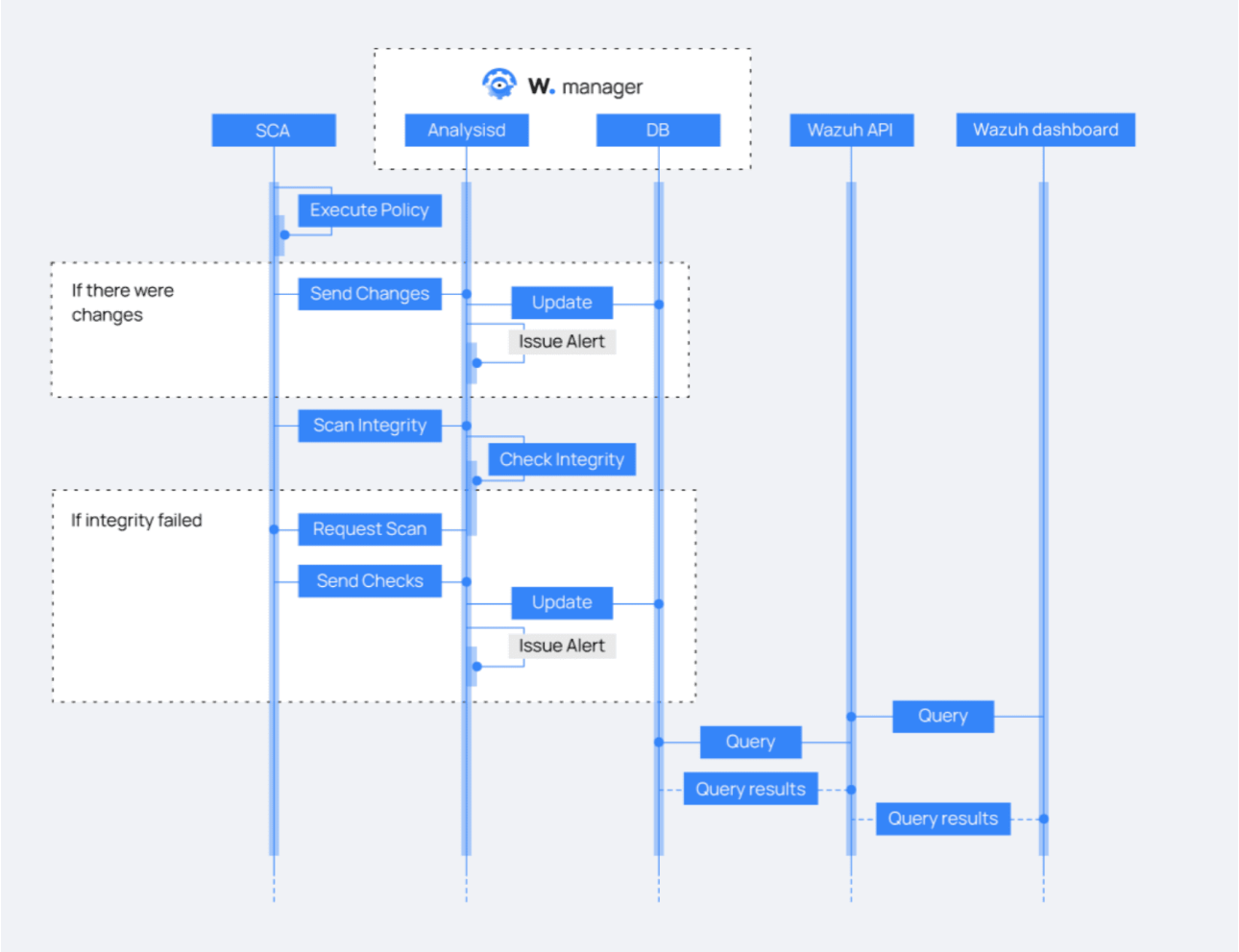


Figure 40: Integrity and Alerting Flow Diagram

OVERVIEW OF AN SCA CHECK

Checks are the core of an SCA policy, as they describe the scan to be performed in the endpoint. The checks contain fields that define what actions the agent should take to scan the endpoint, and how to evaluate the scan results. Each check definition comprises:

- Metadata information including a rationale, remediation, and a description of the check.
- A logical description with the `condition` and `rules` fields.

As part of the metadata, the SCA policy can contain an optional compliance field used to specify if the check is relevant to any compliance specifications. SCA checks usually indicate standards or policies that they aim to comply with. For example, we map CIS benchmark, PCI-DSS, NIST, and TSC controls to the relevant SCA checks.

See below SCA policy ID `2651` for Debian 10 operating system as an example of a policy definition.

```
- id: 2651
    title: "Ensure SSH HostbasedAuthentication is disabled"
    description: "The HostbasedAuthentication parameter specifies if
     ↪  authentication is allowed through trusted hosts via the user of
     ↪  .rhosts, or /etc/hosts.equiv, along with successful public key
     ↪  client host authentication. This option only applies to SSH Protocol
     ↪  Version 2."
    rationale: "Even though the .rhosts files are ineffective if support is
     ↪  disabled in /etc/pam.conf, disabling the ability to use .rhosts
     ↪  files in SSH provides an additional layer of protection."
    remediation: "Edit the /etc/ssh/sshd_config file to set the parameter as
     ↪  follows: HostbasedAuthentication no"
    compliance:
       - cis: ["5.2.9"]
       - cis_csc: ["16.3"]
       - pci_dss: ["4.1"]
       - hipaa: ["164.312.a.2.IV", "164.312.e.1", "164.312.e.2.I",
        ↪  "164.312.e.2.II"]
       - nist_800_53: ["SC.8"]
       - tsc: ["CC6.7"]
    condition: all
    rules:
       - 'c:sshd -T -> r:HostbasedAuthentication\s+no'
```

SCAN RESULTS

SCA scan results appear as alerts with SCA scan data whenever a particular check changes its status between scans. Moreover, Wazuh agents only send those events necessary to keep the global status of the scan updated, avoiding potential events flooding.

Any given check event has three possible results:

- Passed
- Failed
- Not applicable

This result is determined by the set of rules and the rule result aggregator of the check.

Take the following SCA check from policy `cis_debian10.yml` as an example. The example SCA check shown scans the Debian 10 endpoint to verify if you have implemented a "deny all" policy on your endpoint firewall:

```
- id: 2603
    title: "Ensure IPv6 default deny firewall policy"
    description: "A default deny all policy on connections ensures that any
     ↪  unconfigured network usage will be rejected."
    rationale: "With a default accept policy the firewall will accept any
     ↪  packet that is not configured to be denied. It is easier to white
     ↪  list acceptable usage than to black list unacceptable usage."
    remediation: "Run the following commands to implement a default DROP
     ↪  policy: # ip6tables -P INPUT DROP # ip6tables -P OUTPUT DROP #
     ↪  ip6tables -P FORWARD DROP. Notes: Changing firewall settings while
     ↪  connected over network can result in being locked out of the system.
     ↪  Remediation will only affect the active system firewall, be sure to
     ↪  configure the default policy in your firewall management to apply on
     ↪  boot as well."
    compliance:
      - cis: ["3.5.4.2.1"]
      - cis_csc: ["9.4"]
      - pci_dss: ["1.2.1"]
      - tsc: ["CC8.1"]
    condition: all
    rules:
      - "c:ip6tables -L -> r:^Chain INPUT && r:policy DROP"
      - "c:ip6tables -L -> r:^Chain FORWARD && r:policy DROP"
      - "c:ip6tables -L -> r:^Chain OUTPUT && r:policy DROP"
```

After evaluating the aforementioned check, the following event is generated:

```
"data": {
  "sca": {
    "scan_id": "1433689708",
    "check": {
      "result": "failed",
      "remediation": "Run the following commands to implement a default DROP\
```

```
      policy: ip6tables -P INPUT DROP ip6tables -P OUTPUT DROP ip6tables \
      -P FORWARD DROP. Notes: Changing firewall settings while connected \
      over network can result in being locked out of the system. \
      Remediation will only affect the active system firewall, be sure \
      to configure the default policy in your firewall management to apply \
      on boot as well.",
      "compliance": {
        "pci_dss": "1.2.1",
        "tsc": "CC8.1",
        "cis_csc": "9.4",
        "cis": "3.5.4.2.1"
      },
      "description": "A default deny all policy on connections ensures that
      ↪  any unconfigured network usage will be rejected.",
      "id": "2603",
      "title": "Ensure IPv6 default deny firewall policy",
      "rationale": "With a default accept policy the firewall will accept
      ↪  any packet that is not configured to be denied. It is easier to
      ↪  white list acceptable usage than to black list unacceptable
      ↪  usage.",
      "command": [
        "ip6tables -L"
      ]
    },
    "type": "check",
    "policy": "CIS Benchmark for Debian/Linux 10"
  }
}
```


You can view the scan summaries on the Security configuration assessment tab on the Wazuh dashboard.

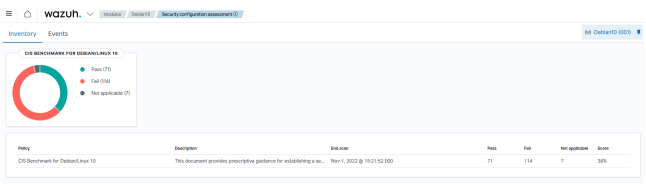


Figure 41: Scan Summary on Security Configuration Assessment

In addition, you can expand each result to display additional information.

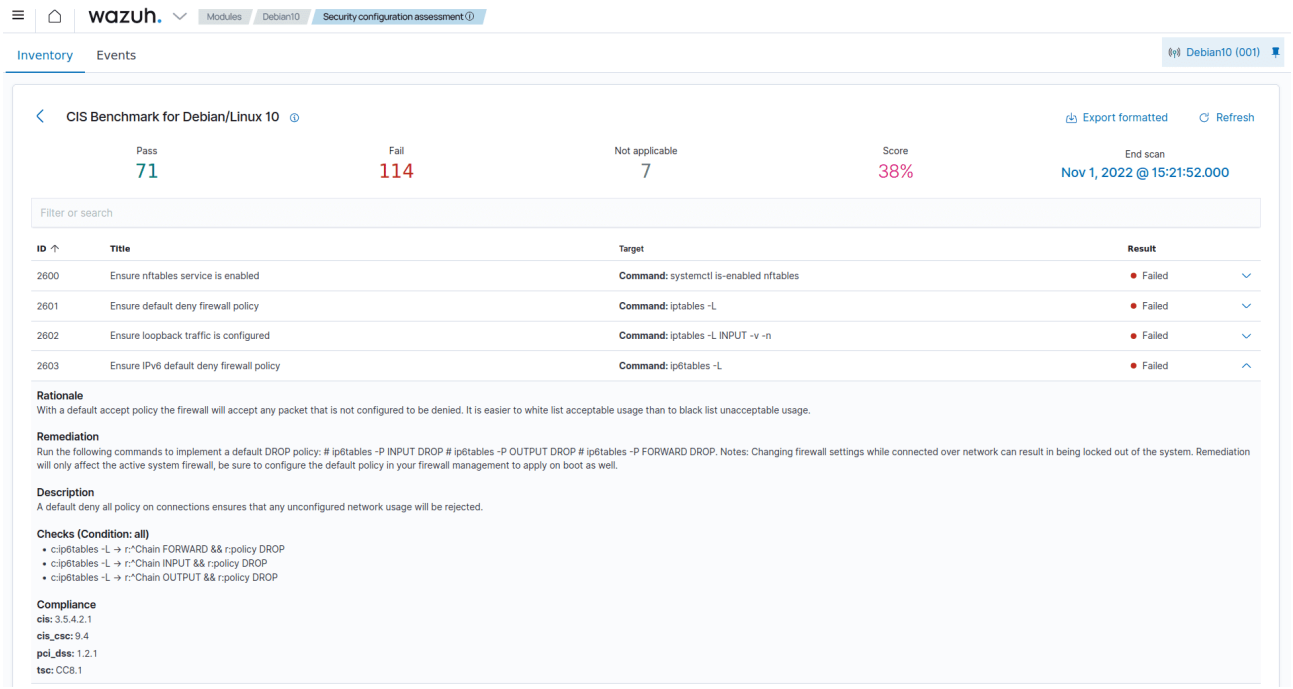


Figure 42: Expanded SCA Scan Result

The above SCA scan result is Failed because the rule did not find Chain INPUT * policy DROP, Chain FORWARD * policy DROP, and Chain OUTPUT * policy DROP in the output of the command ip6tables L. The steps below show how we implement the remediation steps suggested by Wazuh to harden the endpoint:

- Run the following recommended commands on the monitored endpoint to apply the firewall rules:

```
ip6tables -P INPUT DROP
ip6tables -P OUTPUT DROP
ip6tables -P FORWARD DROP
```

- Save the firewall rules and make them persist on system reboot:

```
ip6tables-save > /etc/ip6tables.conf
crontab -l | { cat; echo "@reboot /usr/sbin/ip6tables-restore
↪  /etc/ip6tables.conf"; } | crontab -
```

- Restart the Wazuh agent to trigger a new SCA scan:

```
systemctl restart wazuh-agent
```

The scan result for check `2603` changes to Passed as shown in the image below:

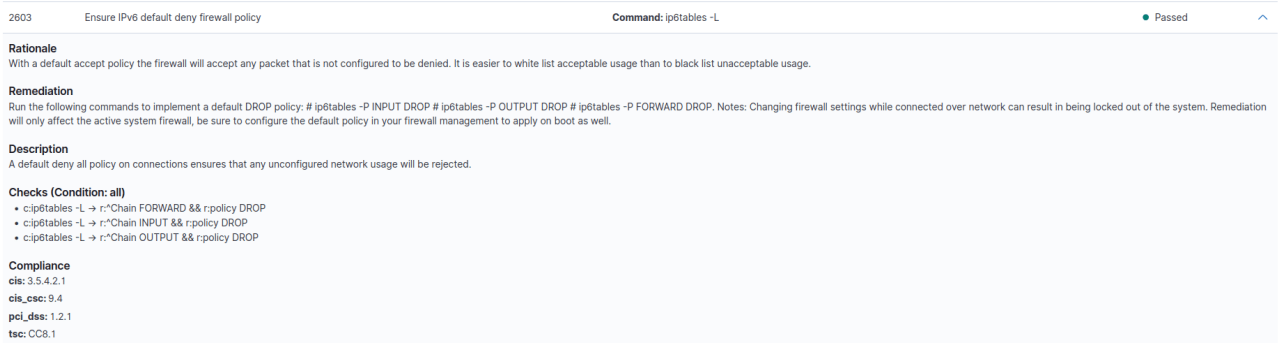


Figure 43: Scan Result for Check 2603

SCA scan result A check is marked as `Not applicable` in case an error occurs while performing the check. In such cases, instead of including the `result` field, the status and `reason` fields are included.

INTEGRITY MECHANISM

Wazuh uses two integrity mechanisms to ensure integrity between agent-side and server-side SCA states. One of the integrity mechanisms ensures the integrity of the policy files and the second ensures the integrity of scan results.

- Integrity of policy Files
- Integrity of Scan results

4.3.2 CONFIGURATION

Wazuh agents include the appropriate policies for their particular operating system during installation. For the full list of officially supported policy files, see the table Available SCA policies. These policies are included with the Wazuh server installation so that they can be easily enabled.

For a detailed description of the various configuration parameters of SCA, please check the SCA reference.

ENABLING AND DISABLING POLICIES

By default, the Wazuh agent runs scans for every policy (.yaml or .yml files) present in their ruleset folder:

- Linux and Unix-based agents: `/var/ossec/ruleset/sca`.
- Windows agents: `C:\Program Files (x86)\ossec-agent\ruleset\sca`.
- macOS agents: `/Library/Ossec/ruleset/sca`.

To enable a policy file outside the Wazuh agent installation folder, add the policy file path to the `⟨sca⟩` block in the Wazuh agent configuration file. An example is shown below:

```
<sca>
    <policies>
        <policy><FULLPATH_TO_CUSTOM_SCA_POLICY_FILE></policy>
    </policies>
</sca>
```

You can also specify a relative path to the Wazuh installation directory:

```
<sca>
    <policies>
        <policy>etc/shared/<CUSTOM_SCA_POLICY_FILE></policy>
    </policies>
</sca>
```

There are two ways to disable policies on the Wazuh agent. The simplest one is renaming the policy file by adding .disabled (or anything different from .yaml or .yml) after their YAML extension.

The second is to disable them from the Wazuh agent ossec.conf file by adding a line such as the following to the ¡policy¿ section of the SCA module:

```
<sca>
    <policies>
        <policy enabled="no">etc/shared/<POLICY_FILE_TO_DISABLE></policy>
    </policies>
</sca>
```

SHARING POLICY FILES AND CONFIGURATION WITH THE WAZUH AGENTS

As described in the centralized configuration section, the Wazuh manager can push files and configurations to connected Wazuh agents.

You can enable this feature to push policy files to the Wazuh agents in defined groups. By default, every Wazuh agent belongs to the default group, which is used here as an example:

On the Wazuh agent, edit the `local_internal_options.conf` file to allow the execution of commands in SCA policies sent from the Wazuh server:

```
echo "sca.remote_commands=1" >> /var/ossec/etc/local_internal_options.conf
```

On the Wazuh server, place a new policy file in the `/var/ossec/etc/shared/default` folder and change its ownership. Replace `<NEW_POLICY_FILE>` with your policy name.

```
chown wazuh:wazuh /var/ossec/etc/shared/default/<NEW\_POLICY\_FILE>
```

Add the following configuration block to the Wazuh server /var/ossec/etc/shared/default/agent.conf file to configure the new policy file in the Wazuh agent:

```
<agent_config>
<!-- Shared agent configuration here -->
    <sca>
        <policies>
            <policy>etc/shared/<NEW_POLICY_FILE></policy>
        </policies>
    </sca>
</agent_config>
```

All files remotely pushed from the Wazuh server are saved in the /¡WAZUH_HOME_DIRECTORY¿/etc/sha directory on the agent endpoints regardless of the group they belong to. We specify the relative file path of the policy in the configuration because the full file path could differ depending on the operating system of the monitored endpoint.

The new `<sca>` block in the Wazuh server `/var/ossec/etc/shared/default/agent`.conf file is merged with the `<sca>` block on the Wazuh agent side, and the new configuration is added.

4.3.3 SIMULATION

We demonstrate the use case of detecting a keyword in a file.

DETECTING KEYWORD IN A FILE

In this use case, we demonstrate how you can configure Wazuh SCA to detect the presence of a keyword in a file. We monitor a file `testfile.txt` for the phrase `password_enabled: yes`. The Wazuh SCA module triggers an alert when it detects the pattern in the file.

Ubuntu endpoint

- We created the test file and add some text to it, including the phrase `password_enabled: yes`.

```
echo -e "config_file\nsecond line of configuration\npassword_enabled:
↪  yes" > /usr/share/testfile.txt
```

- We verified that the file was created and the phrase was present.

```
cat /usr/share/testfile.txt
```

Here is the output:

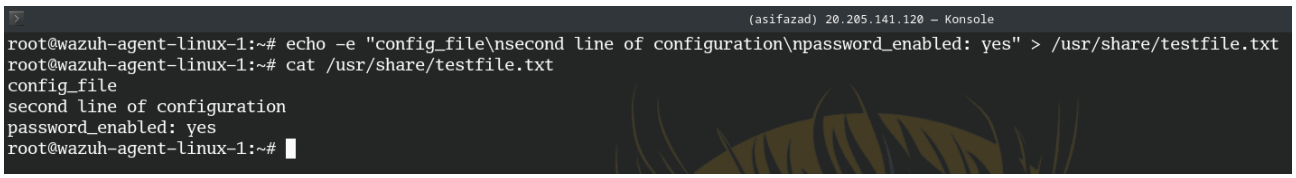


Figure 44: Creating the Test File and Adding the Phrase

- We created a new directory to save your custom policy files:

```
mkdir /var/ossec/etc/custom-sca-files/
```

- We created a new SCA policy file `/var/ossec/etc/custom-sca-files/keywordcheck.yml` and add the following content to it:

```
policy:
  id: "keyword_check"
  file: "keywordcheck.yml"
  name: "SCA use case: Keyword check"
```

```
  description: "Guidance for checking for a keyword or phrase in files
   ↪  on Ubuntu endpoints."
  references:
    - https://documentation.wazuh.com/current/user-manual/capabilities/
     ↪  sec-config-assessment/index.html
    - https://documentation.wazuh.com/current/user-manual/capabilities/
     ↪  sec-config-assessment/creating-custom-policies.html

requirements:
  title: "Check that the desired file exists on the monitored endpoints"
  description: "Requirements for running the SCA scans against
   ↪  endpoints with testfile.txt on them."
  condition: any
  rules:
    - 'f:/usr/share/testfile.txt'

checks:
  - id: 10000
    title: "Ensure password is disabled in the test configuration file"
    description: "Password is enabled in the test configuration file."
    rationale: "Password is considered weak for the custom test
     ↪  application. Threat actors can brute-force your password."
    remediation: "Disable password by setting the value of the
     ↪  password_enabled option to no."
    condition: none
    rules:
      - 'f:/usr/share/testfile.txt -> r:^password_enabled: yes$'
```

- Change the ownership of the file so Wazuh has permission to it:

```
chown wazuh:wazuh /var/ossec/etc/custom-sca-files/keywordcheck.yml
```

- We enabled the policy file by adding the following lines to the `<ossec_config>` block of the Wazuh agent configuration file at `/var/ossec/etc/ossec.conf`

```
<sca>
  <enabled>yes</enabled>
  <scan_on_start>yes</scan_on_start>
  <interval>12h</interval>
  <skip_nfs>yes</skip_nfs>

  <!-- adding a custom sca policy for detecting keyword in a file  -->
  <policies>
    <policy enabled="yes">/var/ossec/etc/custom-sca-files/keywordcheck.yml</policy>
  </policies>
</sca>
```

Figure 45: Enabling the Policy File

- Restart the Wazuh agent to apply the changes and to run the new SCA check:

```
systemctl restart wazuh-agent
```

- All the steps in terminal.

```
root@wazuh-agent-linux-1:~# echo -e "config_file\nsecond line of configuration\npassword_enabled: yes" > /usr/share/testfile.txt
root@wazuh-agent-linux-1:~# cat /usr/share/testfile.txt
config_file
second line of configuration
password_enabled: yes
root@wazuh-agent-linux-1:~# mkdir /var/ossec/etc/custom-sca-files/
root@wazuh-agent-linux-1:~# touch /var/ossec/etc/custom-sca-files/keywordcheck.yml
root@wazuh-agent-linux-1:~# vim /var/ossec/etc/custom-sca-files/keywordcheck.yml
root@wazuh-agent-linux-1:~# chown wazuh:wazuh /var/ossec/etc/custom-sca-files/keywordcheck.yml
root@wazuh-agent-linux-1:~# vim /var/ossec/etc/ossec.conf
root@wazuh-agent-linux-1:~# systemctl restart wazuh-agent
```

Figure 46: All the Steps in Terminal

4.3.4 DASHBOARD UPDATE

DETECTING KEYWORD IN A FILE

On our Wazuh dashboard, we navigated to the SCA tab and selected the Ubuntu endpoint to view the results of the custom SCA check we have created.

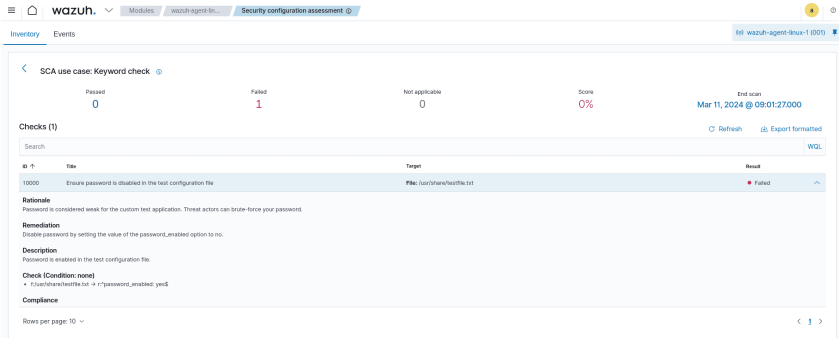


Figure 47: Dashboard Update for the Custom SCA Check

4.4 THREAT HUNTING

Threat hunting is a forward-looking security strategy that involves scrutinizing data from logs, network traffic, and endpoints to find and mitigate cyber threats bypassing conventional security measures. Its goal is to identify latent threats within an IT environment. The method encompasses hypothesis creation, data gathering, analysis, and reaction.

Wazuh bolsters security teams in their threat hunting efforts, enabling swift actions to confine the threat and prevent further harm.

4.4.1 LOG DATA ANALYSIS

This feature is elaborated later in 4.5. Efficient log data collection and analysis are critical for a robust threat-hunting approach. Wazuh facilitates **centralized log data collection**, integrating data from varied sources like endpoints, network devices, and applications for simplified analysis and enhanced monitoring efficiency.

Following is a depiction of the Wazuh dashboard settings for auditing log collection from a monitored endpoint.

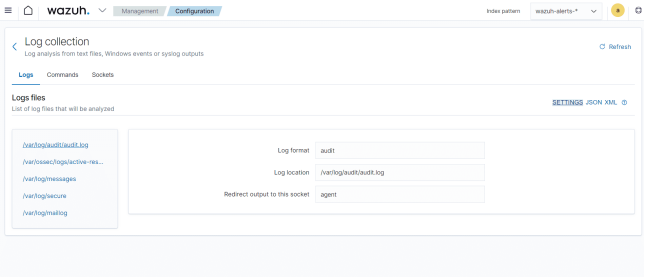


Figure 48: Log Collection Settings on the Wazuh Dashboard

Wazuh employs decoders to parse valuable data from collected log files, breaking down raw data into discernible attributes like timestamps, IP addresses, and event types. Navigate to `Stack Management ⟨ Index patterns ⟨ wazuh-alerts-*` to see the following fields:

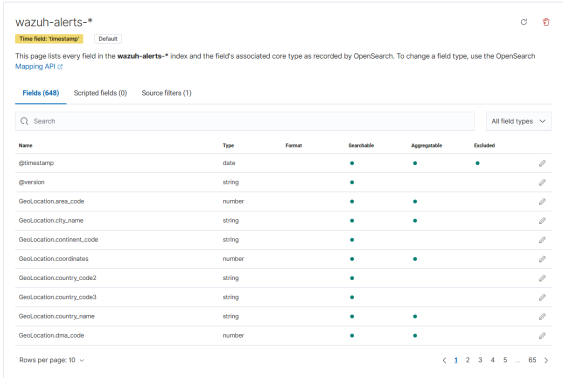


Figure 49: Index Patterns on the Wazuh Dashboard

Wazuh's capabilities extend to **agentless monitoring** and **syslog data collection**, ensuring efficient log management across various formats. Its indexing and querying features allow for swift data retrieval, aiding in quick analysis and investigations. Advanced parsing and real-time analysis fortify proactive threat identification and mitigation.

4.4.2 WAZUH ARCHIVES

Wazuh provides a centralized solution for log storage from monitored endpoints, including non-alert generating logs. By default, Wazuh archives are disabled but can be activated easily. Having access to extensive log details is vital for effective threat hunting, offering a comprehensive view of the environment.

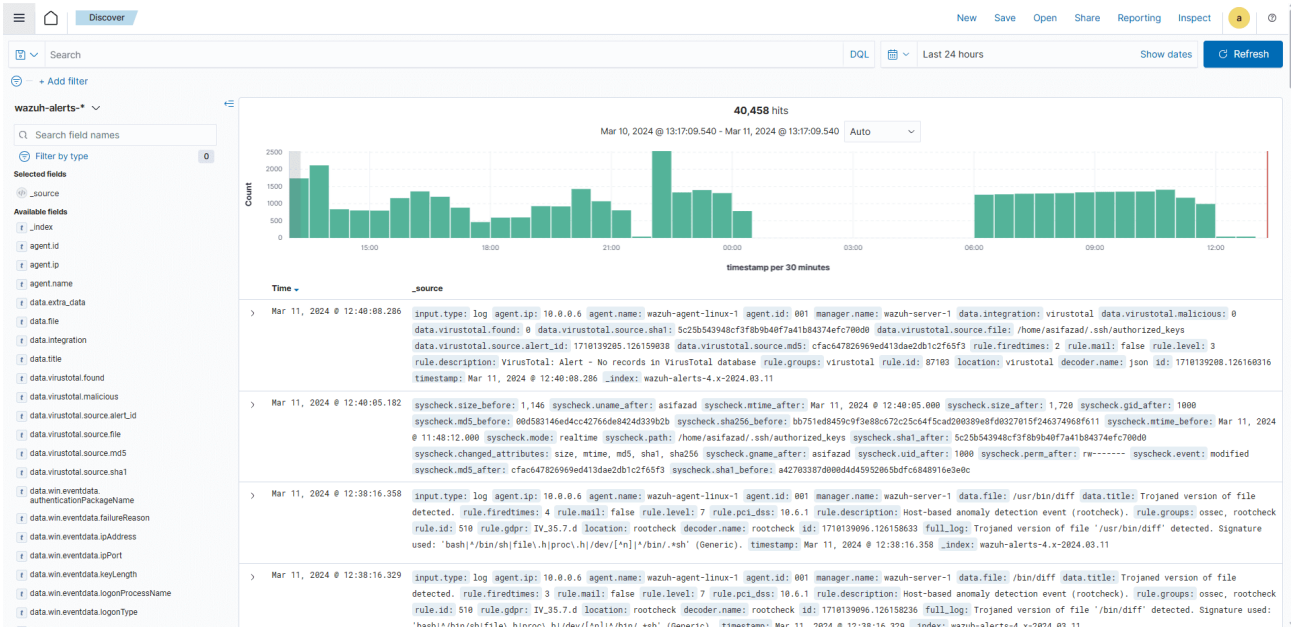


Figure 50: Archived Logs in the Discover Section of the Wazuh Dashboard

4.4.3 MITRE ATT&CK MAPPING

The MITRE ATT&CK framework provides a structured model to identify and understand cyber attackers' tactics, techniques, and procedures (TTPs). Wazuh's integration with the MITRE ATT&CK framework aids in identifying TTPs utilized by adversaries, enabling users to defend against them proactively.

For instance, unusual login activities can be associated with specific techniques within the framework, helping in the implementation of countermeasures. We particularly witnessed this because we had a number of agents that had public IP intentionally exposed for SSH connection. We witnessed a flurry of attacks from different corners of the globe, specially from places like Russia, China or North Korea. Details for these attacks including IP address or location were

available in the generated alerts.

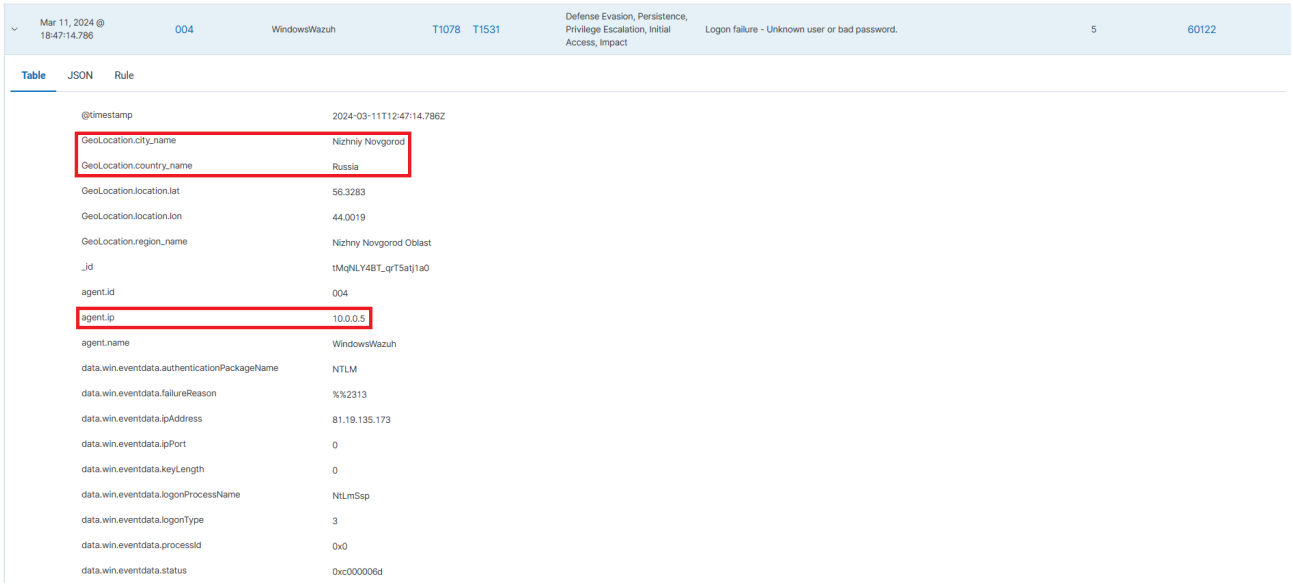


Figure 51: Authentication Attack Details from Generated Alerts

The Wazuh dashboard offered key insights into these attack techniques and their occurrence within the environment.

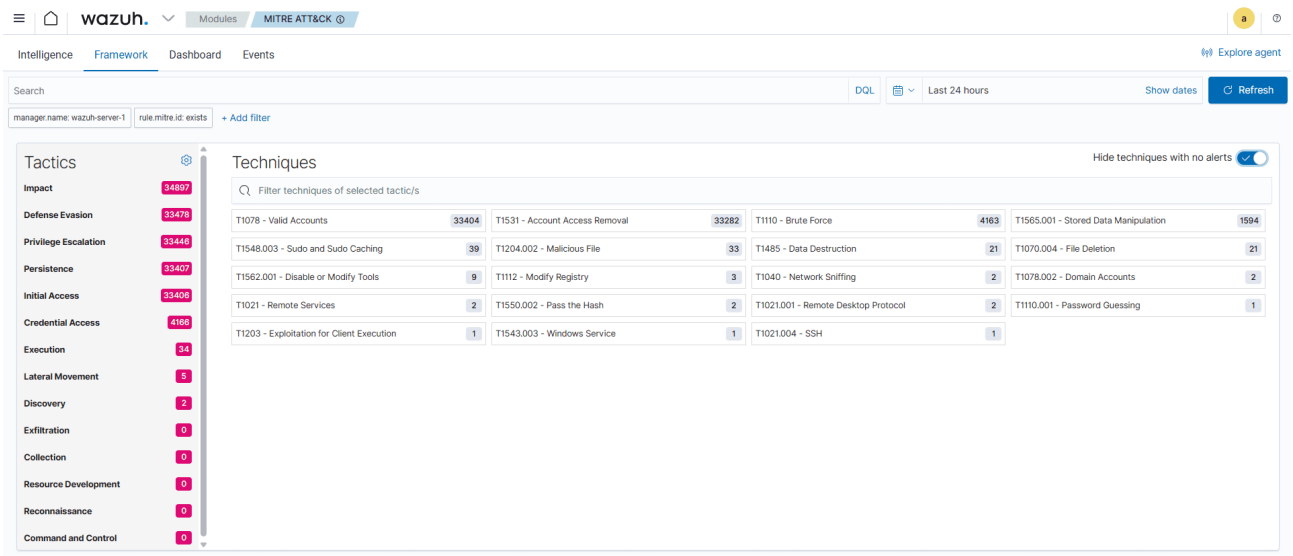


Figure 52: MITRE ATT&CK Module Shows the Possible Attack Techniques

The module generates detailed reports and visualizations, highlighting the frequency and severity of specific TTPs, assisting in compliance tracking and security enhancement.

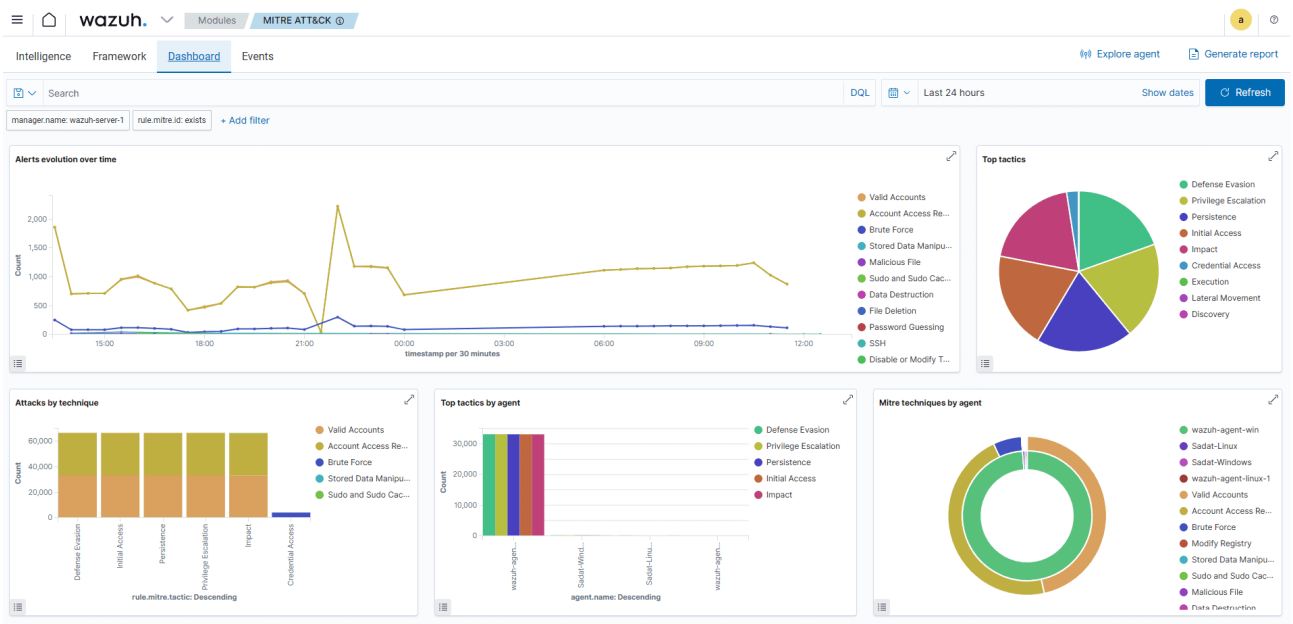


Figure 53: MITRE ATT&CK Module Dashboard on Wazuh

4.4.4 THIRD-PARTY INTEGRATION

Wazuh's compatibility with third-party tools amplifies threat hunting capabilities by consolidating data from diverse sources and automating threat detection and response processes.

Integrations with platforms like VirusTotal (shown in 4.2.3), AlienVault, and MISP, enhance the detection capabilities by allowing cross-referencing of data with threat intelligence feeds.

4.4.5 RULES AND DECODERS

Wazuh's strength in threat hunting is significantly attributed to its comprehensive set of rules, decoders, and pre-configured directives for a multitude of cyber threats and activities.

The management section on the Wazuh dashboard provides insight into both predefined and custom rules applicable to a variety of security incidents.

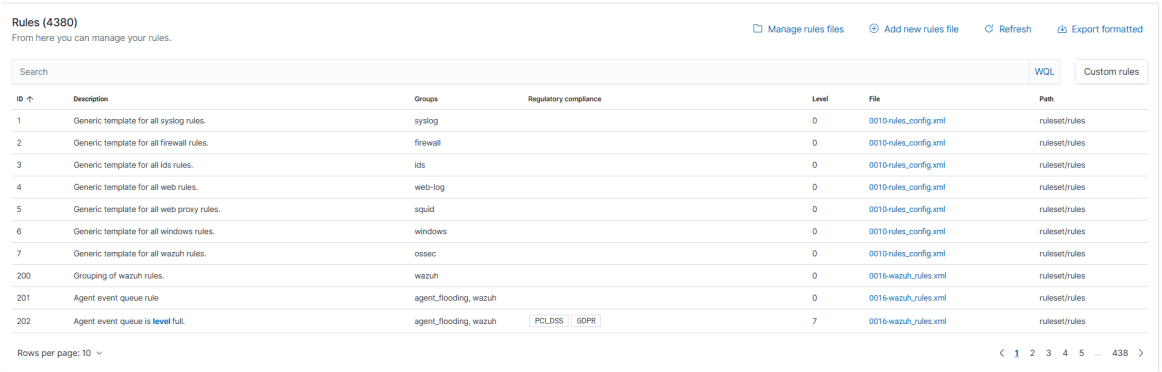


Figure 54: Rules View on the Wazuh Dashboard

Decoders play a crucial role in normalizing and interpreting log data, ensuring that information from various sources is standardized for efficient analysis.

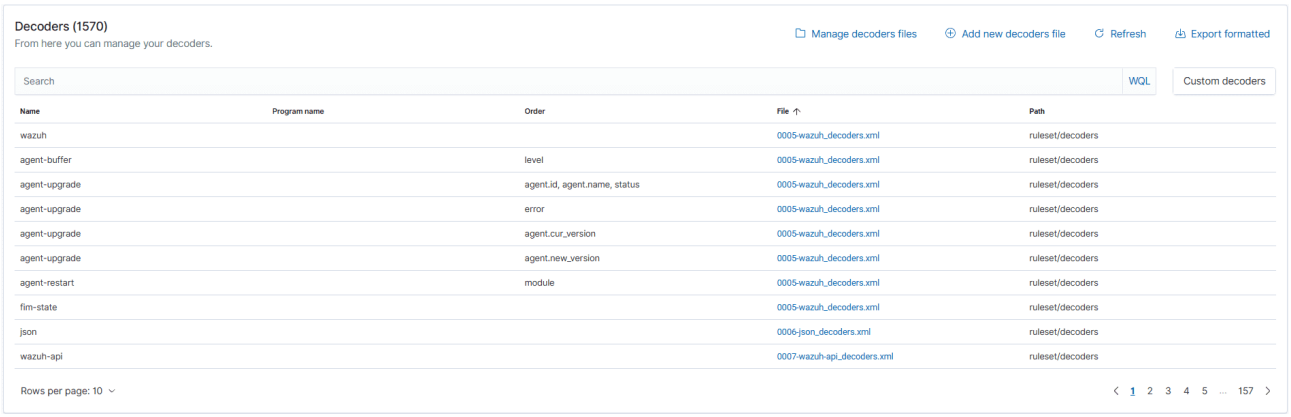


Figure 55: Decoders View on the Wazuh Dashboard

By leveraging Wazuh's capabilities, security teams gain valuable insights, enabling rapid detection of indicators of compromise, anomalous behavior, and potential security breaches.

4.5 LOG DATA ANALYSIS

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

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

4.5.1 HOW IT WORKS

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

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

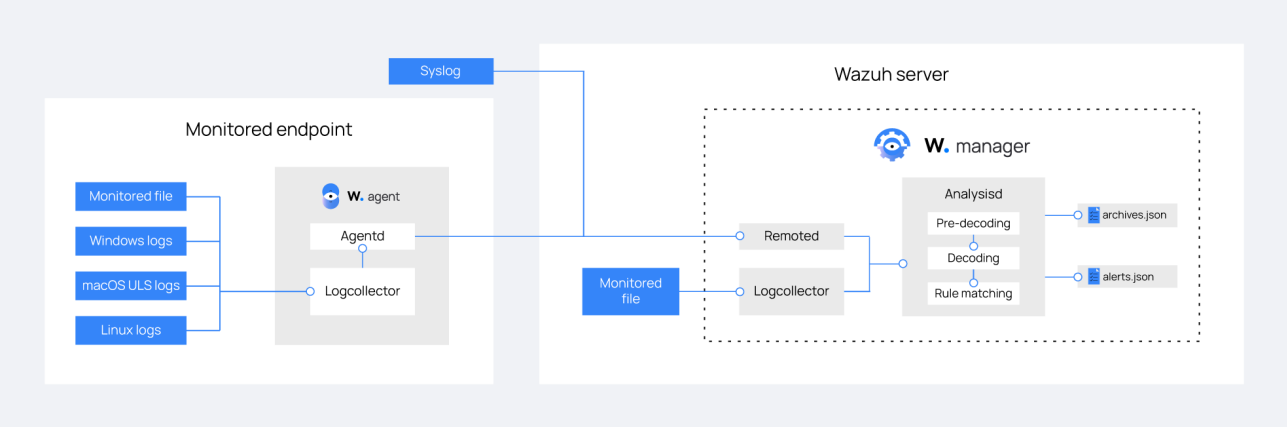


Figure 56: Working Flow of Wazuh Log Data Collection and Analysis

The log data collection process consists of 3 essential phases:

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

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

- **Rule Matching Phase:** Following decoding, the logs are matched against a comprehensive set of predefined rules in the `Analysisd` module. This phase is fundamental to identifying security incidents or policy violations. Each log is scrutinized, and if certain criteria are met, an alert is generated. This matching process not only identifies potential threats but also categorizes them based on severity, relevance, and type, enabling targeted response mechanisms and efficient threat mitigation.

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

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

4.5.2 CONFIGURATION

Wazuh supports two primary methods of log data collection.

USING SYSLOG

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

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

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

In this context:

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

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

```
systemctl restart wazuh-manager
```

USING WAZUH AGENT

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

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

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

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

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

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

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

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

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

4.5.3 SIMULATION

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

LINUX LOG DATA ANALYSIS USING RSYSLOG

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

Ubuntu endpoint

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

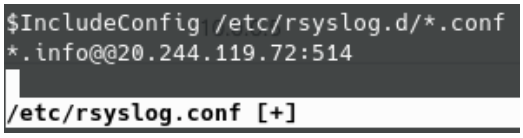


Figure 57: `rsyslog` Configuration

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

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

Figure 58: Restart `rsyslog`

Wazuh server

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

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

Figure 59: Wazuh Server Configuration

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

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

We test the configuration in the next sub-section.

WINDOWS LOG DATA ANALYSIS USING WAZUH AGENT

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

Windows endpoint

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

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

Figure 60: Wazuh Agent Configuration

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

Wazuh server

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

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

Figure 61: Wazuh Server Configuration

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

4.5.4 DASHBOARD UPDATE

LINUX LOG DATA ANALYSIS USING `RSYSLOG`

1. We add the new user `Alice`

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

Figure 62: Adding New User

2. We delete the user `Alice`

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

Figure 63: Deleting the New User

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

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

Figure 64: Alerts for User/Group Creation and Deletion

4. We expand the alert to see more details.

Figure 65: The name of the new user (red), the decoder used to process the log (blue)

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

Figure 66: Details of the Rule Used to Generate the Alert

WINDOWS LOG DATA ANALYSIS USING WAZUH AGENT

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

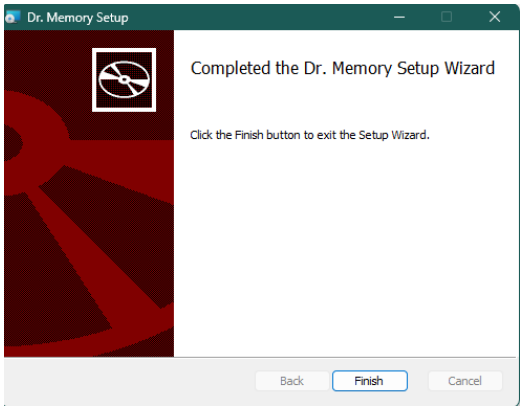


Figure 67: Installation of Dr. Memory

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

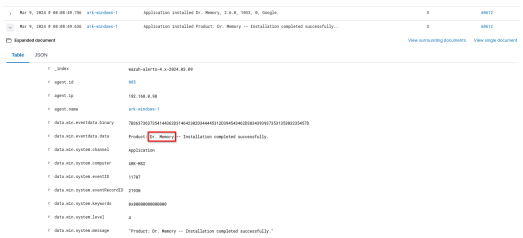


Figure 68: Alert for New Software Installation

5 SOURCE CODE

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

5.1 REPOSITORIES

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

WAZUH

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

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

WAZUH DASHBOARD

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

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

WAZUH DASHBOARD PLUGINS

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

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

WAZUH INDEXER

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

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

5.2 OVERVIEW OF THE CORE WAZUH SOURCE CODE

This section offers a high-level overview of the codebase. Each section of the codebase is discussed briefly below:

- **Architecture:** The `wazuh-db` folder contains a daemon responsible for managing access to SQLite database files. It handles automatic database upgrades, serialized and parallel queries, and other database-related tasks. Additionally, the `Metrics` folder contains metrics aiding in understanding the behavior of Wazuh components. The `syscollector` folder implements modules named Syscollector, Data Provider, DBSync, and RSync, responsible for collecting system information such as processes, hardware details, packages, OS specifics, network, and ports.

- **API:** This section includes code handling all Wazuh APIs and Wazuh API installer functions.

- **Active-response:** Here, active response scripts are managed, including default Wazuh scripts like `restart-wazuh`, `host-deny`, and `disable-account`.

- **addagent:** This section involves code related to managing Wazuh agents, including key management and server connections.

- **agentlessd:** It deals with agentless entry into hosts via SSH connections.

- **analysisd:** Crucial for analyzing events and collected logs, this section contains default decoders, pre-decoders, and rules for matching events. It also manages event creation and storage for dashboard display. The `alerts` folder stores different kinds of alerts seen in the security event section of the dashboard. It encompasses three important steps: pre-decoding, decoding, and rule-matching. All codes for default decoders and pre-decoders are present in this folder and its internal subfolders. The `compiled-rules` subfolder contains all the rules to be matched and relevant codes. Moreover, this section analyzes events and creates and stores new events to show on the dashboard.

- **agent-client:** Responsible for managing the state of agents in client endpoints, including client restarts and forwarding security events.

- **error_messages:** This section contains header files with various error messages used across Wazuh components.

- **headers:** Comprising miscellaneous headers and utility functions, it includes functionalities ranging from cryptography to file queues.

- **init:** Handles new user and group creation, deletion, and agent registration to servers.
- **logcollector:** Manages log data collection via agents or direct transmission to the server using the rsyslog protocol.
- **monitord:** Monitors logs and generates reports based on log data.
- **remoted:** Implements remote communication functionalities like sending messages, handling the syslog protocol, and facilitating shared downloads.
- **reportd:** Generates reports based on various input parameters.
- **Rootcheck:** Allows defining policies to check if agents meet specified requirements, including process checks, file presence, and content patterns. This feature is implemented in this section.
- **utils:** Contains helper functions and code related to agent control, agent listing, and verifying agent configurations. It also houses switch-cases for navigation from the command line.
- **wazuh_db:** Contains schema definitions for different Wazuh databases, including those for agents, modules (covered later), upgrades (all versions), rootcheck, etc.
- **wazuh_modules:** Houses main modules such as agent upgrade, syscollector (gathering system information), and task manager (coordinating and scheduling tasks between manager and agents).
- **syscheck:** Manages system integrity checking, including verifying file integrity and detecting system changes.
- **filebeat:** Integrates Filebeat with Wazuh, enabling log collection and analysis.

5.3 COMPILING THE FRONT-END FROM SOURCE

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

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

1. Remove or disable standalone Docker Engine (if installed). Install Docker Desktop.

2. Configure the docker environment.

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

3. Assign enough resources to Docker Desktop. At least -

   - 8 GB of RAM
   - 4 CPU Cores

4. Save the path to the `plugins` folder inside `wazuh-dashboard-plugins` repository code as an environment variable, by exporting this path on .bashrc, .zhsrc or similar.

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

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

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

6. Clone the repository.

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

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

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

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

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

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

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

   where,

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

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

```
docker run --name os-dev-280-agent-$(date +%s) \
    --network os-dev-2.8.0 \
    --label com.docker.compose.project=os-dev-280 \
    --env WAZUH_AGENT_VERSION=4.7.2 \
    -d ubuntu:20.04 bash -c
    'apt update -y
    apt install -y curl lsb-release
    curl -so \wazuh-agent-${WAZUH_AGENT_VERSION}.deb \
        "https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/"\
        "wazuh-agent_${WAZUH_AGENT_VERSION}-1_amd64.deb" \
        && WAZUH_MANAGER='wazuh.manager' WAZUH_AGENT_GROUP='default' \
        dpkg -i ./wazuh-agent-${WAZUH_AGENT_VERSION}.deb
```

```
    /etc/init.d/wazuh-agent start
    tail -f /var/ossec/logs/ossec.log'
```

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

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

12. Install the dependencies using:

```
yarn install
```

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

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

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

Figure 69: Running the Wazuh Dashboard from Source Code

As a demonstration, we place a new name in the dashboard title bar.

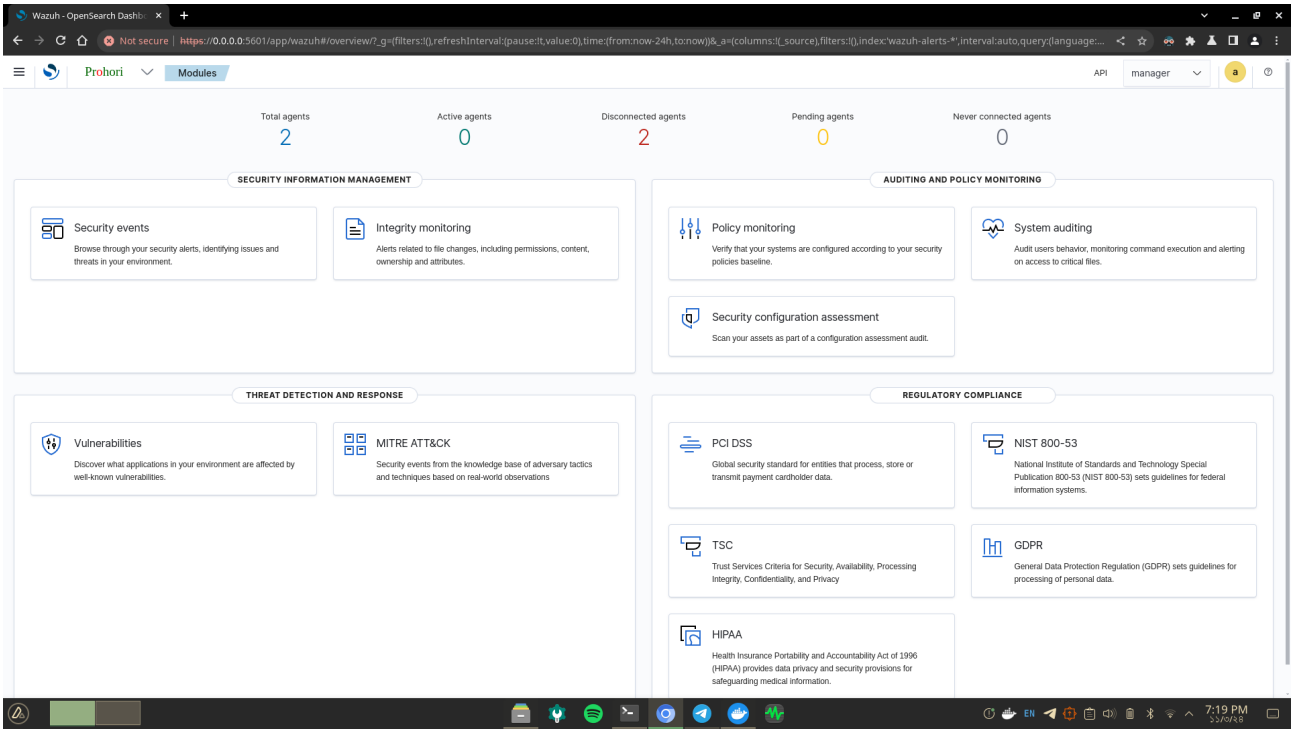


Figure 70: Wazuh Dashboard with Custom Title - Prohori

Needless to say, getting this done opens a door of endless possibilities for us to customize the dashboard and add new features to it. Even more importantly, there is a commercial aspect to it. We can now offer our services to other organizations to customize their Wazuh dashboards as per their requirements.

6 ISSUES FACED

6.1 SERVER CRASH: MACOS NOT SUPPORTING REALTIME MONITORING

- For the File Integrity Module (FIM), we were opting for realtime monitoring for both Windows and Linux.
- But this configuration was not working on macOS. Later, we found out that macOS does not support realtime monitoring to begin with.
- We had to set a monitoring frequency then. Naively, we chose 1 second.
- Because of such frequent logging, the server could not take the load and suffered a crash.
- Later, we changed the frequency to 1 minute and restarted the server. Things started working nicely afterwards.

6.2 RANDOM AUTHENTICATION ERROR ON SERVER

- Strangely enough, at times, we could not login to Wazuh dashboard with appropriate username and passwords. It just said, incorrect username or password.
- We examined the `wazuh-install-files/wazuh-passwords.txt` and saw our credentials were alright.
- Even more strangely, everytime the problem got fixed by a restart of the server.

6.3 SOURCE CODE COMPILATION CHALLENGES

- Documentation and detailed instructions are pretty scant.
- The exact release tag of the repository needs to be checked out for the source code to compile. The master branch doesn't always compile with the officially provided scripts.
- The OpenSearch, OpenSearch Dashboard and Wazuh versions need to be compatible in order for `dev.sh` script to work. However, compatible versions aren't documented anywhere.

7 REFERENCE

- The Official Wazuh Documentation
- Wazuh GitHub Repositories