Bangladesh University of Engineering and Technology

Department of Computer Science and Engineering

Academic Year 2023–2024

CSE 406

Computer Security Sessional

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

Submitted by:

1905001 — Mohammad Sadat Hossain

1905004 — Asif Azad

1905005 — Md. Ashrafur Rahman Khan

Supervisor: Abdur Rashid Tushar

Submission Date: March 10, 2024

Contents

1 Introduction to Wazuh **3**
- 1.1 What is Wazuh? . . . 3
- 1.2 Wazuh Components . . . 3
  - 1.2.1 Wazuh Agent . . . 3
  - 1.2.2 Wazuh Manager . . . 4
- 1.3 Wazuh Architecture . . . 5

2 Installation Prerequisites **6**
- 2.1 System Requirements . . . 6
  - 2.1.1 Hardware Specifications . . . 6
  - 2.1.2 Operating System Compatibility . . . 6
  - 2.1.3 Web Browser Support . . . 6
- 2.2 Configuring the Machines . . . 7
  - 2.2.1 Wazuh Server . . . 7
  - 2.2.2 Wazuh Agents . . . 7

3 Installation **9**
- 3.1 Setting Up the Wazuh Server . . . 9
  - 3.1.1 Quickstart Installation . . . 9
  - 3.1.2 Step-by-step Installation . . . 10
- 3.2 Registering Agents . . . 10
  - 3.2.1 Linux . . . 11
  - 3.2.2 MacOS . . . 12
  - 3.2.3 Windows . . . 13

4 Wazuh Features and Use-cases **15**
- 4.1 Malware Detection . . . 15
  - 4.1.1 CDB Lists and Threat Intelligence . . . 16
    - How it works . . . 16
    - Configuration . . . 16
    - Simulation . . . 19
    - Dashboard Update . . . 19
  - 4.1.2 File Integrity Monitoring and YARA Scanning . . . 20
    - How it works . . . 20
    - Configuration . . . 21
    - Simulation . . . 26

Dashboard Update . . . 29
4.1.3 VirusTotal Integration . . . 30
How it works . . . 30
Configuration . . . 36
Simulation . . . 37
Dashboard Update . . . 37
4.2 Log Data Analysis . . . 40
4.2.1 How it works . . . 40
4.2.2 Configuration . . . 42
Using Syslog . . . 42
Using Wazuh Agent . . . 43
4.2.3 Simulation . . . 44
Linux Log Data Analysis using `rsyslog` . . . 44
Windows Log Data Analysis using Wazuh Agent . . . 45
4.2.4 Dashboard Update . . . 46
Linux Log Data Analysis using `rsyslog` . . . 46
Windows Log Data Analysis using Wazuh Agent . . . 47

5 Source Code **48**
5.1 Repositories . . . 48
5.2 Compiling the Front-end from Source . . . 49

6 Issues Faced **51**
6.1 Server Crash: macOS Not Supporting Realtime Monitoring . . . 51
6.2 Random Authentication Error on Server . . . 52
6.3 Source Code Compilation Challenges . . . 52

7 Reference **52**

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.

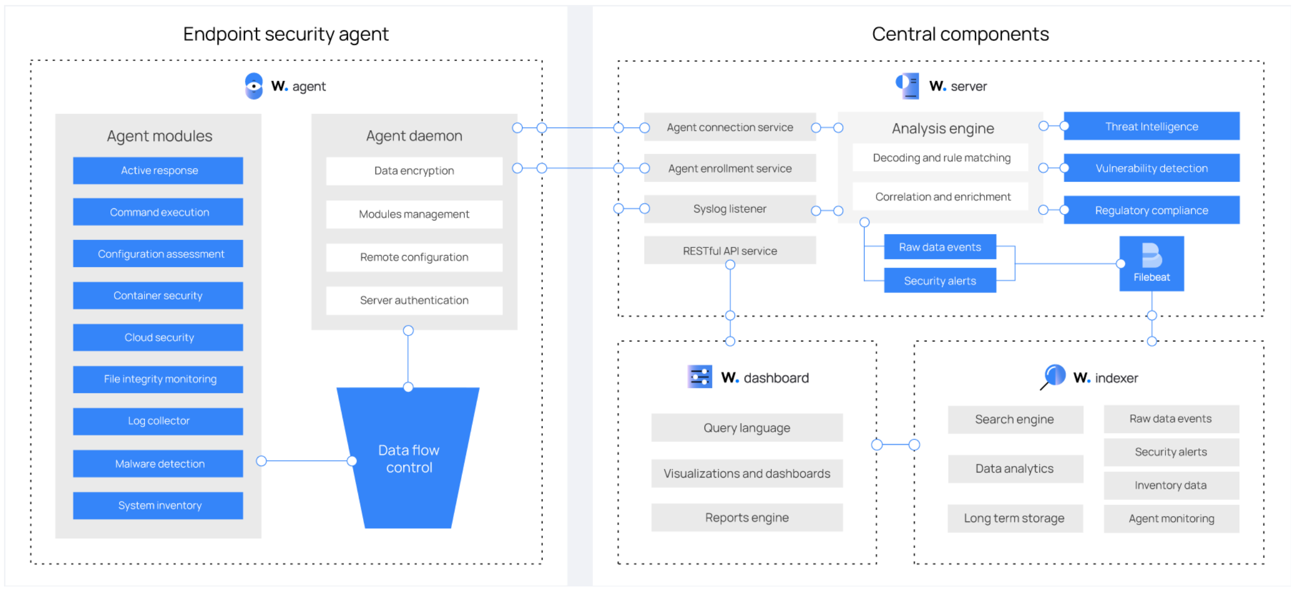


Figure 1: Wazuh Components and Data flow

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.

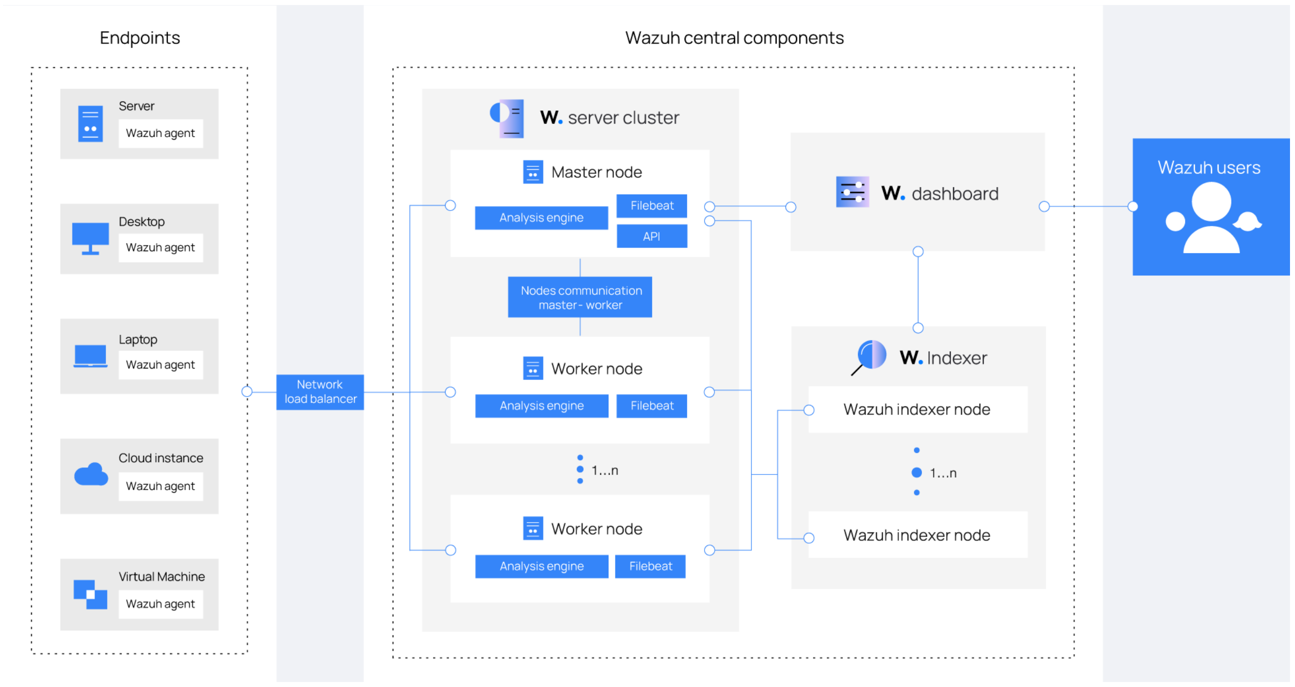


Figure 2: Overview of Wazuh Deployment Architecture

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 B2s, 2 vCPUs, 4GB RAM
- **Public IP:** N/A
- **Private IP:** 10.0.0.6

AGENT ID: 002

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

AGENT ID: 007

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

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

AGENT ID: 008

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

AGENT ID: 009

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

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

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`.

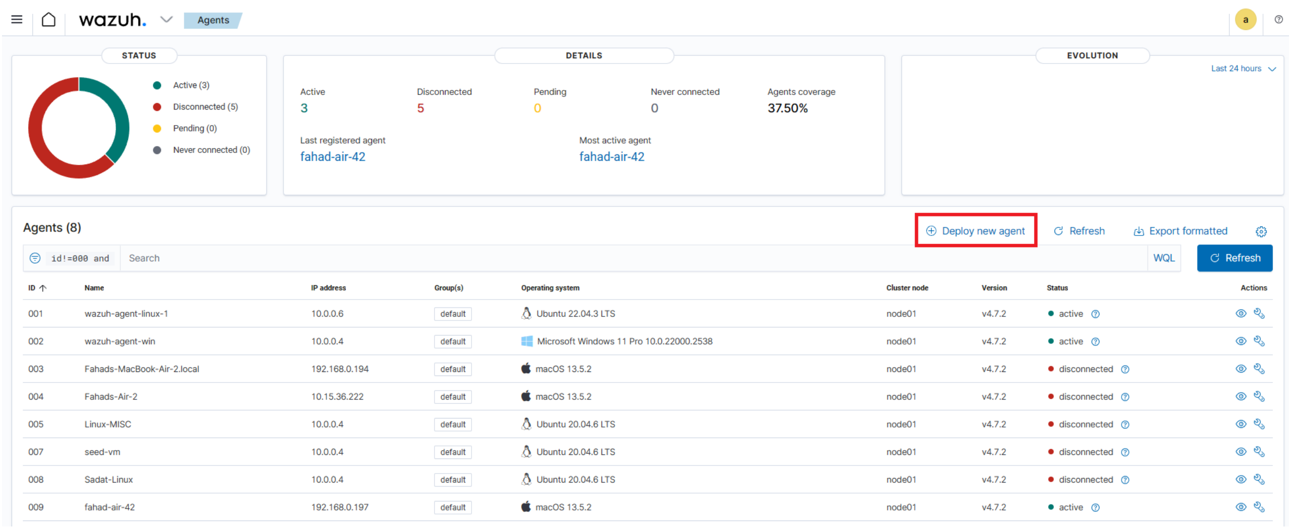
3.1.2 STEP-BY-STEP INSTALLATION

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:



Active	Disconnected	Pending	Never connected	Agents coverage
3	5	0	0	37.50%

Last registered agent: fahad-air-42
Most active agent: fahad-air-42

Agents (8)

`id!=000 and` Search

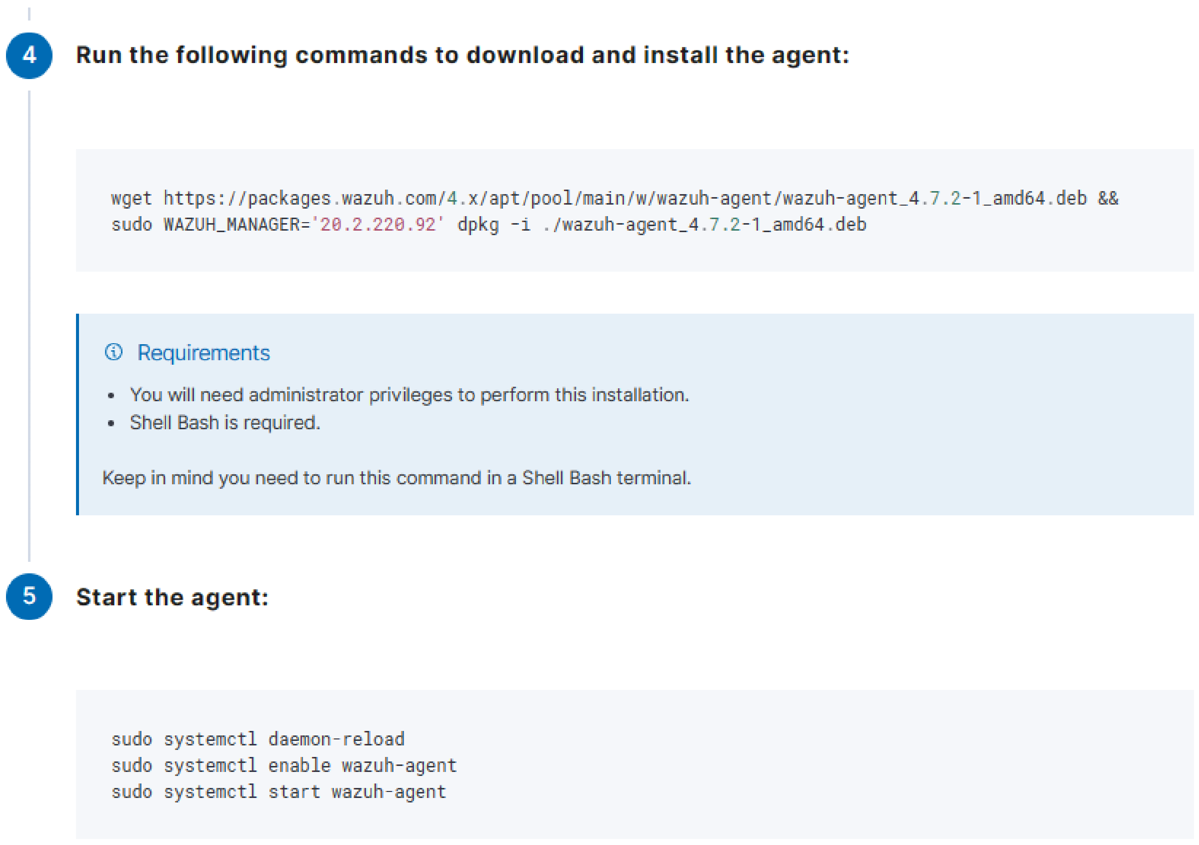
ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	wazuh-agent-linux-1	10.0.0.6	default	Ubuntu 22.04.3 LTS	node01	v4.7.2	active	
002	wazuh-agent-win	10.0.0.4	default	Microsoft Windows 11 Pro 10.0.22000.2538	node01	v4.7.2	active	
003	Fahads-MacBook-Air-2.local	192.168.0.194	default	macOS 13.5.2	node01	v4.7.2	disconnected	
004	Fahads-Air-2	10.15.36.222	default	macOS 13.5.2	node01	v4.7.2	disconnected	
005	Linux-MISC	10.0.0.4	default	Ubuntu 20.04.6 LTS	node01	v4.7.2	disconnected	
007	seed-vm	10.0.0.4	default	Ubuntu 20.04.6 LTS	node01	v4.7.2	disconnected	
008	Sadat-Linux	10.0.0.4	default	Ubuntu 20.04.6 LTS	node01	v4.7.2	disconnected	
009	fahad-air-42	192.168.0.197	default	macOS 13.5.2	node01	v4.7.2	active	

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



4 **Run the following commands to download and install the agent:**

```
wget https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.7.2-1_amd64.deb &&
sudo WAZUH_MANAGER='20.2.220.92' dpkg -i ./wazuh-agent_4.7.2-1_amd64.deb
```

Requirements

- You will need administrator privileges to perform this installation.
- Shell Bash is required.

Keep in mind you need to run this command in a Shell Bash terminal.

5 **Start the agent:**

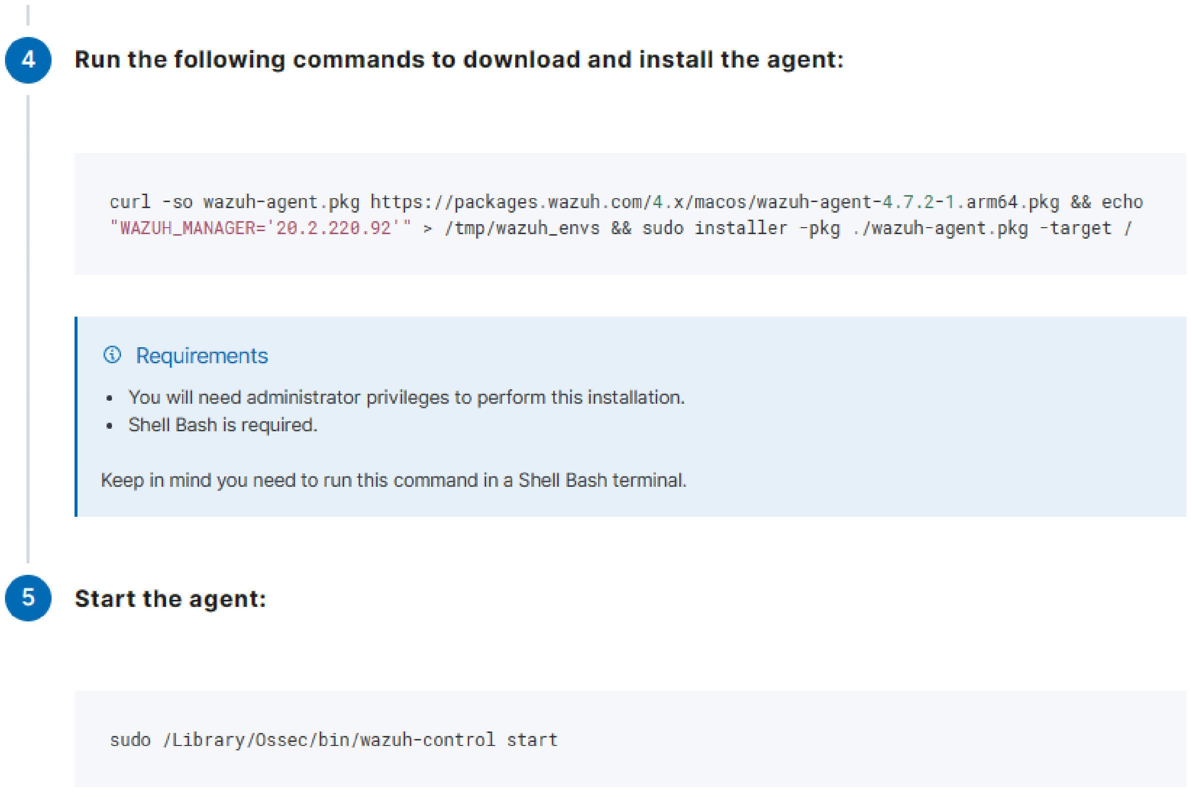
```
sudo systemctl daemon-reload
sudo systemctl enable wazuh-agent
sudo systemctl start wazuh-agent
```

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


4 **Run the following commands to download and install the agent:**

```
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 /
```

Requirements

- You will need administrator privileges to perform this installation.
- Shell Bash is required.

Keep in mind you need to run this command in a Shell Bash terminal.

5 **Start the agent:**

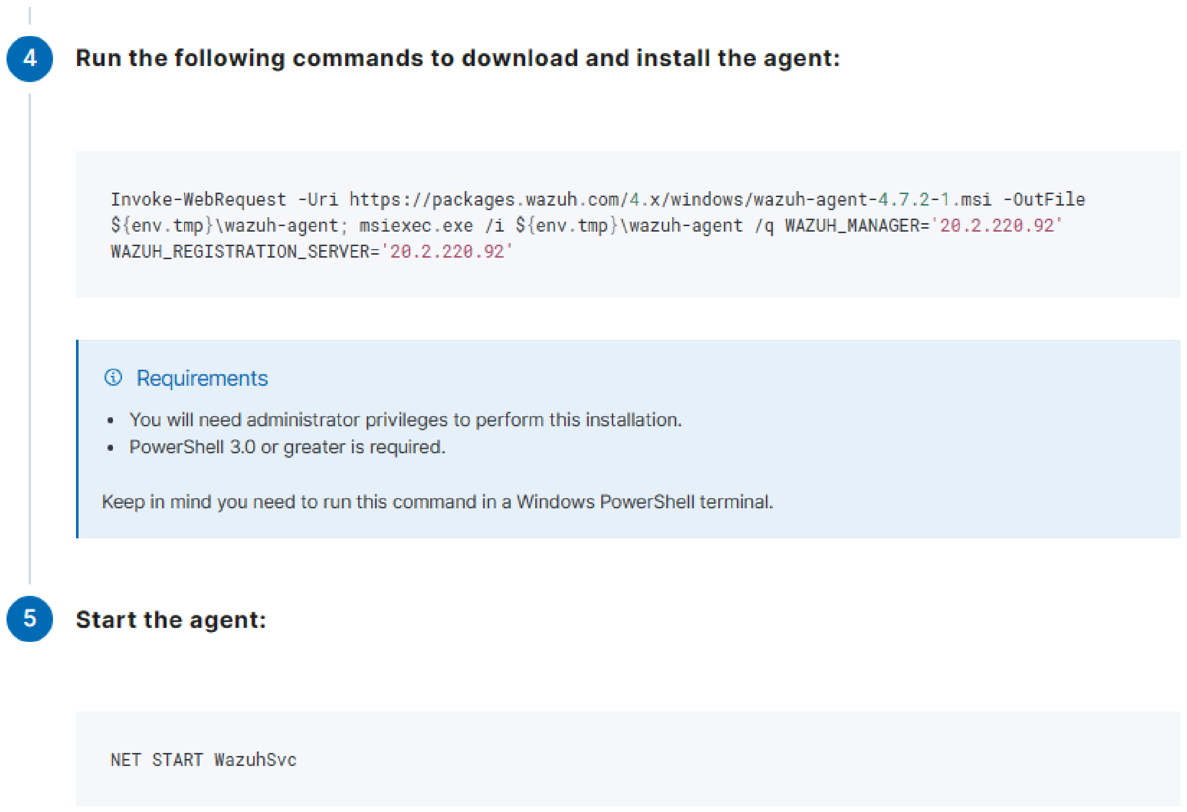
```
sudo /Library/Ossec/bin/wazuh-control start
```

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



4 Run the following commands to download and install the agent:

```
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'
```

Requirements

- You will need administrator privileges to perform this installation.
- PowerShell 3.0 or greater is required.

Keep in mind you need to run this command in a Windows PowerShell terminal.

5 Start the agent:

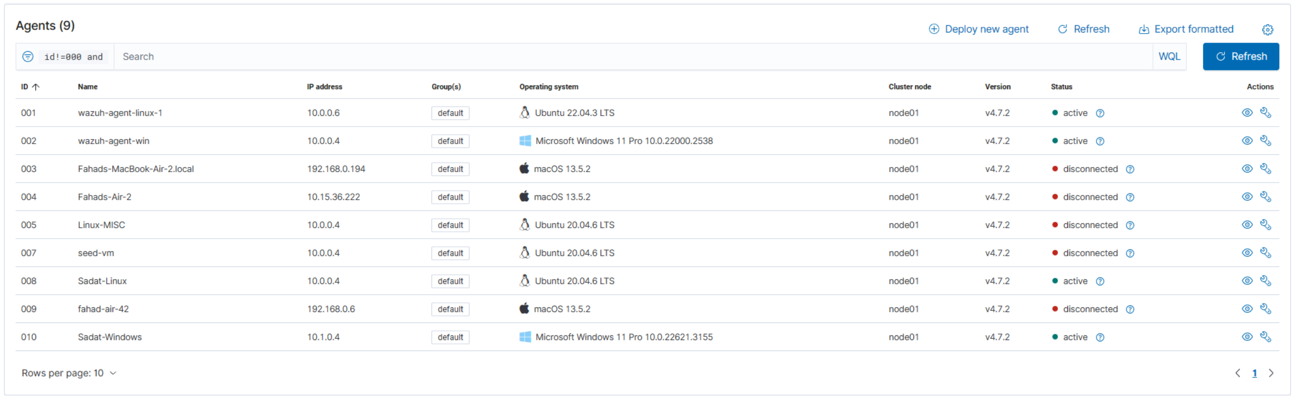
```
NET START WazuhSvc
```

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.


Agents (9)

Deploy new agent | Refresh | Export formatted

`id!=000 and` Search | WQL | Refresh

ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	wazuh-agent-linux-1	10.0.0.6	default	Ubuntu 22.04.3 LTS	node01	v4.7.2	● active	
002	wazuh-agent-win	10.0.0.4	default	Microsoft Windows 11 Pro 10.0.22000.2538	node01	v4.7.2	● active	
003	Fahads-MacBook-Air-2.local	192.168.0.194	default	macOS 13.5.2	node01	v4.7.2	● disconnected	
004	Fahads-Air-2	10.15.36.222	default	macOS 13.5.2	node01	v4.7.2	● disconnected	
005	Linux-MISC	10.0.0.4	default	Ubuntu 20.04.6 LTS	node01	v4.7.2	● disconnected	
007	seed-vm	10.0.0.4	default	Ubuntu 20.04.6 LTS	node01	v4.7.2	● disconnected	
008	Sadat-Linux	10.0.0.4	default	Ubuntu 20.04.6 LTS	node01	v4.7.2	● active	
009	fahad-air-42	192.168.0.6	default	macOS 13.5.2	node01	v4.7.2	● disconnected	
010	Sadat-Windows	10.1.0.4	default	Microsoft Windows 11 Pro 10.0.22621.3155	node01	v4.7.2	● active	

Rows per page: 10

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 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.

4.1.1 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

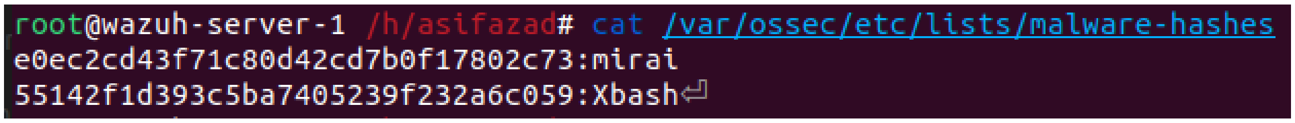
Wazuh server

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.

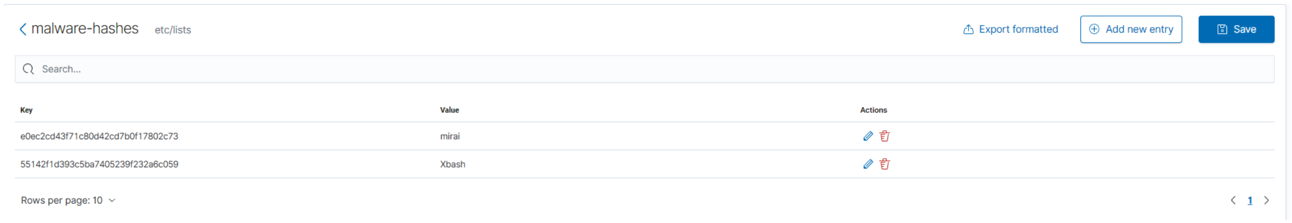
```
e0ec2cd43f71c80d42cd7b0f17802c73:mirai
55142f1d393c5ba7405239f232a6c059:Xbash
```



```
root@wazuh-server-1 /h/asifazad# cat /var/ossec/etc/lists/malware-hashes
e0ec2cd43f71c80d42cd7b0f17802c73:mirai
55142f1d393c5ba7405239f232a6c059:Xbash⏎
```

Figure 8: List of Malware Hashes (Terminal)

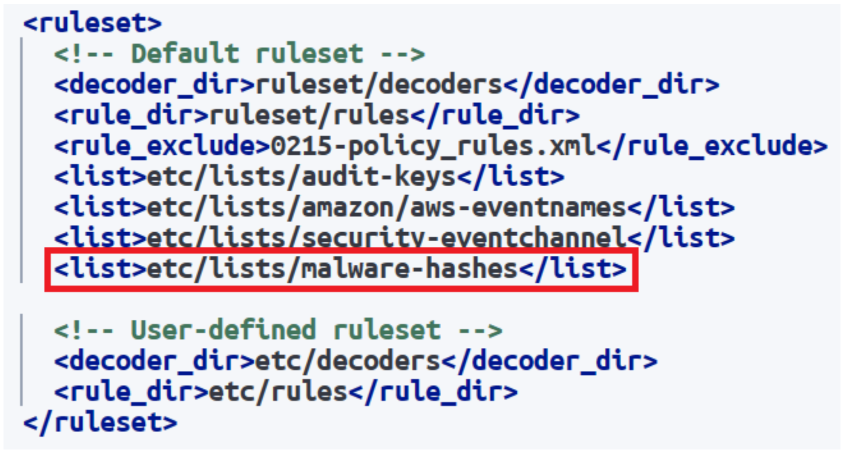
Alternatively, these configurations can also be updated from the Wazuh dashboard, like the following:



Key	Value	Actions
e0ec2cd43f71c80d42cd7b0f17802c73	mirai	
55142f1d393c5ba7405239f232a6c059	Xbash	

Figure 9: List of Malware Hashes (Dashboard)

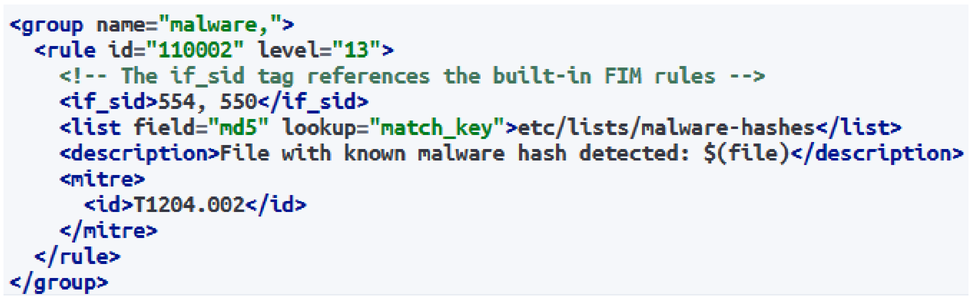
3. Add a reference to the CDB list in the Wazuh manager configuration file `/var/ossec/etc/ossec.conf`. This can be done by specifying the path to the list within the `<ruleset>` block:



```
<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 10: 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>
```

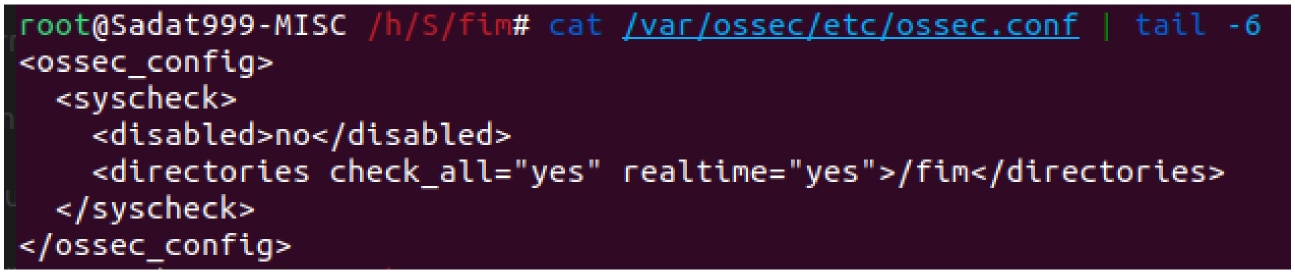
Figure 11: Custom Rule added to Server

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 12: Adding a Monitored Directory

2. Restart the Wazuh agent to apply the changes:

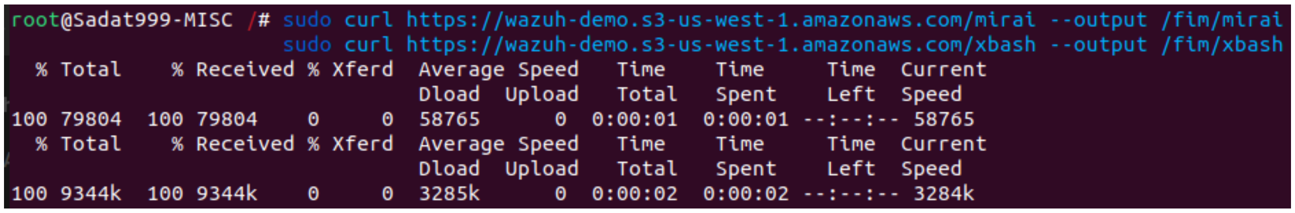
```
systemctl restart wazuh-agent
```

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 13: Manually Downloading the Malwares

DASHBOARD UPDATE

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

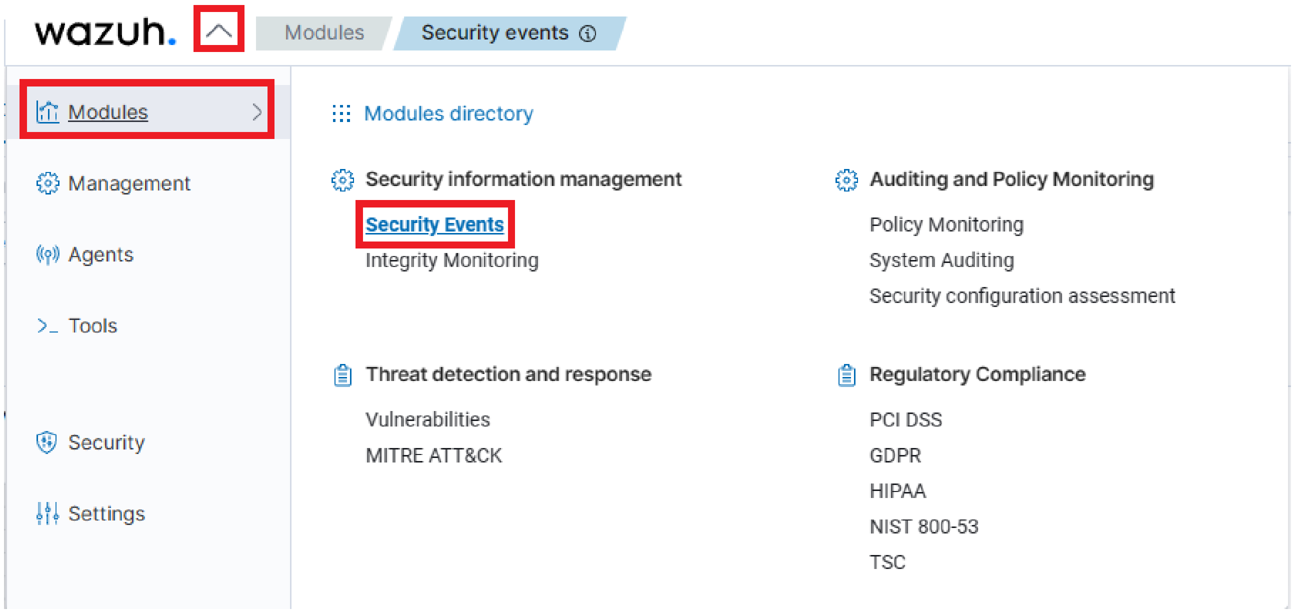


Figure 14: Navigation to Security Events

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.

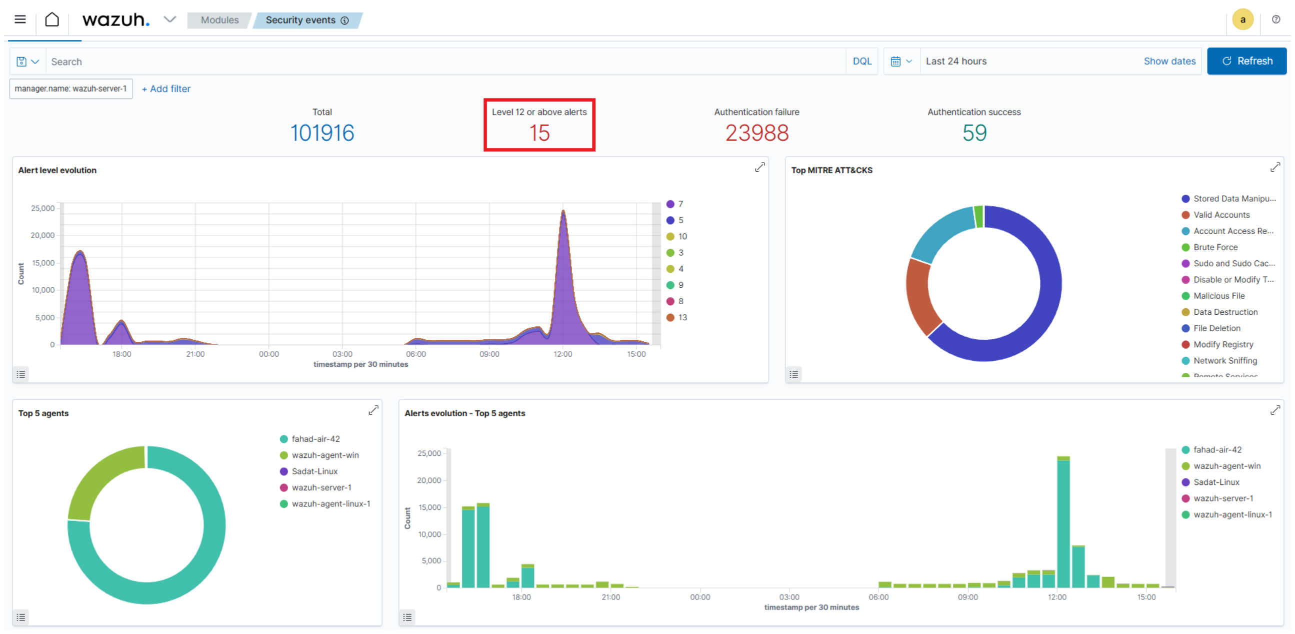
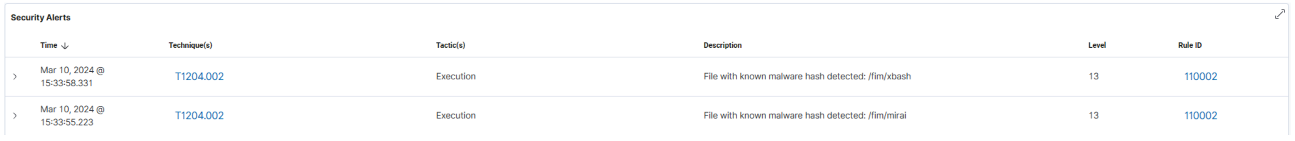


Figure 15: Dashboard after Malware Download

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.



Security Alerts

Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Mar 10, 2024 @ 15:33:58.331	T1204.002	Execution	File with known malware hash detected: /fim/xbash	13	110002
Mar 10, 2024 @ 15:33:55.223	T1204.002	Execution	File with known malware hash detected: /fim/mirai	13	110002

Figure 16: Alerts Generated by CDB Matching

4.1.2 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.

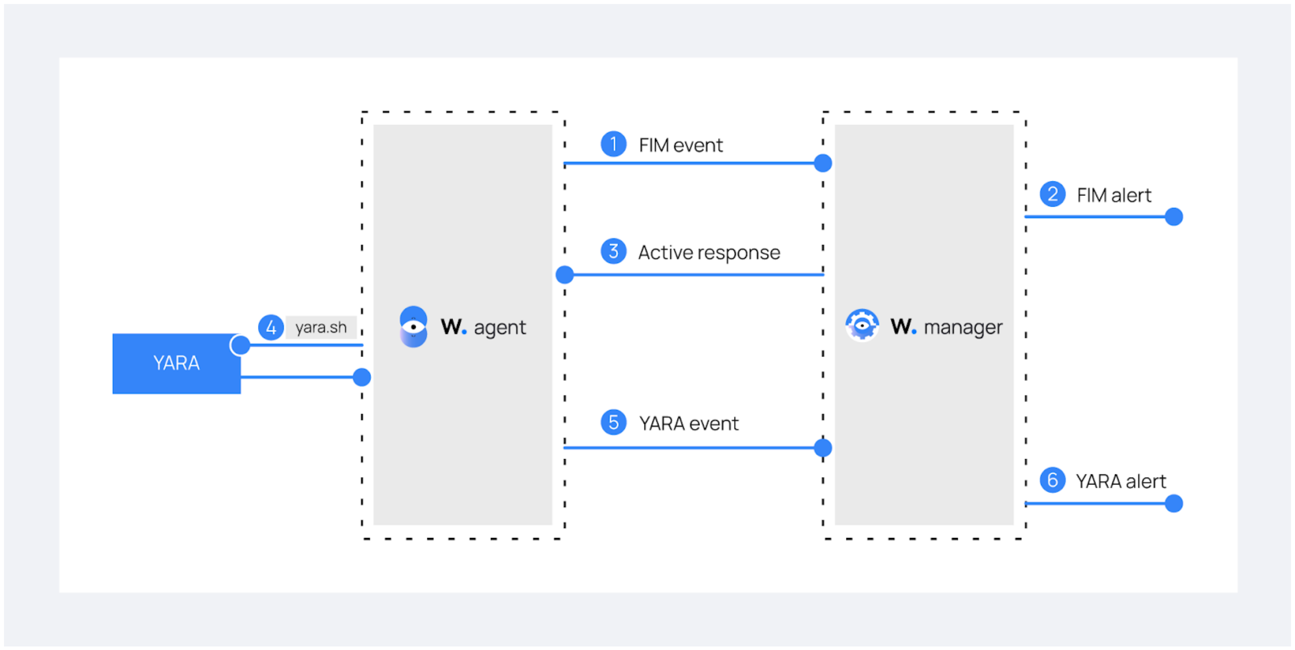


Figure 17: Workflow of Malware Detection through YARA scanning

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.

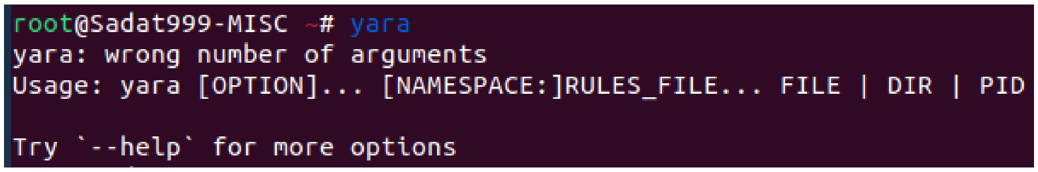
CONFIGURATION

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.



```
root@Sadat999-MISC ~# yara
yara: wrong number of arguments
Usage: yara [OPTION]... [NAMESPACE:]RULES_FILE... FILE | DIR | PID

Try `--help` for more options
```

Figure 18: 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=1111111111111111111111111111111111111111111
↪  1111111111111111111&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 `parameters.alert.sysc` key of the JSON alert holds the value of the file path. 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 `/root/` directory:

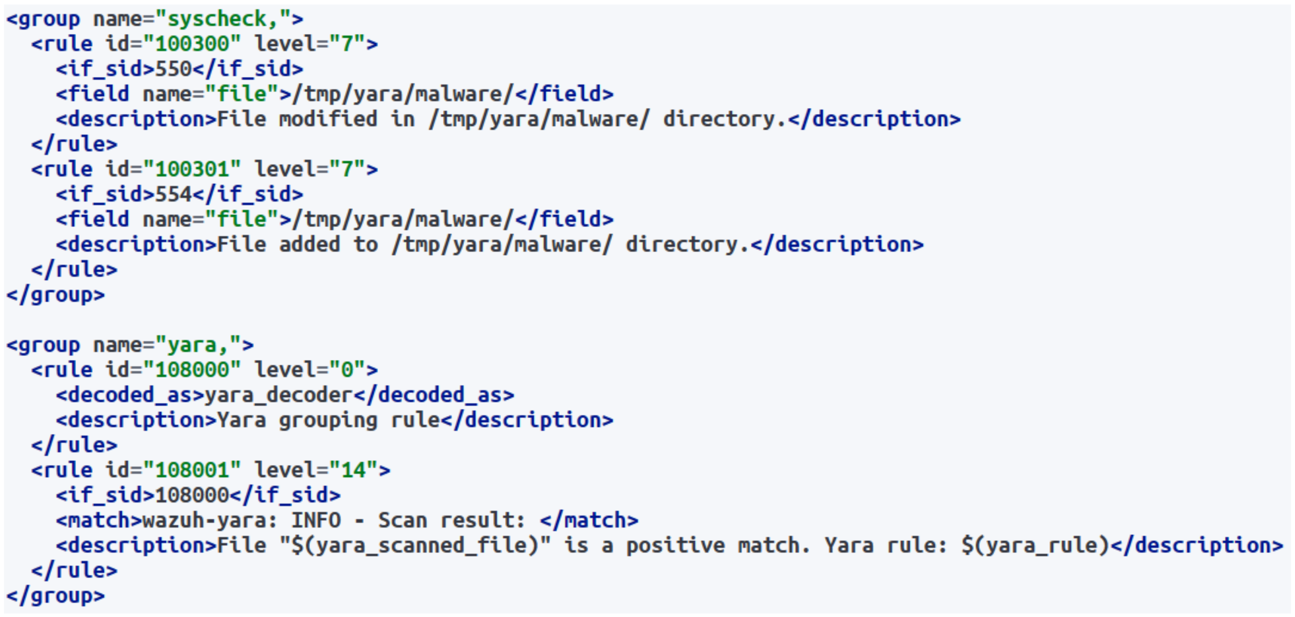
```
<directories realtime="yes">/tmp/yara/malware</directories>
```

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

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

Wazuh server

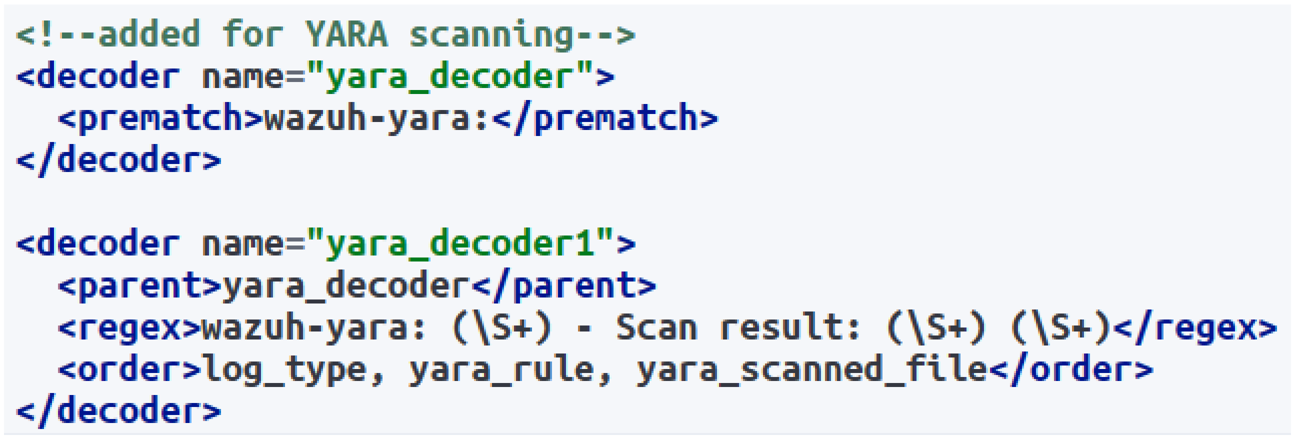
1. Add the following rules to the `/var/ossec/etc/rules/local_rules.xml` file.



```
<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 19: Custom Rules for YARA Scanning

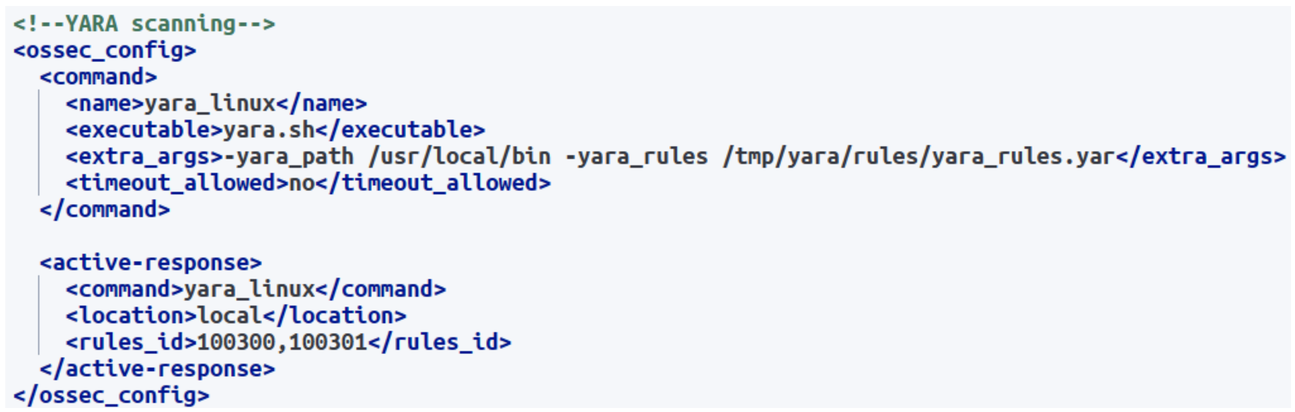
2. Add the following decoders to the Wazuh server `/var/ossec/etc/decoders/local/_decoder.xml` file. This allows extracting the information from YARA scan results.



```
<!--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 20: 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 21: Updating the Configuration for Active Response

SIMULATION

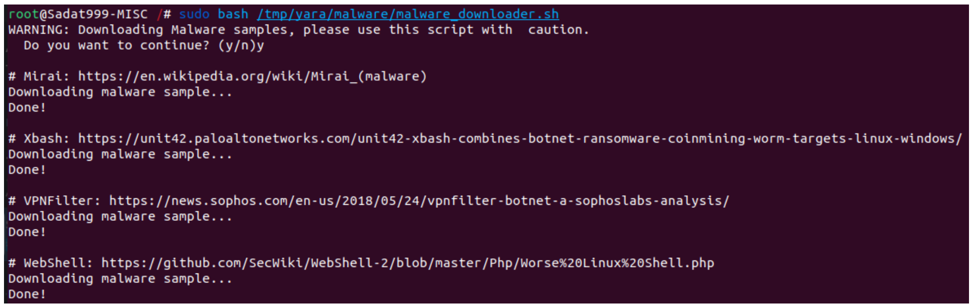
1. Create the script `/tmp/yara/malware/malware_downloader.sh` on the monitored endpoint to download malware samples:

```
#!/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 22: Downloading Four Malwares for YARA Scanning Simulation

DASHBOARD UPDATE

If we navigate like previously shown in 4.1.1, 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.

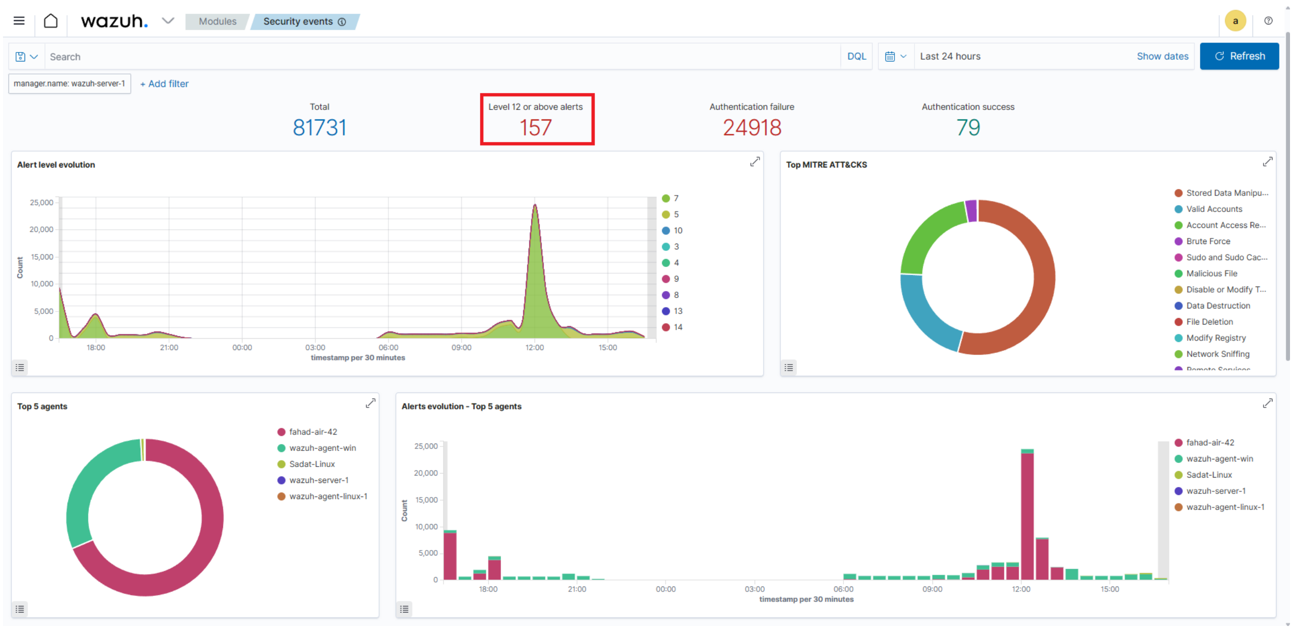
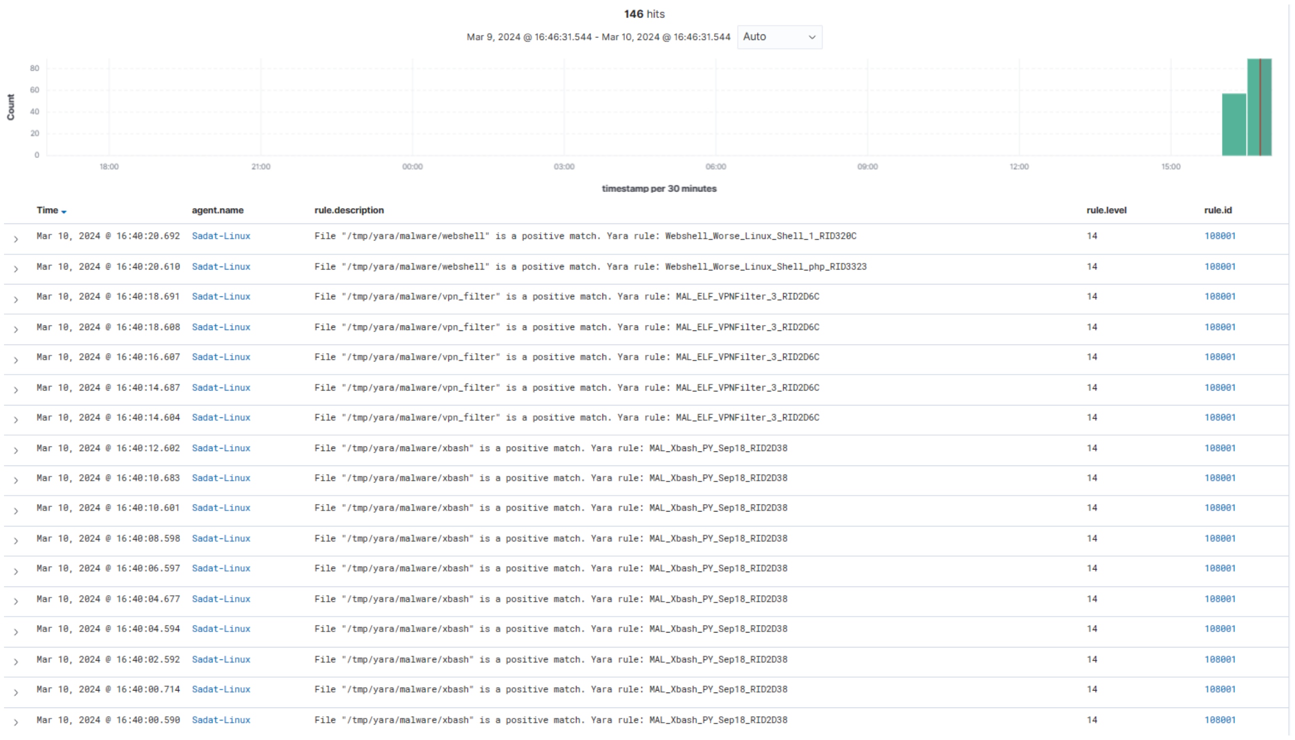


Figure 23: Dashboard after Downloading the Malwares

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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001
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	108001

Figure 24: Generated Alerts after Downloading Four Malwares

4.1.3 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
          ↪  8fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b85
          ↪  5/detection/f-e3b0c44298fc1c149afbf4c8996fb92427ae41e4
          ↪  649b934ca495991b7852b855-1668705588"
      },
      "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
         ↪  fb6489e54d471899f7db9d1663fc695ec2fe2a2c4538aabf651fd0
         ↪  f/detection/f-275a021bbfb6489e54d471899f7db9d1663fc695
         ↪  ec2fe2a2c4538aabf651fd0f-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 4.1.1. If that configuration is already done, the following no more needs to be added.

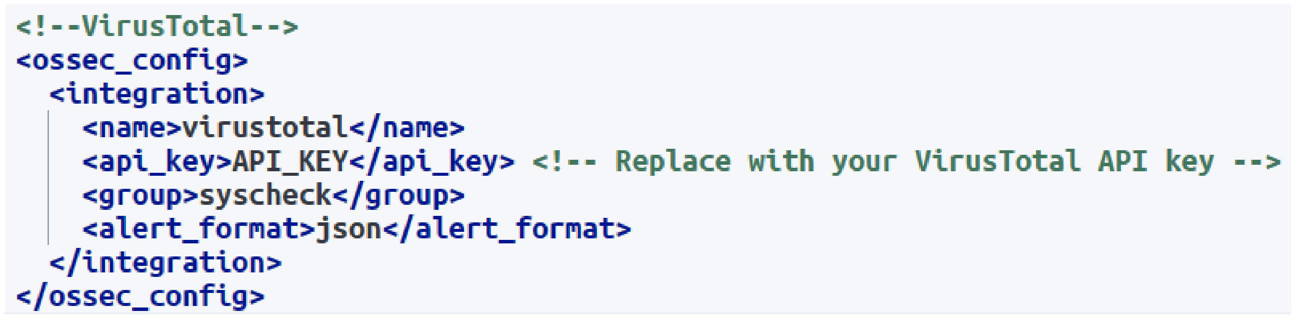
```
<syscheck>
  <directories check_all="yes" realtime="yes">/fim</directories>
</syscheck>
```

2. Restart the Wazuh manager.

```
systemctl restart wazuh-manager
```

Wazuh server

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



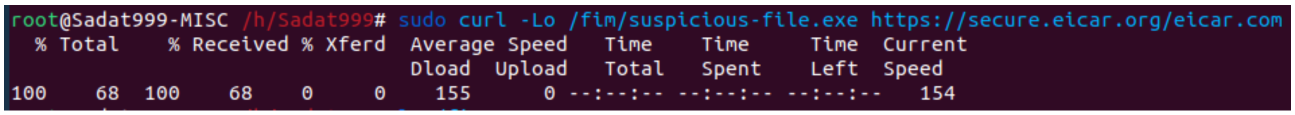
```
<!--VirusTotal-->
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key>API_KEY</api_key> <!-- Replace with your VirusTotal API key -->
    <group>syscheck</group>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```

Figure 25: 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 26: Downloading Malware for VirusTotal Checking

DASHBOARD UPDATE

We will again navigate to “Security events” tab as shown previously in 4.1.1. There we will see a new level 12 alert has been generated because of the malware download (the count was previously 179).

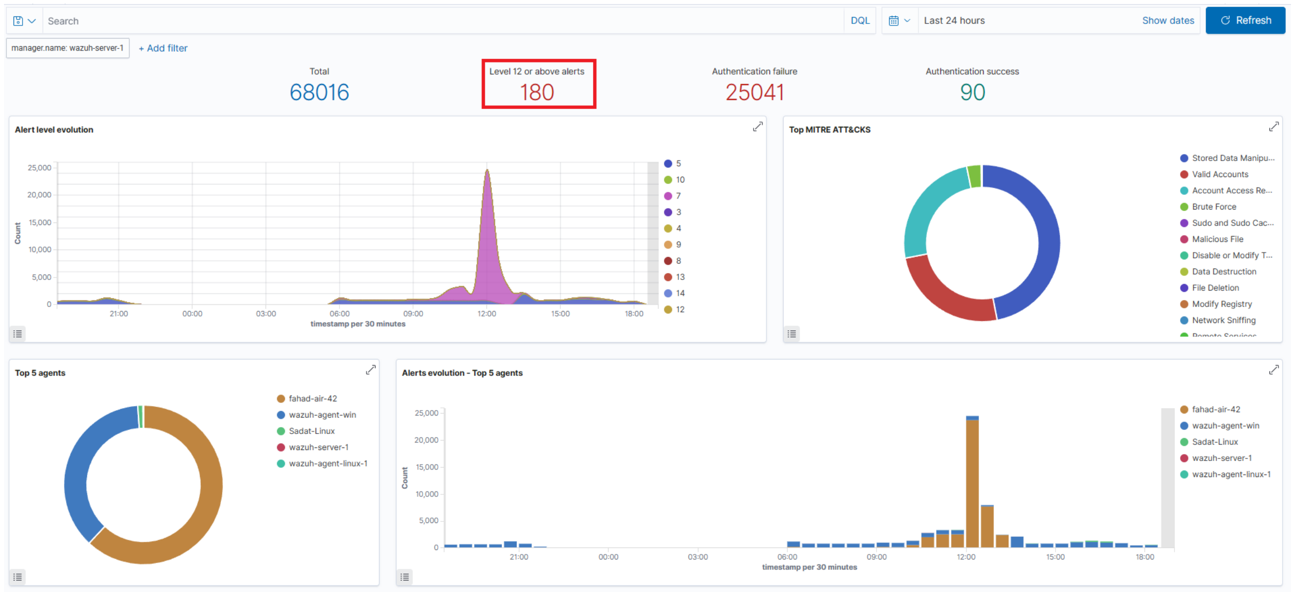


Figure 27: Dashboard after Malware Download

The generated alert goes as follows:

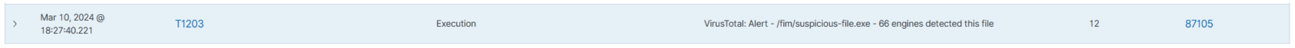

Mar 10, 2024 @ 18:27:40.221 T1203 Execution VirusTotal: Alert - /fim/suspicious-file.exe - 66 engines detected this file 12 87105

Figure 28: 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 LOG DATA ANALYSIS

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

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

4.2.1 HOW IT WORKS

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

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

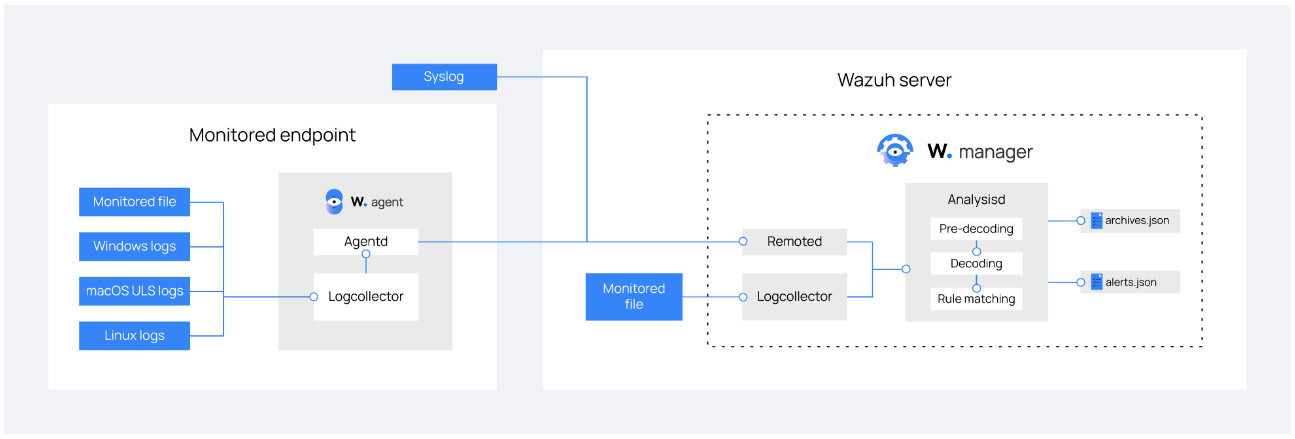


Figure 29: The flow of log data collection and analysis in Wazuh

The log data collection process consists of 3 essential phases:

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

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

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

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

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

4.2.2 CONFIGURATION

Wazuh supports two primary methods of log data collection.

USING 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>
```

4.2.3 SIMULATION

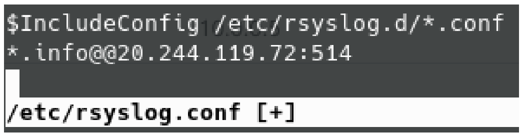
We demonstrate the following two use-cases of Log Data Analysis.

LINUX LOG DATA ANALYSIS USING RSYSLOG

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

Ubuntu endpoint

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

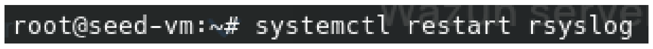


```
$IncludeConfig /etc/rsyslog.d/*.conf
*.info@@20.244.119.72:514

/etc/rsyslog.conf [+]
```

Figure 30: `rsyslog` configuration

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

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

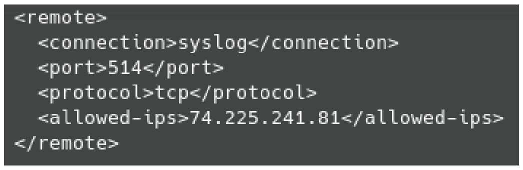


```
root@seed-vm:~# systemctl restart rsyslog
```

Figure 31: 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 32: 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.

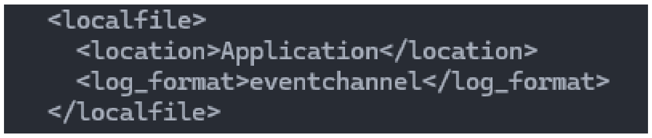
We test the configuration in the next sub-section.

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.



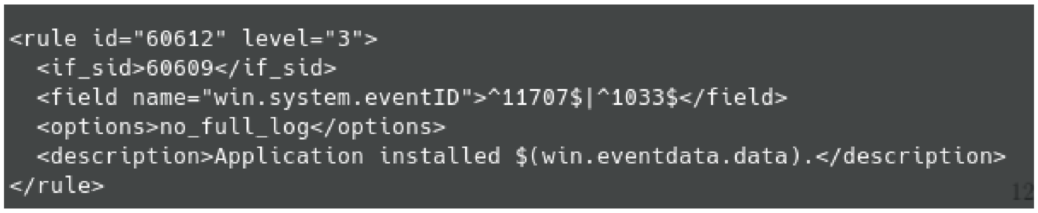
```
<localfile>
  <location>Application</location>
  <log_format>eventchannel</log_format>
</localfile>
```

Figure 33: 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>
```

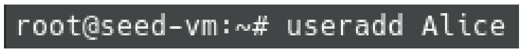
Figure 34: Wazuh server configuration

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

4.2.4 DASHBOARD UPDATE

LINUX LOG DATA ANALYSIS USING `RSYSLOG`

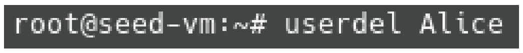
1. We add the new user `Alice`



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

Figure 35: Adding new user

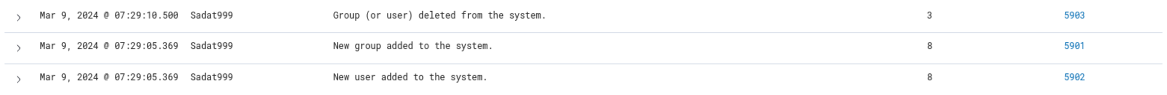
2. We delete the user `Alice`



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

Figure 36: Deleting the new user

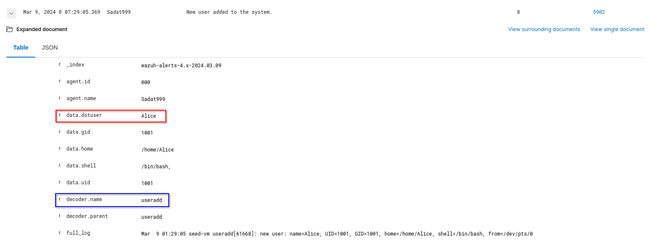
3. We navigate to the `Modules > Security events` tab in the Wazuh Dashboards to view the alerts.



```
> 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 37: Alerts for user/group creation and deletion

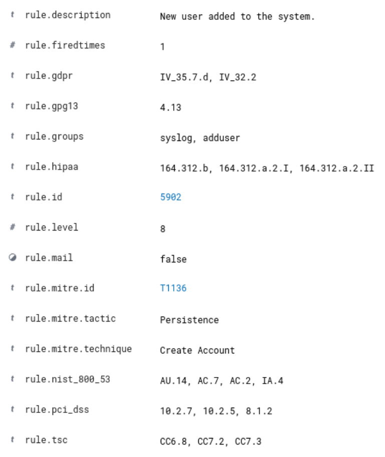
4. We expand the alert to see more details.



```
Mar 9, 2024 @ 07:29:05.369  Sadat999    New user added to the system.    8    5902
```

```
t _index          wazuh-alerts-4.x-2024.03.09
t agent.id        000
t agent.name      Sadat999
t data.dstuser    Alice
t data.gid        1001
t data.home       /home/Alice
t data.shell      /bin/bash,
t data.uid        1001
t decoder.name    useradd
t decoder.parent  useradd
t full_log        Mar  9 01:29:05 seed-vm useradd[61668]: new user: name=Alice, UID=1001, GID=1001, home=/home/Alice, shell=/bin/bash, from=/dev/pts/0
```

Figure 38: The name of the new user (red), the decoder used to process the log (blue)



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

Figure 39: Details of the rule used to generate this alert

WINDOWS LOG DATA ANALYSIS USING WAZUH AGENT

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

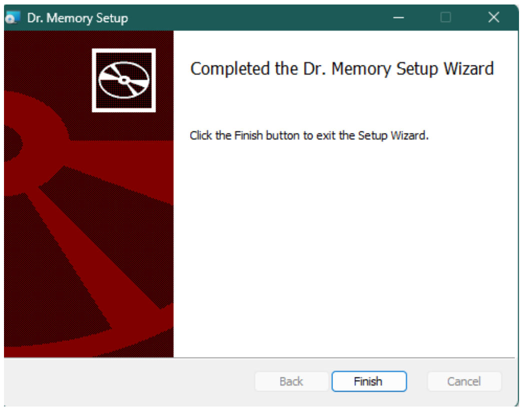
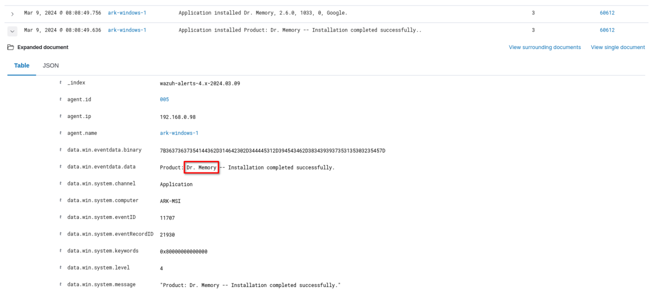

Dr. Memory Setup
Completed the Dr. Memory Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

Figure 40: Installation of Dr. Memory

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



```
Mar 9, 2024 @ 08:08:49.756  ark-windows-1   Application installed Dr. Memory, 2.6.0, 1033, 0, Google.                         3   60612
Mar 9, 2024 @ 08:08:49.636  ark-windows-1   Application installed Product: Dr. Memory -- Installation completed successfully..  3   60612
```

```
_index                       wazuh-alerts-4.x-2024.03.09
agent.id                     005
agent.ip                     192.168.0.98
agent.name                   ark-windows-1
data.win.eventdata.binary    7B36373637354144362D314642302D344445312D394543462D38343939373531353032235457D
data.win.eventdata.data      Product: Dr. Memory -- Installation completed successfully.
data.win.system.channel      Application
data.win.system.computer     ARK-MSI
data.win.system.eventID      11707
data.win.system.eventRecordID 21930
data.win.system.keywords     0x80000000000000
data.win.system.level        4
data.win.system.message      "Product: Dr. Memory -- Installation completed successfully."
```

Figure 41: Alert for new software installation

5 SOURCE CODE

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

5.1 REPOSITORIES

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

WAZUH

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

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

WAZUH DASHBOARD

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

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

WAZUH DASHBOARD PLUGINS

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

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

WAZUH INDEXER

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

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

5.2 COMPILING THE FRONT-END FROM SOURCE

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

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

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

2. Configure the docker environment.

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

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

   - 8 GB of RAM
   - 4 CPU Cores

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

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

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

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

6. Clone the repository.

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

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

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

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

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

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

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

   where,

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

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

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

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

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

12. Install the dependencies using:

```
yarn install
```

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

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

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

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

7 REFERENCE

- The Official Wazuh Documentation
- Wazuh GitHub Repositories