HISNEO Casablanca

École Nationale des Sciences Appliquées
Oujda

Prepared by: ESSATAB ACHRAF
On September 16, 2024
Supervised by: Abdellah Sghir

Integrating Watchguard Firebox Firewall with Wazuh Server And Use Case Implementation

Contents

1 Acknowledgements

I would like to express my deepest gratitude to HISNEO for providing me with the opportunity to complete my internship within their innovative cybersecurity services environment. This one-month experience has been incredibly enriching, allowing me to apply the theoretical knowledge gained during my studies and develop new practical skills in a real-world context.

A special thanks to Abdellah Sghir, who supervised me throughout the internship. Their guidance, feedback, and constant support were invaluable to the success of this internship.

This document serves as a comprehensive report of my one-month internship, outlining the tasks I was assigned, the skills I developed, and the insights I gained during my time at Hisneo.

The internship was completed as part of my second year of engineering studies in the field of information security and cybersecurity at ENSA Oujda.

2 HISNEO: An Innovative Moroccan Cybersecurity Services Company

2.1 Introduction

Founded in 2022, HISNEO is a Pure Player in the field of cybersecurity, created by two experts with more than 20 years of experience in the industry. HISNEO's mission is to support its clients in protecting sensitive information and systems that are essential to their activities. The company aims to help organizations build solid digital trust, adapting to the evolving risks and uses that come with the digital transformation of society.

2.2 Mission and Vision

HISNEO places a strong emphasis on guiding its clients through the complexities of cybersecurity. Its mission is to protect critical digital assets while promoting digital trust. The company's approach is shaped by the ever-changing landscape of cybersecurity threats and the new challenges that arise with the increasing digitization of all aspects of modern business and society.

Through its services, HISNEO seeks to:

Support Clients: In safeguarding their sensitive information and vital systems. Build Digital Trust: By ensuring a secure digital environment for business operations. Adapt to Digital Transformation: By mitigating risks associated with new technologies and practices.

2.3 Cyber Defense Consulting Services

HISNEO offers a range of consulting services designed to help organizations enhance their cybersecurity posture. These services aim to assist clients in selecting and applying the best practices needed to protect their digital infrastructure. The consulting services include the following:

Cyber Defense Strategies: HISNEO advises organizations on selecting robust cybersecurity frameworks that align with industry standards. Security Assessment: The company evaluates the current level of security within a client's information systems, identifying potential vulnerabilities and suggesting improvements. Strengthening Cybersecurity: HISNEO helps clients reinforce their cybersecurity measures, ensuring they are better prepared to prevent and respond to cyber threats.

2.4 Comprehensive IT Security Strategy

A robust IT security strategy is vital for any organization, and HISNEO emphasizes the importance of controlling three key pillars:

People: Training and raising awareness within teams is a critical component of a strong cybersecurity framework. HISNEO offers specialized programs to educate and empower employees to recognize and respond to potential threats. Processes: Streamlining and improving the processes that govern information security ensures better risk management and quicker responses to incidents. Technology: HISNEO provides cutting-edge technological solutions to strengthen defenses against evolving cyber threats.

2.5 Managed Cybersecurity Services

HISNEO also provides a suite of managed cybersecurity services to give clients ongoing protection. These services include:

Monitoring: Continuous surveillance of systems to detect and respond to threats in real-time. Responsiveness: Quick action to counter and prevent cyber attacks before they can inflict significant damage. By offering these managed services, HISNEO ensures that clients can focus on their core business activities while relying on HISNEO's expertise to maintain security and prevent disruptions.

2.6 Conclusion

HISNEO has established itself as a forward-thinking, innovative cybersecurity company. With its experienced leadership and client-focused services, it plays a crucial role in helping organizations navigate the complex world of cybersecurity. The company's approach, combining consulting, strategy, and managed services, ensures that its clients are well-equipped to face the challenges of an increasingly digital world.

3 Testing On The Cloud

As a good practice, before deploying any tool on a production environment it must be tested beforehand. That's what this section is about, testing the core components of wazuh, an open source SIEM and XDR tool in a testing environment. We chose to take azure cloud virtual machines as our testing environment. The reason is that we weren't equipped with the necessary computing resources that fit wazuh requirements.

3.1 Setting an Azure Virtual Machine For Wazuh server

Microsoft azure is known for its ease of use. Going to the portal[1], we find a sandwich menu. Click the menu and choose All Resources then Create a new resource, you will pop-up with the page illustrated below. Under virtual machine, press create button and you will be presented to configure your new

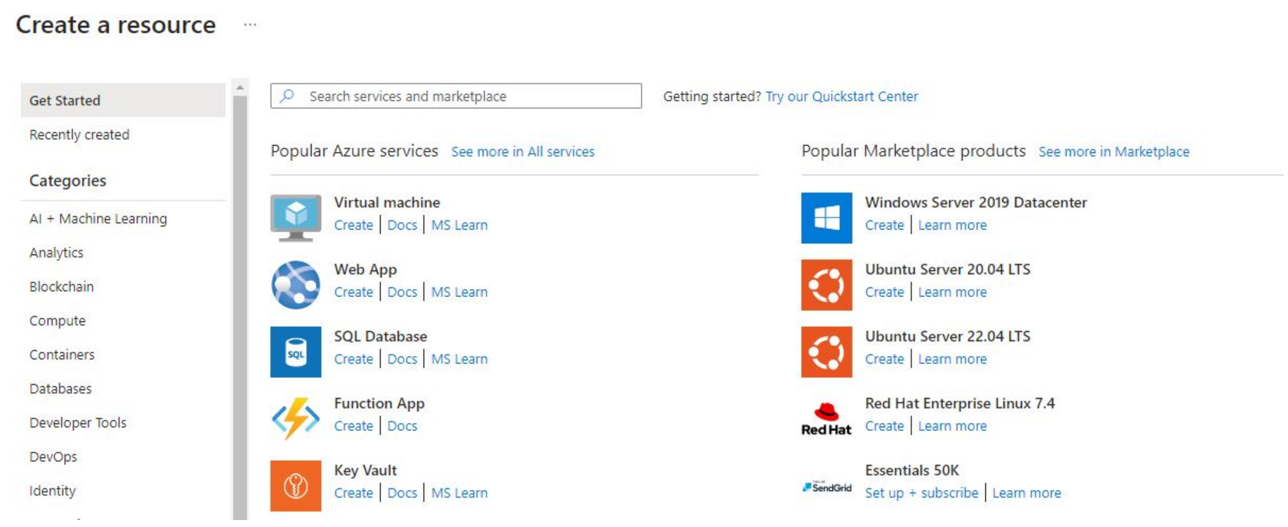
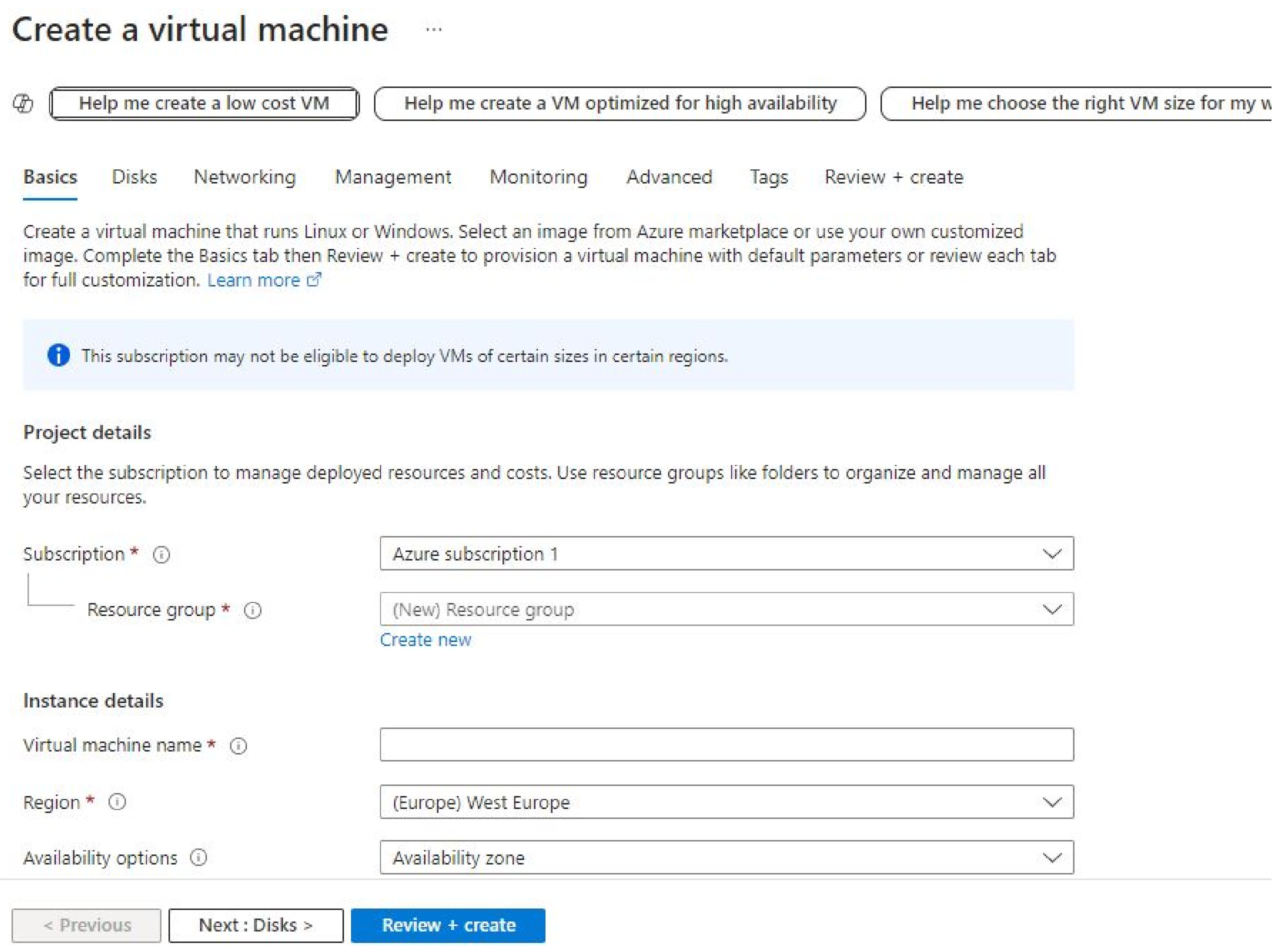

Create a resource
Get Started
Recently created
Categories
AI + Machine Learning
Analytics
Blockchain
Compute
Containers
Databases
Developer Tools
DevOps
Identity
Search services and marketplace
Getting started? Try our Quickstart Center
Popular Azure services See more in All services
Virtual machine Create | Docs | MS Learn
Web App Create | Docs | MS Learn
SQL Database Create | Docs | MS Learn
Function App Create | Docs
Key Vault Create | Docs | MS Learn
Popular Marketplace products See more in Marketplace
Windows Server 2019 Datacenter Create | Learn more
Ubuntu Server 20.04 LTS Create | Learn more
Ubuntu Server 22.04 LTS Create | Learn more
Red Hat Enterprise Linux 7.4 Create | Learn more
Essentials 50K Set up + subscribe | Learn more

Figure 1: Resource creation page

virtual machine as shown in the figure 2.

For testing purposes, we chose an OS image of Ubuntu Server 24.04 LTS, as this is widely used on production environments. The underlying computing architecture is known as *size* in the jargon of azure, is set to: Standard_D4s_v3 - 4 vcpus, 16 GiB memory. For networks with many log ingestion per second, we need at least more RAM.

[1]The web user interface


Create a virtual machine

Help me create a low cost VM
Help me create a VM optimized for high availability
Help me choose the right VM size for my w
Basics
Disks
Networking
Management
Monitoring
Advanced
Tags
Review + create

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization. Learn more

This subscription may not be eligible to deploy VMs of certain sizes in certain regions.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure subscription 1
Resource group *	(New) Resource group
	Create new

Instance details

Virtual machine name *	
Region *	(Europe) West Europe
Availability options	Availability zone

Figure 2: VM creation page

3.2 Installing the Core Components

There are three components that constitute the fundamental setup of wazuh. Those components can be installed either on a single node or they can be distributed among a cluster of nodes and the choice is determined by the requirements of the user, especially the number of the monitored endpoints. The components are:

3.2.1 Wazuh Manager

This piece of the server receives the data from the monitored endpoint and performs analysis such as decoding the received data and match them against a set of predefined rules. The manager include another sub-component, Filebeat, responsible of storing data into a database of another component called Wazuh-indexer.

3.2.2 Wazuh Indexer

It is a search and analytics engine based on Opensearch. The main purpose of this component is to fulfill read and write queries coming from other components.

3.2.3 Wazuh Dashboard

provide a web-based interface that facilitates the server configurations and event visualization. Each component can be duplicated as much as the user is required to, for purposes such as fault-tolerance and high performance. Wazuh can also be installed in a containerized environment. The interconnections between components is depicted in figure 4. The local network in the figure represent the subnet where the virtual machines of two monitored systems and the SIEM pertain. One monitored system is located on the internet.

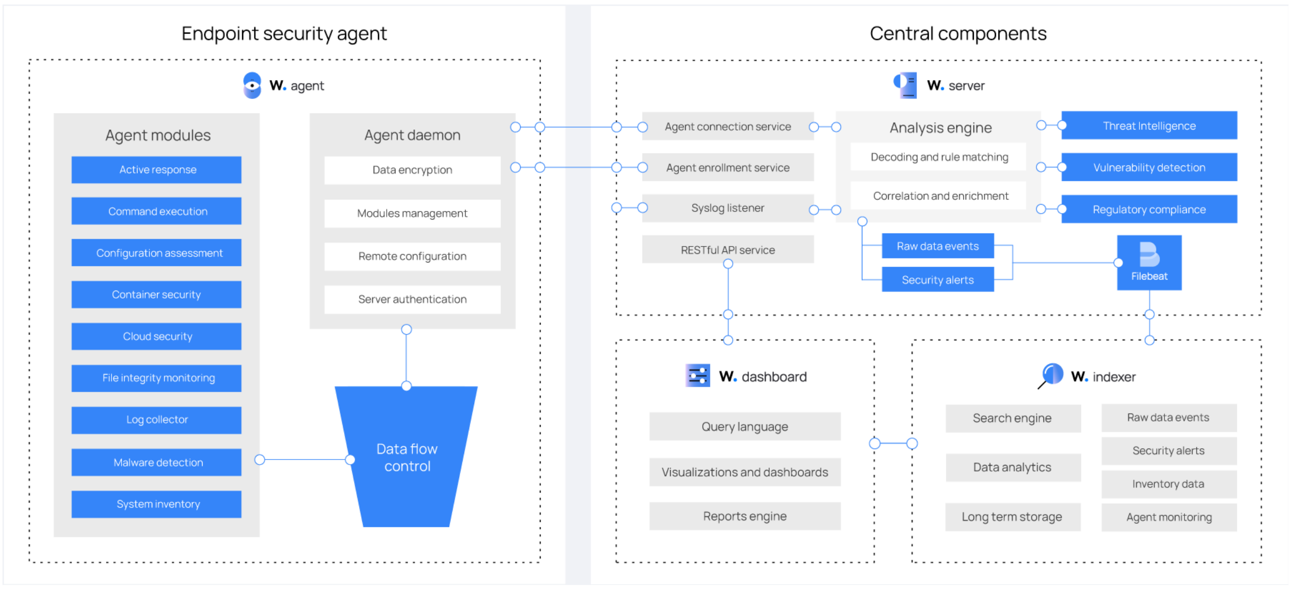


Figure 3: Wazuh Core Components Communication

3.2.4 Server Installation

Before starting the installation process, we need to configure the networking state of the virtual machine that would host wazuh server. Wazuh server must have a static IP, this is because one of our monitored systems isn't located on the same subnet. To solve the problem we used the DNS service provided by azure, so we mapped the monitored systems with the server's domain name instead of its IP address. If we are behind a firewall some ports must be open. For inbound connections, port 443 would be used to access the dashboard, ports 1514 and 1515 would be used to communicate with the monitored system and subscribe it to the server. Also port 514 udp is used in case we are monitoring a syslog based system. Go to the official site of wazuh, and follow the quick start[2] installation guide. This would install all the components in a single node.

[2]https://documentation.wazuh.com/current/quickstart.html

3.3 Agents Installation

For this project I tried the SIEM tool before migrating it to the production environment of HISNEO. The trial was a full emulation of what will be in the production environment. The test environment includes the monitored systems along with the monitoring tools (see the figure below). The wazuh server is responsible for collecting logs it receives from the monitored endpoints, parsing them, analysing them and respond in customized manner to user-specified events. Three systems were con-

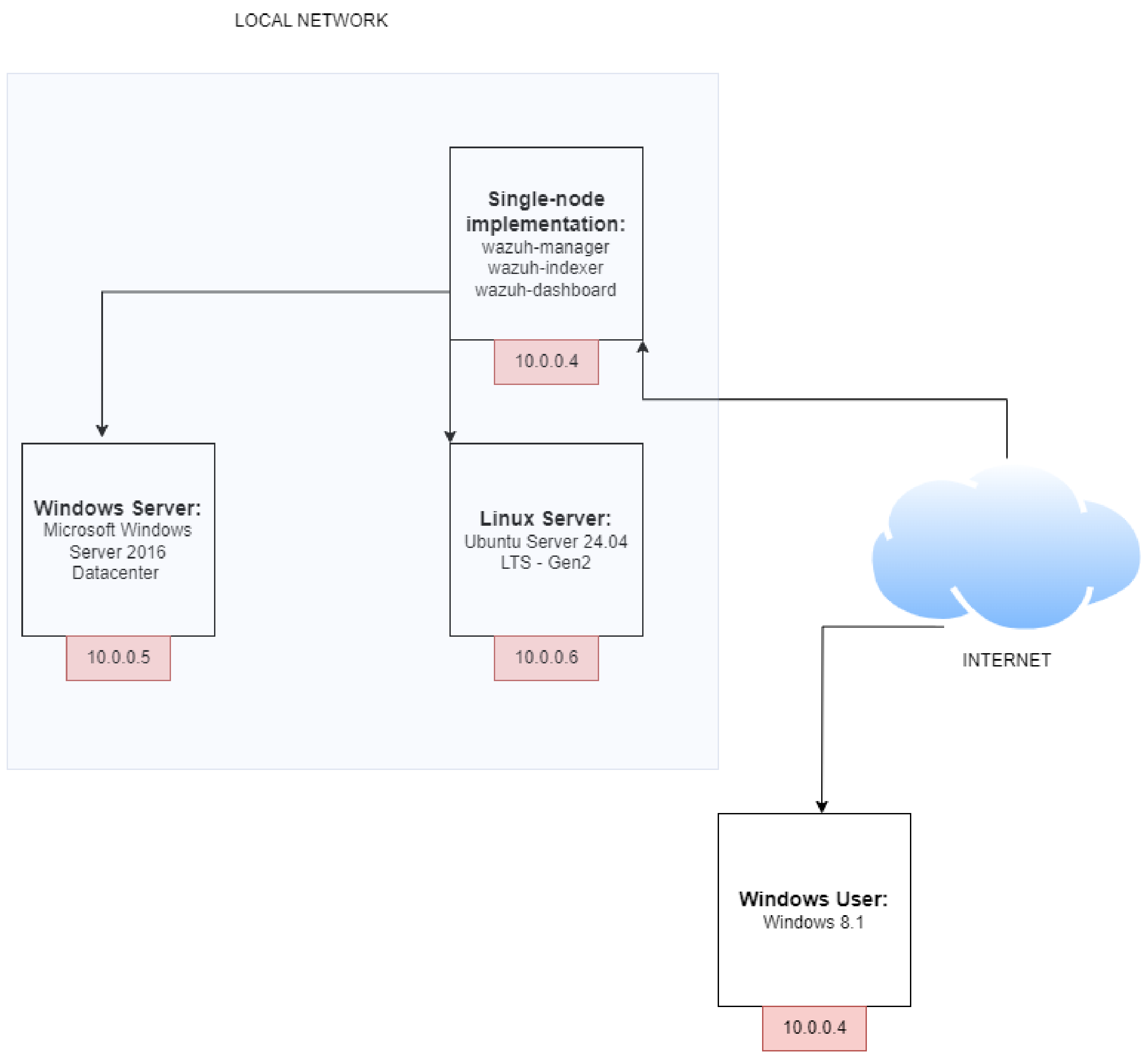
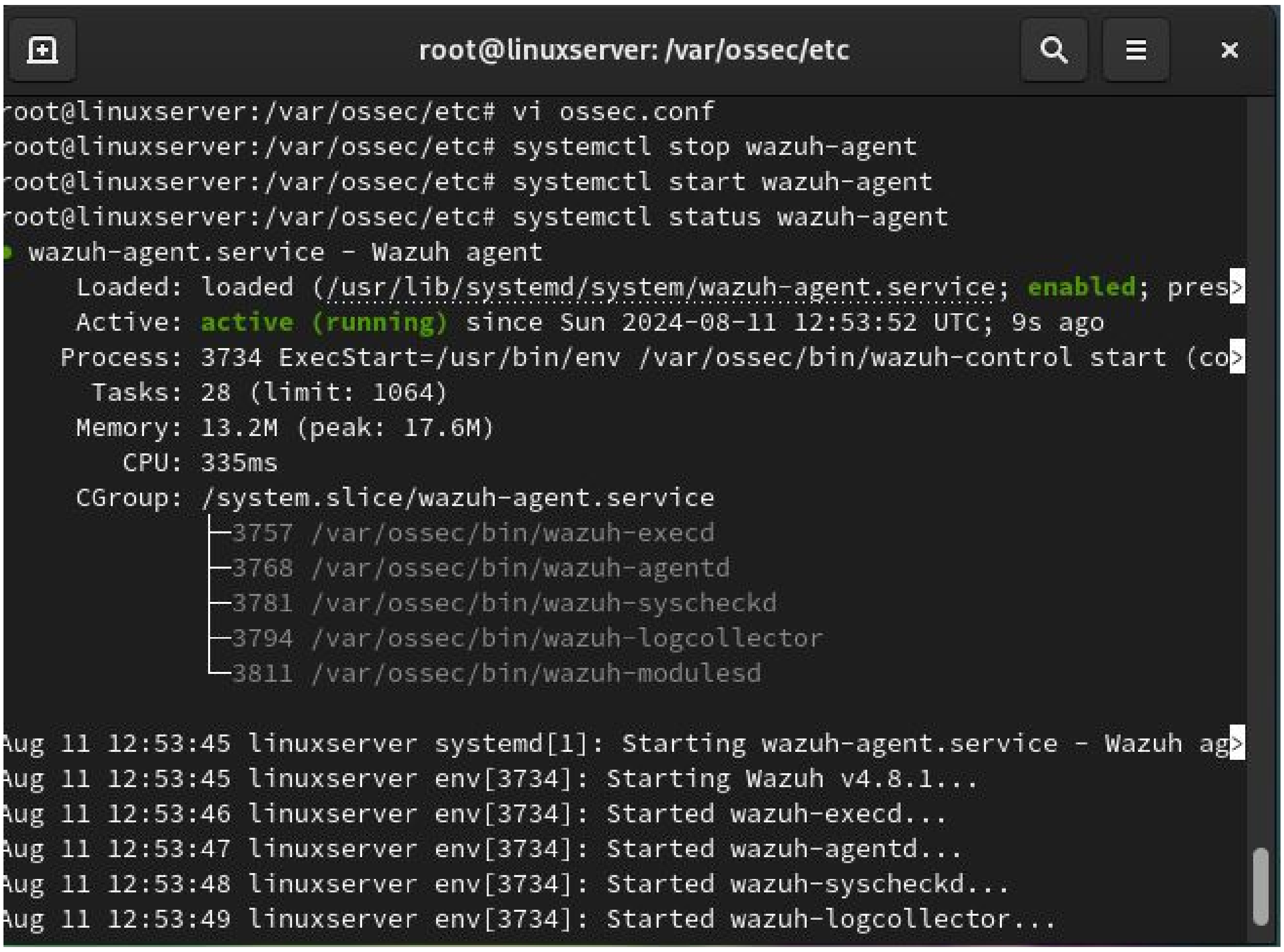


Figure 4: The overall architecture

nected with the server, Ubuntu Server 24.04 LTS - Gen2, Microsoft Windows Server 2016 Datacenter and a windows 8.1 workstation. Each of those systems has a lightweight program installed on it called agent. The Wazuh agent is multi-platform and runs on the endpoints that the user wants to monitor. It communicates with the Wazuh server, sending data in near real-time through an encrypted and authenticated channel. The Wazuh agent provides the following key features:

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

The installation in a linux system differs from a system running windows. For Linux flavors, we followed the guide in the official site.[3] To make sure it is installed successfully run
systemctl status wazuh-agent
This gave what is depicted in figure 5.



root@linuxserver: /var/ossec/etc

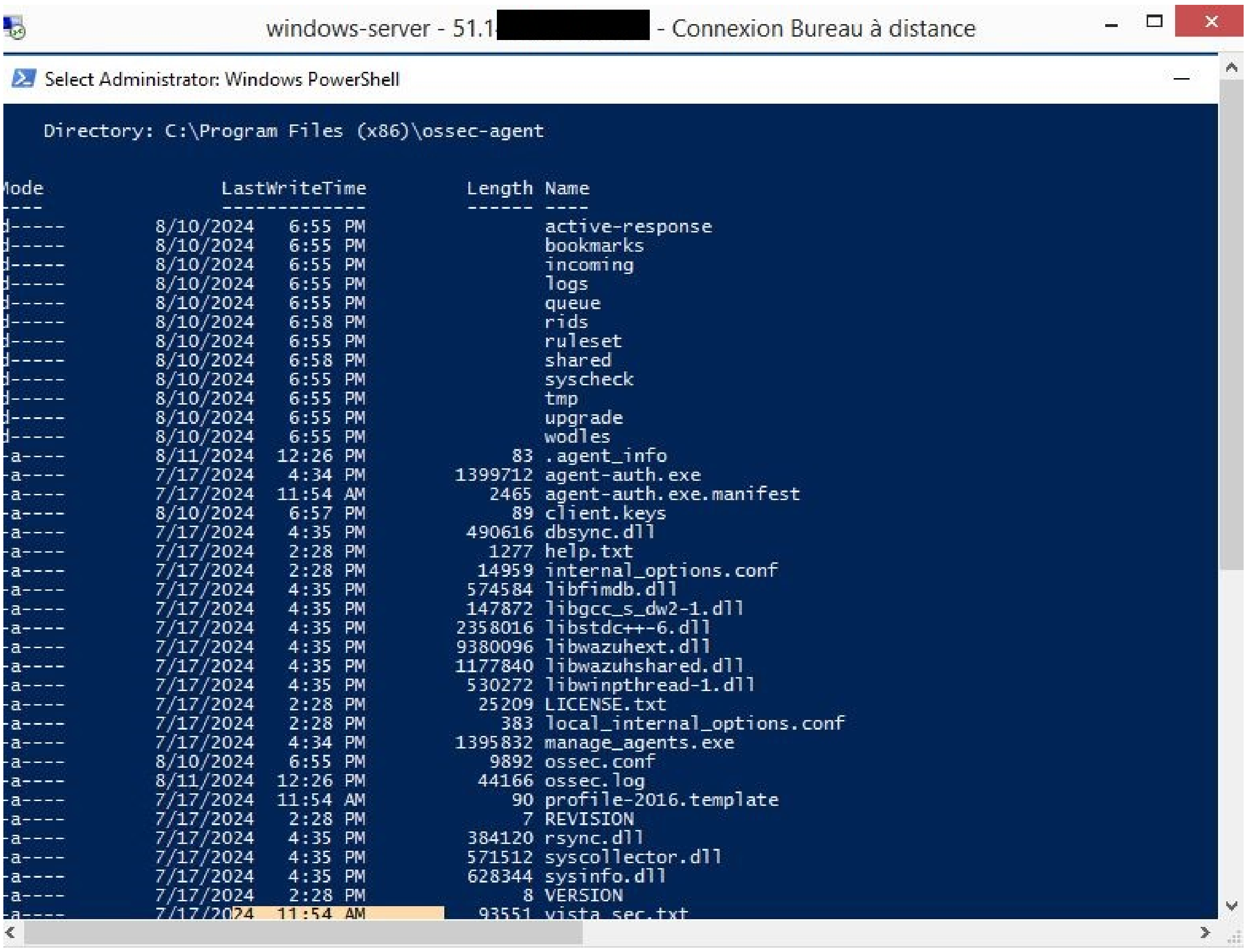
```
oot@linuxserver:/var/ossec/etc# vi ossec.conf
oot@linuxserver:/var/ossec/etc# systemctl stop wazuh-agent
oot@linuxserver:/var/ossec/etc# systemctl start wazuh-agent
oot@linuxserver:/var/ossec/etc# systemctl status wazuh-agent
 wazuh-agent.service - Wazuh agent
     Loaded: loaded (/usr/lib/systemd/system/wazuh-agent.service; enabled; pres>
     Active: active (running) since Sun 2024-08-11 12:53:52 UTC; 9s ago
    Process: 3734 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (co>
      Tasks: 28 (limit: 1064)
     Memory: 13.2M (peak: 17.6M)
        CPU: 335ms
     CGroup: /system.slice/wazuh-agent.service
             ├─3757 /var/ossec/bin/wazuh-execd
             ├─3768 /var/ossec/bin/wazuh-agentd
             ├─3781 /var/ossec/bin/wazuh-syscheckd
             ├─3794 /var/ossec/bin/wazuh-logcollector
             └─3811 /var/ossec/bin/wazuh-modulesd

Aug 11 12:53:45 linuxserver systemd[1]: Starting wazuh-agent.service - Wazuh ag>
Aug 11 12:53:45 linuxserver env[3734]: Starting Wazuh v4.8.1...
Aug 11 12:53:46 linuxserver env[3734]: Started wazuh-execd...
Aug 11 12:53:47 linuxserver env[3734]: Started wazuh-agentd...
Aug 11 12:53:48 linuxserver env[3734]: Started wazuh-syscheckd...
Aug 11 12:53:49 linuxserver env[3734]: Started wazuh-logcollector...
```

Figure 5: Wazuh-agent in linux system

For the workstation and Windows server 2016 the installation follows its specific guide.[4] All the modules part of wazuh-agent would have a place under the directory ossec-agent as depicted in figure 6.

[3] https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-linux.html

[4] https://documentation.wazuh.com/current/installation-guide/wazuh-agent/wazuh-agent-package-windows.html



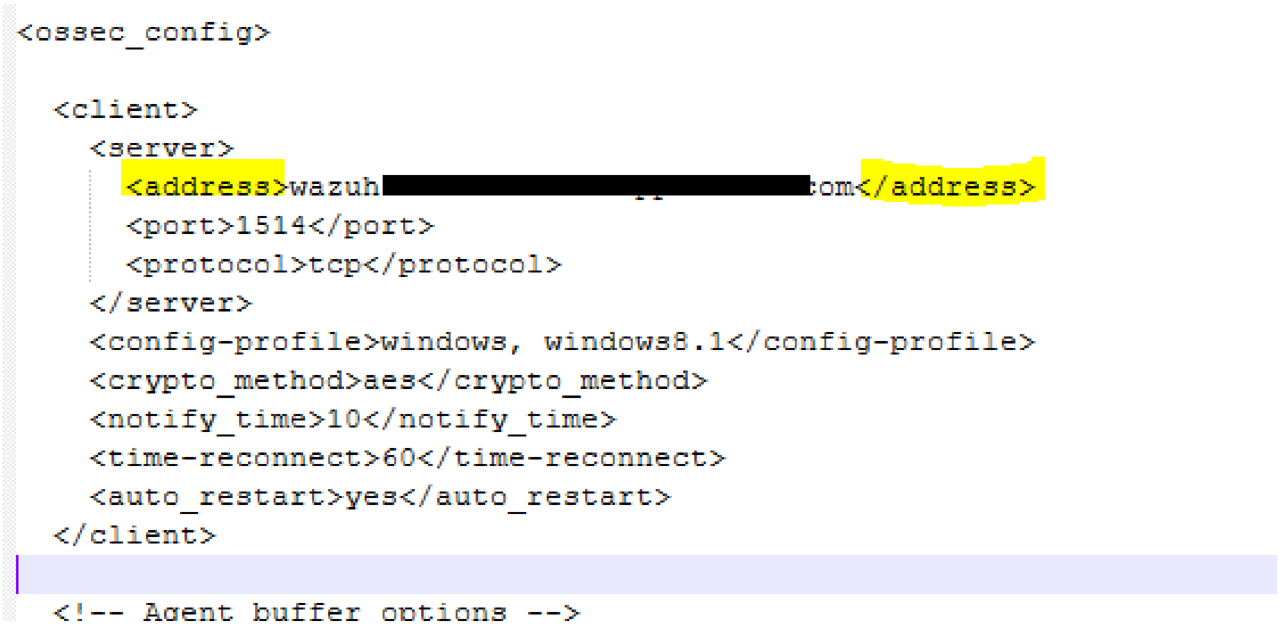
windows-server - 51.1[redacted] - Connexion Bureau à distance

Select Administrator: Windows PowerShell

```
    Directory: C:\Program Files (x86)\ossec-agent


ode                 LastWriteTime         Length Name
----                -------------         ------ ----
d-----        8/10/2024   6:55 PM                active-response
d-----        8/10/2024   6:55 PM                bookmarks
d-----        8/10/2024   6:55 PM                incoming
d-----        8/10/2024   6:55 PM                logs
d-----        8/10/2024   6:55 PM                queue
d-----        8/10/2024   6:58 PM                rids
d-----        8/10/2024   6:55 PM                ruleset
d-----        8/10/2024   6:58 PM                shared
d-----        8/10/2024   6:55 PM                syscheck
d-----        8/10/2024   6:55 PM                tmp
d-----        8/10/2024   6:55 PM                upgrade
d-----        8/10/2024   6:55 PM                wodles
-a----        8/11/2024  12:26 PM             83 .agent_info
-a----        7/17/2024   4:34 PM        1399712 agent-auth.exe
-a----        7/17/2024  11:54 AM           2465 agent-auth.exe.manifest
-a----        8/10/2024   6:57 PM             89 client.keys
-a----        7/17/2024   4:35 PM         490616 dbsync.dll
-a----        7/17/2024   2:28 PM           1277 help.txt
-a----        7/17/2024   2:28 PM          14959 internal_options.conf
-a----        7/17/2024   4:35 PM         574584 libfimdb.dll
-a----        7/17/2024   4:35 PM         147872 libgcc_s_dw2-1.dll
-a----        7/17/2024   4:35 PM        2358016 libstdc++-6.dll
-a----        7/17/2024   4:35 PM        9380096 libwazuhext.dll
-a----        7/17/2024   4:35 PM        1177840 libwazuhshared.dll
-a----        7/17/2024   4:35 PM         530272 libwinpthread-1.dll
-a----        7/17/2024   2:28 PM          25209 LICENSE.txt
-a----        7/17/2024   2:28 PM            383 local_internal_options.conf
-a----        7/17/2024   4:34 PM        1395832 manage_agents.exe
-a----        8/10/2024   6:55 PM           9892 ossec.conf
-a----        8/11/2024  12:26 PM          44166 ossec.log
-a----        7/17/2024  11:54 AM             90 profile-2016.template
-a----        7/17/2024   2:28 PM              7 REVISION
-a----        7/17/2024   4:35 PM         384120 rsync.dll
-a----        7/17/2024   4:35 PM         571512 syscollector.dll
-a----        7/17/2024   4:35 PM         628344 sysinfo.dll
-a----        7/17/2024   2:28 PM              8 VERSION
-a----        7/17/2024  11:54 AM          93551 vista_sec.txt
```

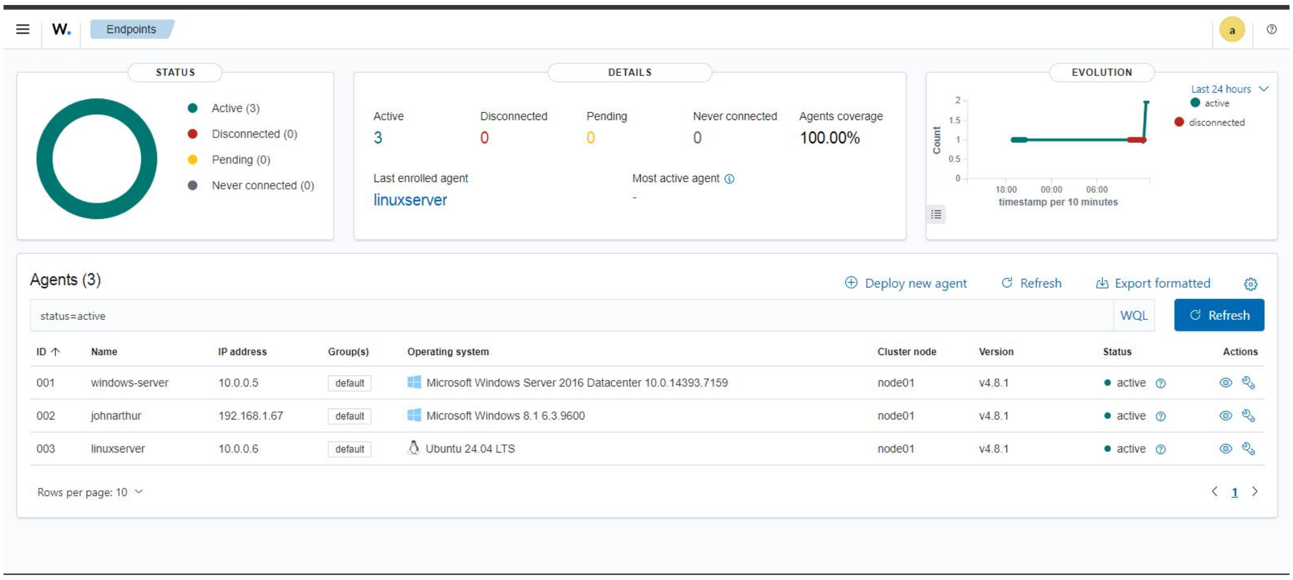
Figure 6: Wazuh agent in windows system

Upon installation, the agents initiate a communication with the wazuh server on port 1515 to be enrolled. The server would share a public key with the agent for encrypted communication, and the connection starts. An agent would send log data and execute workloads related to its modules as it is configured. The configuration is set on ossec/ossec.conf file that follows an XML format. When the IP/Domain name of the server is set in this file under the address tag (see figure 7), we would start seeing events happening in the agent in the dashboard of wazuh as in figure 8.



```
<ossec_config>

  <client>
    <server>
      <address>wazuh[redacted]om</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>windows, windows8.1</config-profile>
    <crypto_method>aes</crypto_method>
    <notify_time>10</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
  </client>

  <!-- Agent buffer options -->
```

Figure 7: Address tag in ossec.conf file



Active	Disconnected	Pending	Never connected	Agents coverage
3	0	0	0	100.00%

Last enrolled agent: linuxserver
Most active agent: -

Agents (3)

Deploy new agent · Refresh · Export formatted

status=active

ID	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status	Actions
001	windows-server	10.0.0.5	default	Microsoft Windows Server 2016 Datacenter 10.0.14393.7159	node01	v4.8.1	active	
002	johnarthur	192.168.1.67	default	Microsoft Windows 8.1 6.3.9600	node01	v4.8.1	active	
003	linuxserver	10.0.0.6	default	Ubuntu 24.04 LTS	node01	v4.8.1	active	

Rows per page: 10

Figure 8: Connected endpoints on wazuh dashboard

4 Passing to the Production Environment

At the production environment the main purpose was to achieve two goals:

- Monitor network flow passing through the company's firewall.
  The source of all events that the server will deal with was a network appliance. WatchGuard Firebox is a firewall appliance that provides network security. Firebox is a powerful network security device that controls all traffic between the external network and the trusted network. Installed in the company's environment, Watchguard's firewall was configured to send log messages

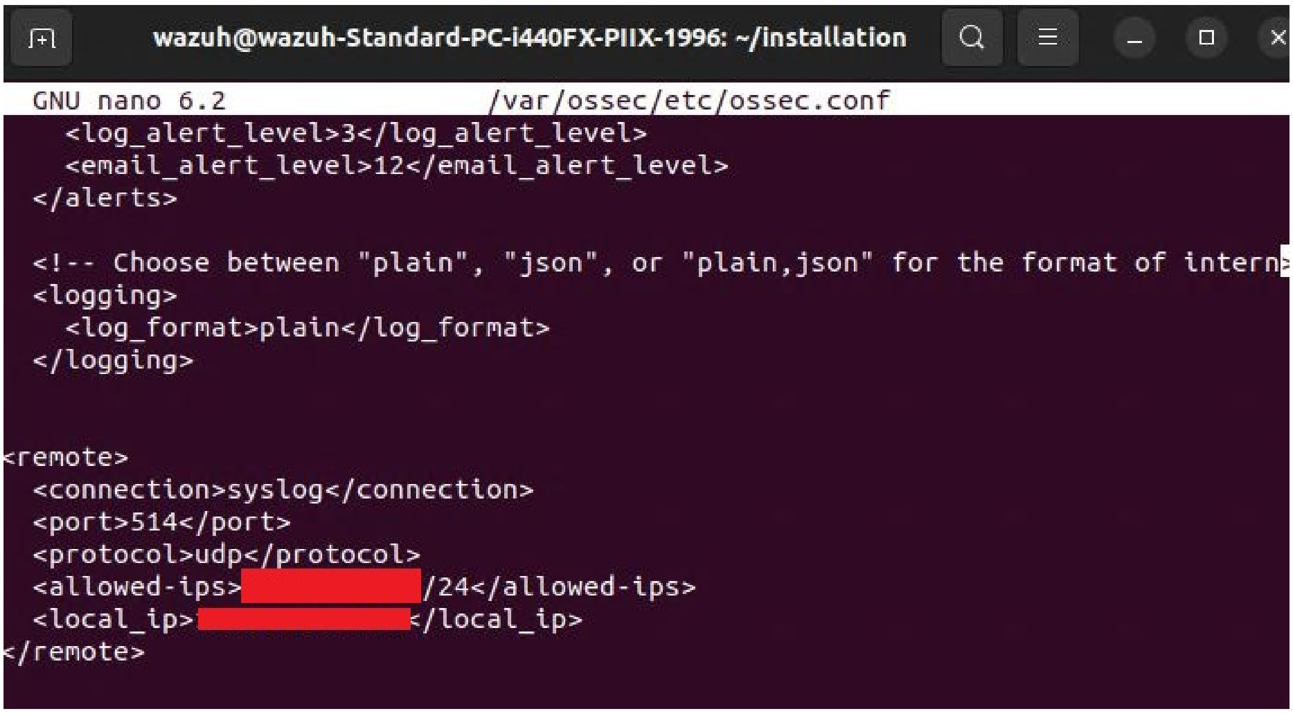
to wazuh server via a protocol called syslog.

- Enrich alerts with data that align with security frameworks and regulatory standards

4.1 Receiving Syslog Messages

4.1.1 Syslog protocol

This protocol provides a transport to allow a device to send event notification messages across IP networks to event message collectors, also known as syslog servers. The protocol is simply designed to transport these event messages from the generating device to the collector. Figure 10 shows an example of a syslog message generated by Firebox.
According to RFC 3164, a syslog message starts by a priority value, an integer calculated using two integers, the facility number and the severity number. Following the priority number is a timestamp indicating the date and the time of the event. Next to the timestamp is the IP address or the hostname of the sender. Then a tag indicating the process that generated the log can be followed by a process identifier (PID) between square brackets. And the message starts after the discussed metadata:
<PRI> TIMESTAMP HOSTNAME TAG[PID]: MESSAGE
The next step after I installed wazuh server at HISNEO's environment was to enable it to receive syslog messages generated by the firewall. To do so, I added a specific tag in the configuration file */var/ossec/etc/ossec.conf*: Under the tag <*ossec_config*> the following <remote> xml element is responsible for enabling the server to intercept log messages coming from the firewall that is characterized by its local ip address.



```
 GNU nano 6.2                 /var/ossec/etc/ossec.conf
   <log_alert_level>3</log_alert_level>
   <email_alert_level>12</email_alert_level>
 </alerts>

 <!-- Choose between "plain", "json", or "plain,json" for the format of intern
 <logging>
   <log_format>plain</log_format>
 </logging>


<remote>
 <connection>syslog</connection>
 <port>514</port>
 <protocol>udp</protocol>
 <allowed-ips>[redacted]/24</allowed-ips>
 <local_ip>[redacted]</local_ip>
</remote>
```

Figure 9: Connected endpoints on wazuh dashboard

Configuration changes would have effect only after restarting *Wazuh-Manager*. Immediately after the manager get started we see port 514 udp[5] open and listening for incoming messages.

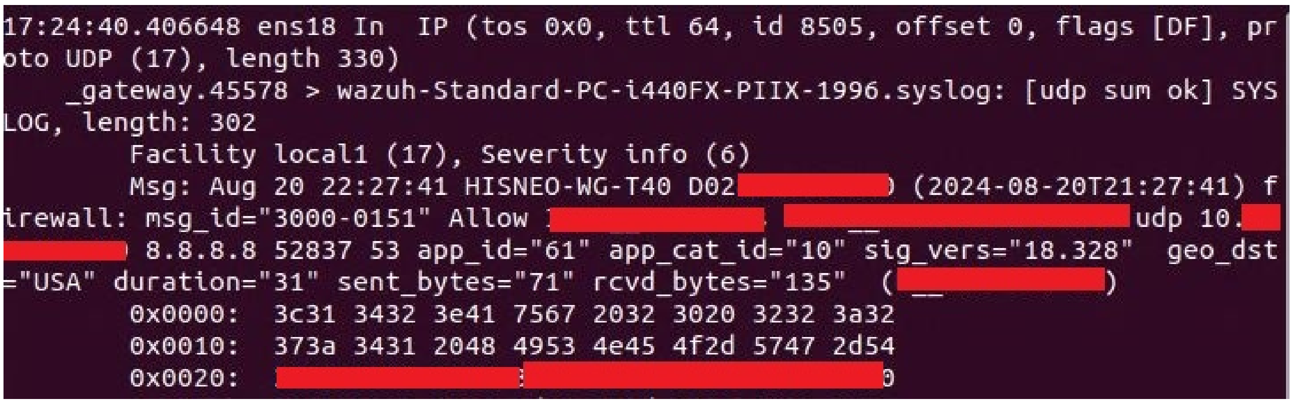
[5]The default port used for receiving syslog messages

4.1.2 Default Decoders

In Wazuh, the component responsible for parsing logs is the Logcollector. It collects log data from various sources and sends it to the Wazuh manager. Once the logs reach the manager, the Analysisd component processes and parses the logs, applying decoding and rules to generate security alerts based on patterns and conditions. The decoders are used to extract information from received log messages. Upon receiving a log message, decoders divide the information into fields to prepare them for analysis. Decoders are the way to spot interesting information in a log message, using XML syntax they specify how log data should be parsed and normalized. They include elements such as < decoders> to define a decoder, <regex> for pattern matching definition and <order> to specify the decoding sequence.

4.1.3 Firebox's Logs Format

Each log message generated by Firebox includes a string of data about the traffic passing through the firewall. We used a linux utility called tcpdump to intercept syslog messages targeting wazuh server on port 514, the result is shown in figure 10.



```
17:24:40.406648 ens18 In  IP (tos 0x0, ttl 64, id 8505, offset 0, flags [DF], pr
oto UDP (17), length 330)
    _gateway.45578 > wazuh-Standard-PC-i440FX-PIIX-1996.syslog: [udp sum ok] SYS
LOG, length: 302
        Facility local1 (17), Severity info (6)
        Msg: Aug 20 22:27:41 HISNEO-WG-T40 D02[redacted]) (2024-08-20T21:27:41) f
irewall: msg_id="3000-0151" Allow [redacted] [redacted]udp 10.[redacted]
[redacted] 8.8.8.8 52837 53 app_id="61" app_cat_id="10" sig_vers="18.328"  geo_dst
="USA" duration="31" sent_bytes="71" rcvd_bytes="135"  ([redacted])
        0x0000:  3c31 3432 3e41 7567 2032 3020 3232 3a32
        0x0010:  373a 3431 2048 4953 4e45 4f2d 5747 2d54
        0x0020:  [redacted][redacted]0
```

Figure 10: Raw log message generated by firebox

Each log message generated by the Firebox contains detailed information regarding network traffic. When reviewing these logs in the Traffic Monitor, various colors are used to visually distinguish specific data points. Below is a breakdown of a typical traffic log message and its components, using the following example log entry:

2024-08-02 17:38:43 Member2 (2024-08-02T17:38:43) Allow 192.168.228.202 10.0.1.1 webcache/tcp 42973 8080 3-Trusted 1-WCI Allowed 60 63 (Outgoing-proxy-00) proc_id="firewall" rc="100" src_ip_nat="69.164. tcp_info="offset 10 S 2982213793 win 2105" msg_id="3000-0148"

Time Stamp The log message begins with a time stamp that records when the message was created. The time stamp reflects the time zone and current time configured on the Firebox. In the example:

- **Time Stamp**: `2024-08-02 17:38:43`

FireCluster Member Information If the Firebox is part of a FireCluster, the log message indicates the member number of the FireCluster. In the example:

- **Cluster Member**: `Member2`

Disposition The disposition of the traffic is specified as either "Allow" or "Deny." For proxy-managed traffic, even if the packet is altered or stripped, it may still be marked as "Allow." In the example:

- **Disposition**: `Allow`

Source and Destination Addresses The source and destination IP addresses of the traffic are recorded. If NAT is applied, the NAT addresses appear later in the log. In the example:

- **Source Address**: `192.168.228.202`
- **Destination Address**: `10.0.1.1`

Service and Protocol The log message includes the service and protocol used to manage the traffic. If the protocol is not determined, the port number is shown. In the example:

- **Service/Protocol**: `webcache/tcp`

Source and Destination Ports The source port identifies the return traffic, while the destination port indicates the service used. In the example:

- **Source Port**: `42973`
- **Destination Port**: `8080`

Source and Destination Interfaces These are the interfaces handling the connection. In the example:

- **Source Interface**: `3-Trusted`
- **Destination Interface**: `1-WCI`

Connection Action This specifies the action applied to the connection, such as "Allowed," "Dropped," or "Stripped." In the example:

- **Connection Action**: `Allowed`

Packet Length and TTL The packet length (in bytes) and the Time To Live (TTL) value are included. In the example:

- **Packet Length**: `60`
- **TTL**: `63`

Policy Name The policy managing the traffic is identified, and a unique identifier is appended to the policy name. In the example:

- **Policy Name**: `(Outgoing-proxy-00)`

Process Information The log message specifies the process responsible for handling the traffic. In the example:

- **Process**: `proc_id="firewall"`

Return Code A return code is provided for the packet, which is used in report generation. In the example:

- **Return Code**: `rc="100"`

NAT Address If NAT is applied, the source IP address is replaced by a NAT address. In the example:

- **Source NAT Address**: `src_ip_nat="69.164.168.163"`

Packet Size Details For TCP traffic, the message includes information on the packet's offset, sequence, and window size. In the example:

- **TCP Info**: `tcp_info="offset 10 S 2982213793 win 2105"`

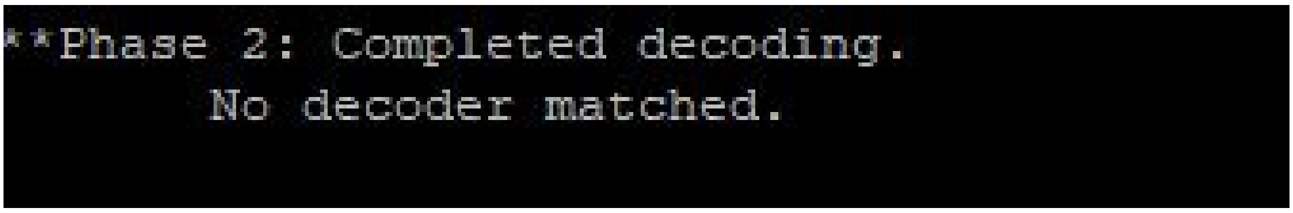
Message Identification Number The message includes a unique message identification number for tracking and cataloging. When searching for this ID in the Log Catalog, hyphens must be removed. In the example:

- **Message ID**: `msg_id="3000-0148"`

By understanding the structure of each log message, network administrators can effectively monitor traffic, troubleshoot issues, and analyze network security events.

4.1.4 Problem: Unknown Log Format

To make sure of the ability of wazuh decoders to parse firebox log messages, we ran an utility located at */var/ossec/bin/ossec_logtest*. We run the utility and we pass to it a log sample. The result shown in figure 11 indicates the absence of any decoder that is able to parse firebox logs.



```
**Phase 2: Completed decoding.
       No decoder matched.
```

Figure 11: Absence of compatible decoders

The firewall WatchGuard Firebox is formatting logs in a manner that is unknown by the default wazuh decoders. In fact, wazuh includes decoders compatible with typical syslog, the problem is in the redundant date between parenthesis included to every log message. So, it is crucial to go with custom decoders and rules. But before we can build our own decoders, we need the logs in our hands, so, we can determine their structures. Based on the format we would find - we can find ways to spot the important fields in the logs - configure custom decoders.

4.1.5 Watchguard Firebox Log Catalog

The product's vendor published a full documentation on the syntaxe and symantics of log messages that the firewall generates. The documentation is publicly available in the official site[6]

[6]The structure and meaning of events are available at: www.watchguard.com/help/docs/fireware/11/en-US/log_catalog/index.html

Firebox devices can send several types of log messages for events that occur on the Firebox. Each message includes the message type in the text of the message. The log messages types are:

- Traffic
- Alarm
- Event
- Debug (Diagnostic)
- Statistic

Traffic log messages They show how the packet filter and proxy policies were applied to the traffic passing through Firebox. When NAT is applied, this set of messages can include details about the translation.

Alarm log messages Firebox can be configured to run a specific command when a condition is satisfied, for example when it detects a denial of service attack. Upon the execution of the command it will send a log message indicating so.

Event log messages Those are logs indicating administrative activities such as:
Firebox start up and shut down
Firebox and VPN authentication
Problems with Firebox hardware components

Debug log messages Help the system administrator troubleshoot problems within Firebox, they fall into 4 levels:

- Error
- Warning
- Information
- Debug

Statistic Log Messages They include performance related information such as external interface performance, VPN bandwidth statistics, and Security Services statistics.

As we saw in figure 10, log messages come with a message identifier and message body. Each one has a description on the logcatalog. The logcatalog also specifies the variable parts in the log messages and defines their type and meaning, which will be a reference when creating custom rules and decoders.

Example of log definition in logcatalog: Taking an example from the section of logcatalog that defines management related events, it will be clear how we can use this information to parse those logs in wazuh server. A log with an identifier 3E000003 for example, is described as "a user log in attempt failed. The log message specifies the user type, user name, IP address, and the failure reason, if available." The components of logs with this identifier also are defined in the catalog, *%s%s%s from %s log in attempt was rejected*, and the meaning of each component is also included. *${user_type}* *${user_name}${auth_server}* *{ipaddr}* are respectively representing the strings in the log

format. Sometimes this type of messages can be appended with the virtual IP address of who failed the attempt and the reason of rejection, both fields are strings.p Using this information, we can map each portion of the following log with a specific name in a standardized manner:
Management user admin from 10.0.1.2 log in attempt was rejected.

4.1.6 Solution: Custom Decoders

Coupling the information from the logcatalog with wazuh capability to define custom decoders solves the problem of unknown log format. This is explained on more details in the next section.

4.2 Configuring Rules and Decoders

Decoders operate in two phases: pre-decoding and decoding. During the pre-decoding phase, general information such as a timestamp, hostname, and program name are extracted when a syslog-like header is present.

The initial extraction provides essential context for further analysis. In the subsequent decoding phase, decoders parse and interpret the remaining log data, extracting additional relevant information. They play a crucial role in parsing and interpreting log data, allowing Wazuh to understand and respond to different types of events. However, as we showed in the previous section, Firebox syslog messages are different than typical syslog messages. Hence we need to develop decoders from scratch, including pre-decoders.

4.2.1 Decoders' syntax

Events will comes from Firebox as syslog messages. It is the responsibility of the decoders to extract the information from the received messages. The decoders separate the information into named blocks to prepare them for subsequent analysis. To understand how decoders work, it will be easier through an example. Consider the following log sample:
Apr 14 19:28:21 gorilla sshd[31274]: Connection closed by 192.168.1.33

The first step should always be to run the /var/ossec/bin/wazuh-logtest utility and input the event log to test the current decoder and rule before creating our own decoder.

With the event log above, we obtain the following result:

```
  Type one log per line

Apr 14 19:28:21 gorilla sshd[31274]: Connection closed by 192.168.1.33

**Phase 1: Completed pre-decoding.
        full event: 'Apr 14 19:28:21 gorilla sshd[31274]: Connection closed by 192.168.1.33'
        timestamp: 'Apr 14 19:28:21'
        hostname: 'gorilla'
        program_name: 'sshd'

**Phase 2: Completed decoding.
        name: 'sshd'
        parent: 'sshd'
        srcip: '192.168.1.33'
```

The log sample provided to the logtest utility passed through two phases. The pre-decoding phase extracted general information such as the timestamp, the hostname and the program related to the log message because the sample is a regular syslog message.

Option	Values	Description
decoder	Name of the decoder	This attribute defines the decoder.
parent	Any decoder's name	It will reference a parent decoder and the current one will become a child decoder.
accumulate	None	It allows tracking events over multiple log messages.
program_name	Any regex, sregex or pcre2 expression.	Sets a program name as a condition for applying the decoder. The log header must have a program name matching the regula expression.
prematch	Any regex or pcre2 expression.	Sets a regular expression as a condition for applying the decode The log must match the regular expression without considering Syslog-like header.
regex	Any regex or pcre2 expression.	The decoder will use this option to find fields of interest and extract them.
order	See order table	The values that regex will extract will be stored in these groups.
fts	See fts table	First time seen.
ftscomment	Any String	Adds a comment to fts.
plugin_decoder	See below	Specifies a plugin that will do the decoding. Useful when the extraction with regex is not feasible.
use_own_name	True	Only for child decoders.
json_null_field	String	Adds the option of deciding how a null value from a JSON will be stored.
json_array_structure	String	Adds the option of deciding how an array structure from a JSON will be stored.
var	Name for the variable.	Defines variables that can be reused inside the same file.
type	See type table	It will set the type of log that the decoder is going to match.

The second phase take information from the primary phase and extracts more information. In this case the string ”Connection closed by 192.168.1.33” was subject of analysis by the second phase decoder that extracts the IP address 192.168.1.33. Under /var/ossec/rules/decoders is a list of xml files. Each file is a compilation of decoders related somehow. For example var/ossec/rules/decoders/0310-ssh_decoders.xml assemble decoders to parse logs from sshd service.

There are several decoder configuration options. Above, you can find a description of the XML labels used to configure decoders.

Always in the same file that contains sshd related decoders. If we consider the following part of the file:

```
<decoder name="sshd">
  <program_name>^sshd</program_name>
</decoder>

<decoder name="sshd-success">
  <parent>sshd</parent>
  <prematch>^Accepted</prematch>
  <regex offset="after_prematch">^ \S+ for (\S+) from (\S+) port </regex>
  <order>user, srcip</order>
  <fts>name, user, location</fts>
</decoder>
```

The decoder option serves as the root element of a decoder file in Wazuh. It encapsulates the definition of a decoder, including its name, type, and the specific attributes that dictate how it processes and extracts information from log messages. In our example, we put the definitions of two decoders under this element. sshd and sshd-success are the names of each decoder. The decoder sshd is mapped to log messages that have program name sshd. It matches the regular expression that defines the name sshd. The decoder sshd-success get triggered only after sshd decoder. This behavior is specified using the <parent >that. The element prematch matches the Accept pattern at the beginning of the log message that was matched by the decoder sshd. This decoder will match the following log message.

```
sshd[8813]: Accepted password for root from 192.168.10.1 port 1066 ssh2
```

After it is matched, the decoder will extract two fields from this log message using *regex* and *order* elements (see the above table). More details are available at the documentation site[7].

4.2.2 Pattern Matching

As we saw in the previous section, wazuh decoders are hghly based on pattern matching. Regular expressions or regex are sequences of characters that define a pattern. Wazuh supports three variants of regular expressions:

- regex
- sregex
- perl compatible regex

The type used in this project is perl compatible regex also known as PCRE2. A comprehensive documentation of its syntax can be found at pcre.org[8].

4.2.3 Wazuh Custom Rules

Wazuh rules are a set of conditions written in XML format that define how log data should be interpreted. These rules are used by the Wazuh manager to detect specific patterns or behaviors within log messages and generate alerts or responses accordingly. They play a crucial role in threat detection, as they allow the system to identify potential security incidents based on predefined criteria. The following table lists commonly used xml elements used to configure custom rules.

Each rule typically consists of elements such as <rule>, <description>, <group>and <field>, among others. In which, the <description>element provides a clear explanation of the rule's purpose and functionality. Within the <group>element, rules are categorized based on their relevance or priority and the <field>element specifies the log data fields to be evaluated for matching conditions. Overall, the rules syntax in Wazuh facilitates the precise definition of criteria for detecting specific patterns or behaviors in log messages, contributing to effective threat detection and response mechanisms. Additionally, each rule is assigned an ID and a level. The rule's level determines the severity of the alert triggered when the rule conditions are met. Levels range from 0 (ignored) to 16 (severe attack), with each level indicating a different level of security relevance.

[7] https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/decoders.html
[8] https://perldoc.perl.org/perlre

Option	Values	Description
rule	See this table below.	Declares a new rule and its defining options.
match	Any regular expression.	Attempts to find a match in the log using sregex by default, deciding if the rule should be triggered.
regex	Any regular expression.	Does the same as match, but with regex as default.
decoded_as	Any decoder's name.	Matches with logs that have been decoded by a specific decoder.
category	Any type.	Matches logs with the corresponding decoder's type.
field	Name and any regular expression.	Compares a field extracted by the decoder in order with a regular expression.
srcip	Any IP address.	Compares the IP address with the IP decoded as srcip.
dstip	Any IP address.	Compares the IP address with the IP decoded as dstip.
srcport	Any regular expression.	Compares a regular expression representing a port with a value decoded as srcport.
dstport	Any regular expression.	Compares a regular expression representing a port with a value decoded as dstport.
data	Any regular expression.	Compares a regular expression representing data with a value decoded as data.
extra_data	Any regular expression.	Compares a regular expression representing extra data with a value decoded as extra_data.
user	Any regular expression.	Compares a regular expression representing a user with a value decoded as user.
system_name	Any regular expression.	Compares a regular expression representing a system name with a value decoded as system_name.
program_name	Any regular expression.	Compares a regular expression representing a program name with a value pre-decoded as program_name.
protocol	Any regular expression.	Compares a regular expression representing a protocol with a value decoded as protocol.
hostname	Any regular expression.	Compares a regular expression representing a hostname with a value pre-decoded as hostname.
time	Any time range. e.g. (hh:mm-hh:mm)	Checks if the event was generated during that time range.
weekday	monday - sunday, weekdays, weekends	Checks whether the event was generated during certain weekdays.
id	Any regular expression.	Compares a regular expression representing an ID with a value decoded as id
url	Any regular expression.	Compares a regular expression representing a URL with value decoded as url
location	Any regular expression.	Compares a regular expression representing a location with a value pre-decoded as location.
action	Any String or regular expression.	Compares a string or regular expression representing an actionwith a value decoded as action.
status	Any regular expression.	Compares a regular expression representing a status with a value decoded as status.
srcgeoip	Any regular expression.	Compares a regular expression representing a GeoIP source with a value decoded as srcgeoip.

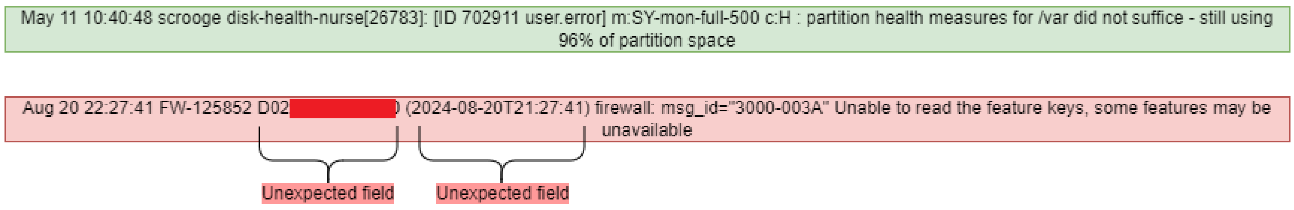
Table 1: Examples of rules related xml elements

Syntax and classification of rules When it comes to rule creation, there is an important concept that must be known. When an alert is raised, the user must assess the severity of the event behind that alert. This can be achieved by assigning alerts a number between and 0 and 16. The number is included in rule element which is the label that starts the block defining a rule. The <rule>element can have several options but required to have an id option which must be unique to each rule. Identifiers for user created rules must start from 100000 and end at 999999 to avoid conflict with default rules. A full list of xml elements supported to define rules is available at the official documentation[9].

4.2.4 WatchGuard Firebox

We get inspired by the available decoders and rules created by the community of wazuh to tolerate our server with Firebox syslog messages. We give in this document only examples of firebox-specific rules and decoders.

Decoders In the section about the problem with firebox's syslog messages (section 2.1.4), we mentioned that what makes the message unable to be parsed as usual syslog messages is a redundant timestamp enclosed between parenthesis. There is also an unexpected serial number encoded in log messages if we compare them with usual syslog messages. Figure 12 illustrates the difference. To



```
May 11 10:40:48 scrooge disk-health-nurse[26783]: [ID 702911 user.error] m:SY-mon-full-500 c:H : partition health measures for /var did not suffice - still using 96% of partition space
```

```
Aug 20 22:27:41 FW-125852 D02[redacted] (2024-08-20T21:27:41) firewall: msg_id="3000-003A" Unable to read the feature keys, some features may be unavailable
```

Figure 12: The difference between typical syslog messages and firebox's messages

counter this problem, our custom decoders must be based on a parent decoder that takes into consideration the specific format of firebox log messages. Let's take the example of predecoder of log messages originated by the firewall process of firebox. The firewall's pre-decoder represented bellow will trigger only if the serial number and the redundant date match in the log message.

```
<decoder name="firebox_firewall_predecoder">
  <prematch type="pcre2">(D02\w+)\s(\(\S+\))\sfirewall: </prematch>
</decoder>
```

An example of a child decoder that extracts the message identifier of a log message which is an important piece of information to map the message with the log catalog is presented bellow.

```
<decoder name="firebox_firewall_decoder">
  <parent>watchguard-firebox-firewall</parent>
  <regex type="pcre2" offset="after_parent">msg_id=\\?"([0-9a-fA-F]{4}-[0-9a-fA-F]{4})\\?"</re
  <order>id</order>
</decoder>
```

This decoder would be triggered only if its parent does. When the analysisd component acknowledges the presence of firebox firewall log message, it triggers the above decoder. The decoder will take the string that comes after *firewall:* , and match the expression inside regex element. This expression

[9] https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/rules.html

starts by the string *msg_id=* followed by the format of identifiers as defined in the logcatalog of firebox. The identifier is 8 digits hexadecimal number separated to two equal parts by a dash will be labeled as id. The letters in the number are either capital or small letter.

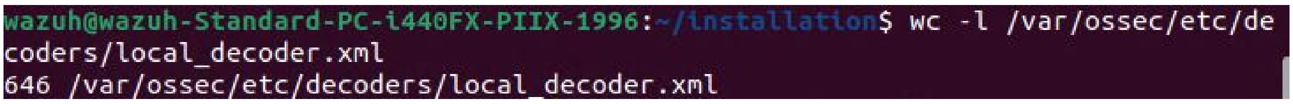
An other example related with the next section is a predecoder of authentication related processes.

```
<decoder name="watchguard-firebox-auth">
  <prematch type="pcre2">(D02\w+)\s(\(\S+\))\s(admd|wgagent|sessiond|sslvpn|wgcgi)\[\w+\]: </p
</decoder>
```

admd, wgagent, sessiond and wgcgi are management processes supporting user authentication. sslvpn is the name of the process responsible of vpn utility, it generates log messages whenever a user failed to login to the intranet. The following decoder is based on the above predecoder. It extracts the message identifier of log messages related to authentication-enabled processes.

```
<decoder name="watchguard-firebox-auth-info">
  <parent>watchguard-firebox-auth</parent>
  <regex type="pcre2" offset="after_parent">msg_id="([0-9a-fA-F]{4}-[0-9a-fA-F]{4})"</regex>
  <order>id</order>
</decoder>
```

Custom decoders are placed under the directory */var/ossec/etc/decoders* as shown in figure 13.



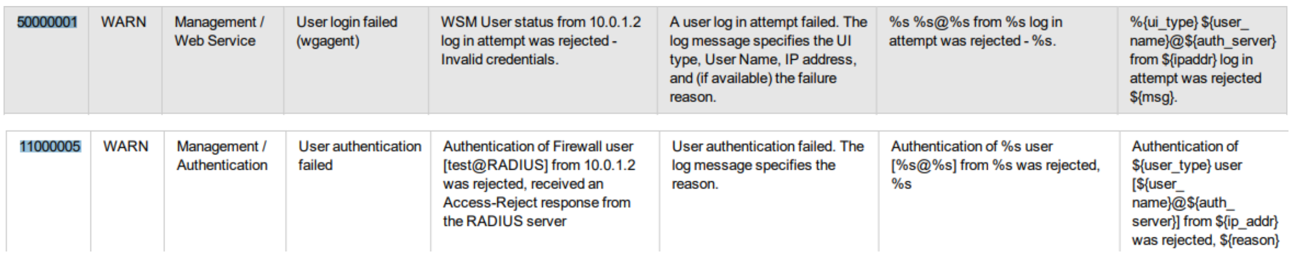
```
wazuh@wazuh-Standard-PC-i440FX-PIIX-1996:~/installation$ wc -l /var/ossec/etc/de
coders/local_decoder.xml
646 /var/ossec/etc/decoders/local_decoder.xml
```

Figure 13: WatchGuard Firebox custom decoders xml file

Rules Apart from assigning to each alert a level that reflects the severity of a security event behind, Wazuh provide more ways to help the security analysts assessing such event. <group>xml element is a powerful way to tag alerts. Tagging helps analysts to conduct investigations, perform statistics among other thing by enriching alerts with information that isn't present in the original log message. Wazuh also can be aligned with MITRE ATTACK framework[10] by using <mitre>element. The following uses the decoder presented as an example in the previous section.

```
  <rule id="791201" level="5">
    <decoded_as>watchguard-firebox-auth</decoded_as>
    <id type="pcre2">1100-0005|5000-0001</id>
    <action>rejected</action>
    <description>Watchguard: $(account_type) $(srcuser) login $(action) from $(srcip) - $(reas
    <mitre>
      <id>T1133</id>
    </mitre>
    <group>wg_userloginfailed,authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,gpg13_7.8,gd
  </rule>
```

Upon extraction of the message identifier of authentication-related log messages. This rule matches the id numbers related with a failed login attempt. Figure 14 shows the messages having those

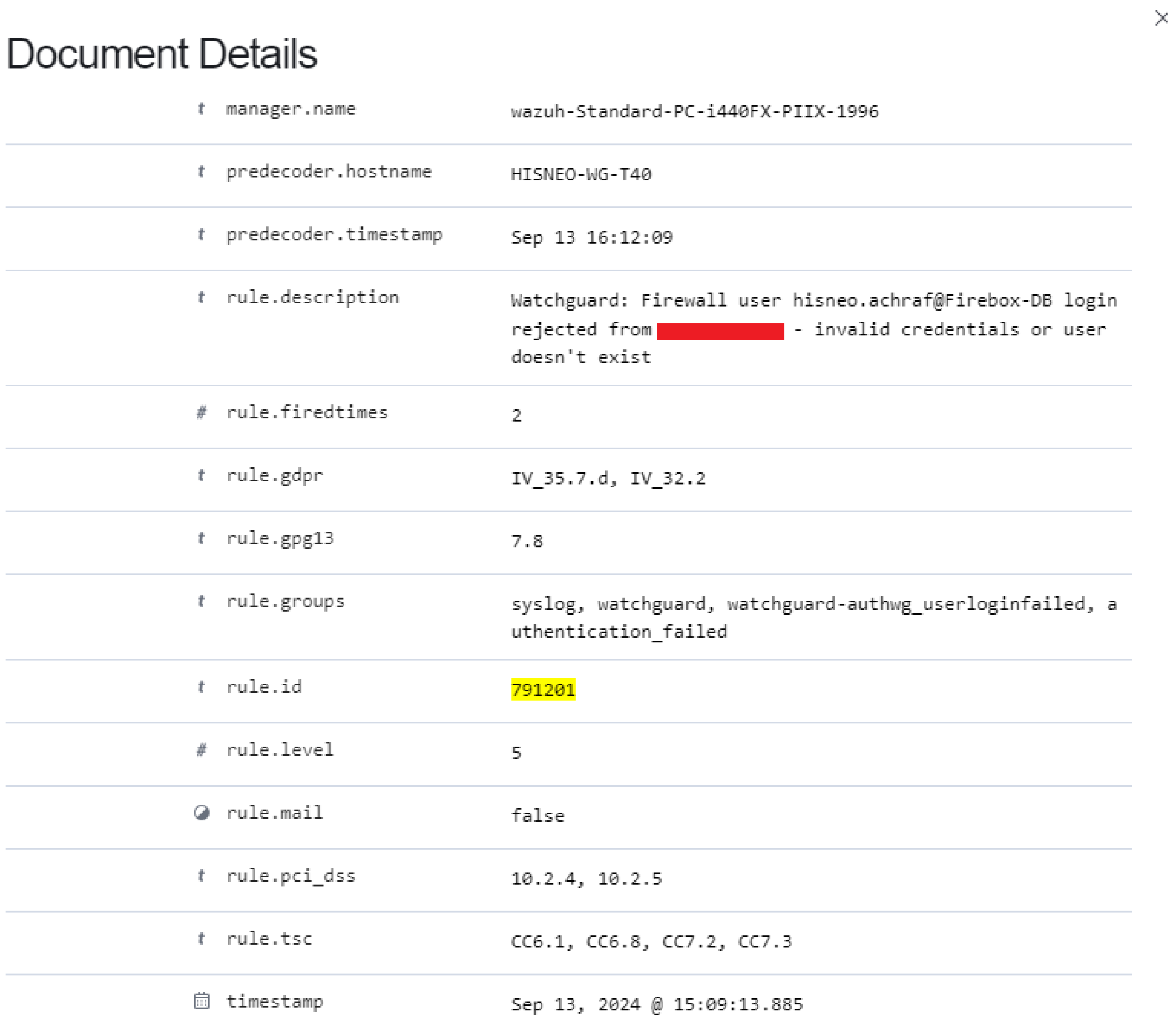


50000001	WARN	Management / Web Service	User login failed (wgagent)	WSM User status from 10.0.1.2 log in attempt was rejected - Invalid credentials.	A user log in attempt failed. The log message specifies the UI type, User Name, IP address, and (if available) the failure reason.	%s %s@%s from %s log in attempt was rejected - %s.	%{ui_type} ${user_name}@${auth_server} from ${ipaddr} log in attempt was rejected ${msg}.
11000005	WARN	Management / Authentication	User authentication failed	Authentication of Firewall user [test@RADIUS] from 10.0.1.2 was rejected, received an Access-Reject response from the RADIUS server	User authentication failed. The log message specifies the reason.	Authentication of %s user [%s@%s] from %s was rejected, %s	Authentication of ${user_type} user [${user_name}@${auth_server}] from ${ip_addr} was rejected, ${reason}

Figure 14: Failed login attempts log messages in logcatalog

identifiers in the logcatalog of watchguard firebox, the reference of rule creation. When matched we would see in the dashboard of wazuh an alert with level 5. The alert would include the description that specifies the origin of the attempt and tags that helps the security analyst to correlate the alert with mitre attack framwork, category of attack and regulatory standards (see figure 15).
Custom rules need to be placed under the directory */var/ossec/etc/rules* as figure 16 shows.

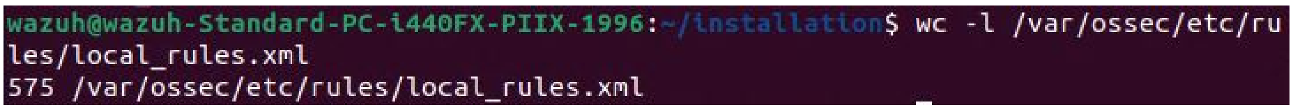
[10] https://attack.mitre.org/



Document Details

t manager.name	wazuh-Standard-PC-i440FX-PIIX-1996
t predecoder.hostname	HISNEO-WG-T40
t predecoder.timestamp	Sep 13 16:12:09
t rule.description	Watchguard: Firewall user hisneo.achraf@Firebox-DB login rejected from [redacted] - invalid credentials or user doesn't exist
# rule.firedtimes	2
t rule.gdpr	IV_35.7.d, IV_32.2
t rule.gpg13	7.8
t rule.groups	syslog, watchguard, watchguard-authwg_userloginfailed, authentication_failed
t rule.id	791201
# rule.level	5
rule.mail	false
t rule.pci_dss	10.2.4, 10.2.5
t rule.tsc	CC6.1, CC6.8, CC7.2, CC7.3
timestamp	Sep 13, 2024 @ 15:09:13.885

Figure 15: Alert with tags



```
wazuh@wazuh-Standard-PC-i440FX-PIIX-1996:~/installation$ wc -l /var/ossec/etc/ru
les/local_rules.xml
575 /var/ossec/etc/rules/local_rules.xml
```

Figure 16: XML file of firebox's rules

5 The power of wazuh

5.1 Introduction

This project ends by showcasing some of wazuh's capabilities through a concrete use case.

Consider the following scenario. A user of a monitored system was successfully manipulated to download a malicious file attached in a spearfishing email. The malicious file if opened, it will establish a channel with a command and control server, compromising the user's whole system. Using wazuh, the attack can be stopped at an early phase, upon the download of the file.

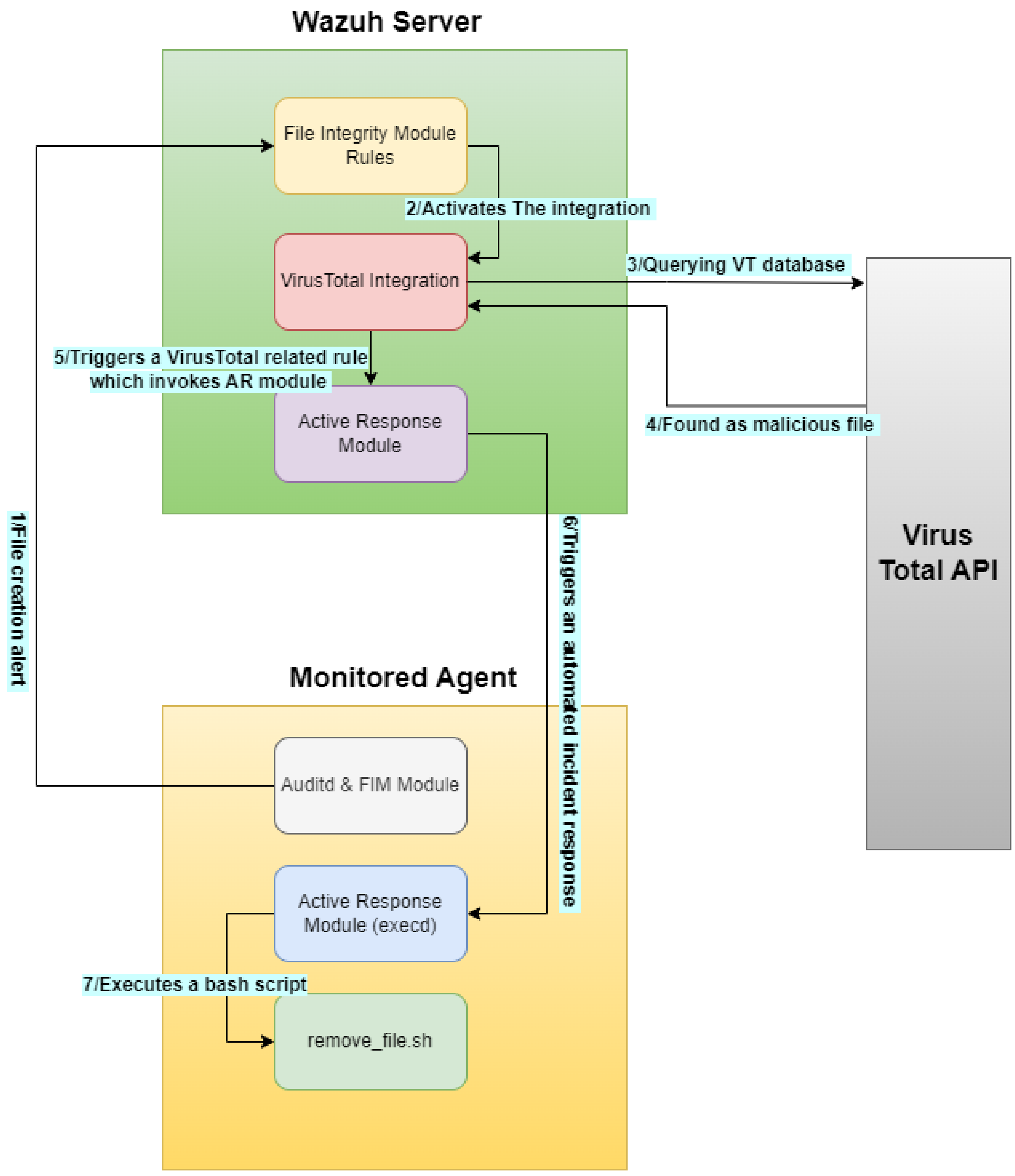

Wazuh Server
File Integrity Module Rules
2/Activates The integration
VirusTotal Integration
3/Querying VT database
4/Found as malicious file
5/Triggers a VirusTotal related rule which invokes AR module
Active Response Module
Virus Total API
1/File creation alert
6/Triggers an automated incident response
Monitored Agent
Auditd & FIM Module
Active Response Module (execd)
7/Executes a bash script
remove_file.sh

Figure 17: Schema of the involved components

To do it, three capabilities of wazuh are coupled (see section 17).

- The FIM module
  Stands for file integrity monitoring module is responsible for monitoring changes on a specific file or folder. Any addition, modification or deletion of file in a specified path would trigger an

alert.

- The Integrator
  A module responsible for communicating data with third parties (service providers or applications) would be triggered upon the generation of FIM's alert. It will detect the malicious nature of the file.

- Active response module
  Makes it possible to automate the incident response process by automatically deleting the malicious file in the agent.

5.2 File Integrity Monitoring Module

The agent first need to be configured to collect file-related logs. This is what FIM module is used for. The following is the configuration we added to the file /var/ossec/etc/ossec.conf inside the element <ossec_config>.

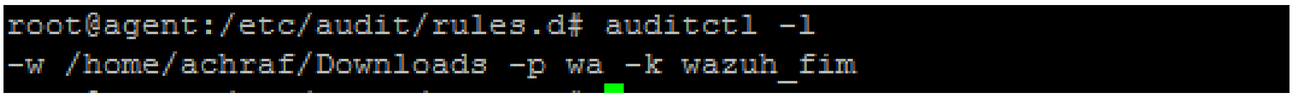
```
<!-- Added by user -->

<syscheck>
<directories check_all="yes" whodata="yes">/home/achraf/Downloads</directories>

</syscheck>

<!-- End ----------->
```

There are many options to set FIM configuration. A full list of such options is found in the documentation website[11].

The section specifies the directory where files downloaded from the internet are stored, this is the string inside <directories>xml element. This element can include many other options. One option with great importance when investigating security incidents is whodata, it is set to yes. whodata would enrich the logs generated by the FIM module with information related to file change such as the process, the parent process and the user who commited the change among other information. This option can only work if the agent has auditd daemon installed, up and running. In an ubuntu system this is done by executing the following in a shell:

```
apt-get install auditd
```

Restart the wazuh agent and the deamon would be integrated with FIM module. The daemon would watch for changes in directories in real-time by hooking to the system calls that handle the filesystem. auditd can be configured for such monitoring using a specific commands (see figure 18).



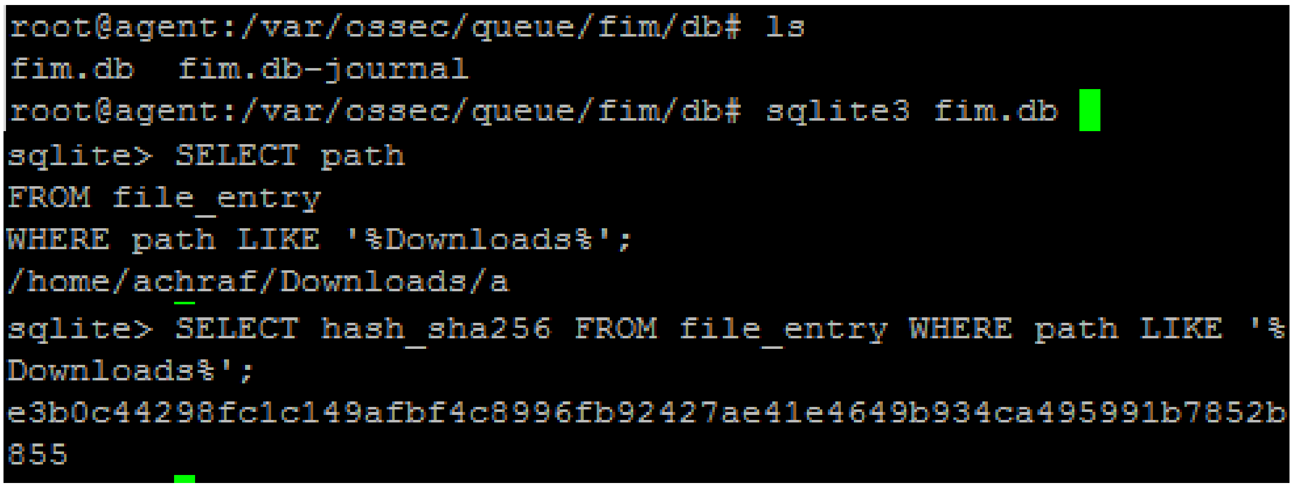
```
root@agent:/etc/audit/rules.d# auditctl -l
-w /home/achraf/Downloads -p wa -k wazuh_fim
```

Figure 18: auditd-specific command

The command in the figure was added by the FIM module after we included the syscheck section in the configuration file, and restarting wazuh agent. The rule will watch for changes happening in

[11] https://documentation.wazuh.com/current/user-manual/reference/ossec-conf/syscheck.html

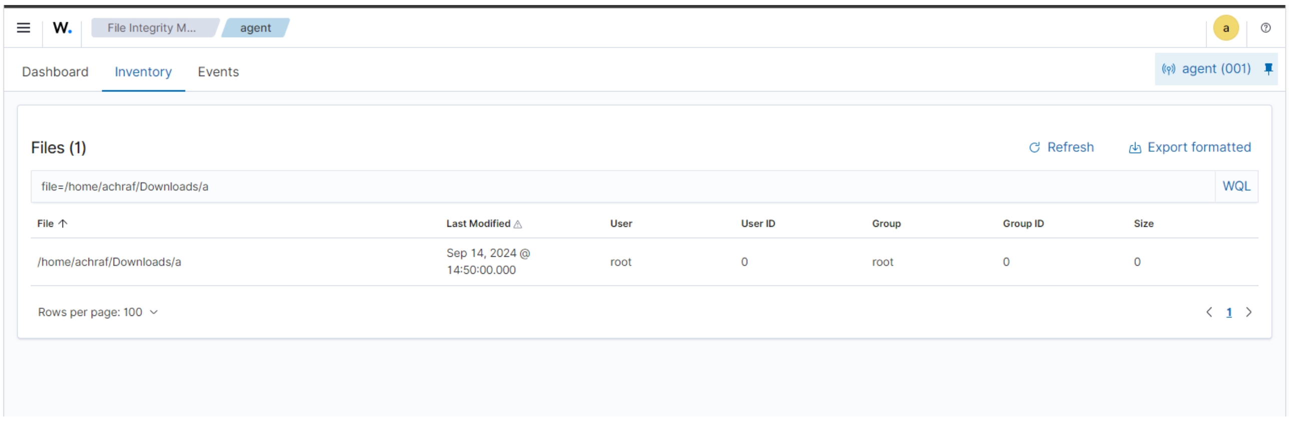
the directory /home/achraf/Downloads in real time. Whenever a change is detected, it generates the corresponded log tagged with the string wazuh_fim. It stores logs in the file /var/log/audit/audit.log. The agent's FIM module would trigger when auditd generates logs pertaining to its own command.



```
root@agent:/var/ossec/queue/fim/db# ls
fim.db  fim.db-journal
root@agent:/var/ossec/queue/fim/db# sqlite3 fim.db
sqlite> SELECT path
FROM file_entry
WHERE path LIKE '%Downloads%';
/home/achraf/Downloads/a
sqlite> SELECT hash_sha256 FROM file_entry WHERE path LIKE '%
Downloads%';
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b
855
```

Figure 19: SQLITE database stored in the agent

FIM module uses two databases in-order to detect file-integrity related events. A local sqlite database located in the agent that stores the names and the hashes of each file under a monitored directory among other Information. And one located in the server that keeps an inventory of agents' files. Figure 19 shows the content of the local database upon the creation of a new file under /home/achraf/Downloads. Figure 20 is wazuh dashboard displaying the information on the same file that is stored in the server's database.



Files (1)

file=/home/achraf/Downloads/a

File	Last Modified	User	User ID	Group	Group ID	Size
/home/achraf/Downloads/a	Sep 14, 2024 @ 14:50:00.000	root	0	root	0	0

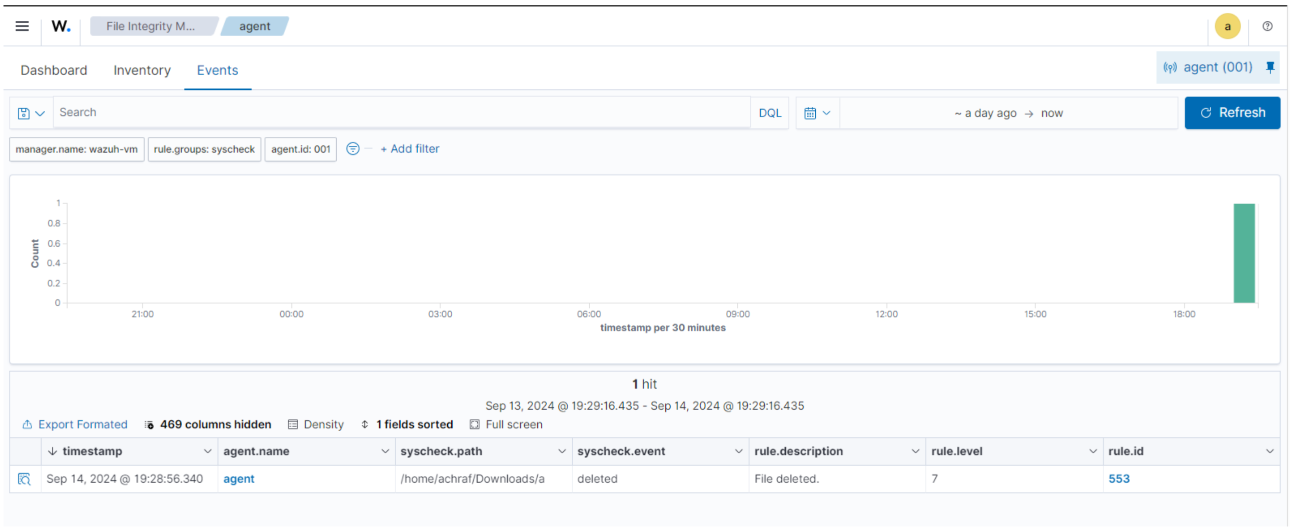
Rows per page: 100

Figure 20: The content of server-based FIM database

Syscheck alerts are generated when an entry in the database get changed, deleted or added.

Wazuh includes out-of-the-box rules that trigger alerts on the creation, modification, or deletion of monitored files. The figure below shows alert upon the deletion of the file /home/achraf/Downloads/a created in the previous step.

One can use custom Wazuh FIM rules to monitor changes to files and directories based on specific criteria such as filename, permissions, and content. For example, you can create a custom rule to



manager.name: wazuh-vm | rule.groups: syscheck | agent.id: 001

1 hit

Sep 13, 2024 @ 19:29:16.435 - Sep 14, 2024 @ 19:29:16.435

469 columns hidden · 1 fields sorted

timestamp	agent.name	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
Sep 14, 2024 @ 19:28:56.340	agent	/home/achraf/Downloads/a	deleted	File deleted.	7	553

Figure 21: Alert 553 - File deleted

FIM field	Alert field	Field description
file	path	File path in the current event
size	size_after	File size in the current event
hard_links	hard_links	List of hard links of the file
mode	mode	FIM event mode
perm	perm_after,win_perm_after	File permissions
uid	uid_after	User ID of the owner of the file

Table 2: Some alerts' field correlations with FIM's keywords

detect changes to a critical system file or configuration file. Whenever a user or process modifies the file, Wazuh triggers a specific alert indicating the change and the details of the modification.

Creating custom FIM rules can extend the out-of-the-box detection capability of the Wazuh FIM module. This makes it easier to identify and respond to security incidents such as data breaches, insider threats, and other cyberattacks that involve file manipulation or modification. The most important thing to know when creating rules related to the FIM module is the semantic of fields decoded by the same module.

Table 2 shows examples of how to use the fields decoded from FIM events in custom rules. It explains what the decoded FIM events fields represent in the Wazuh alert fields.

5.3 Integrating VirusTotal

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.

VirusTotal is a free service with numerous useful features. We highlight the following ones relevant to our purpose:

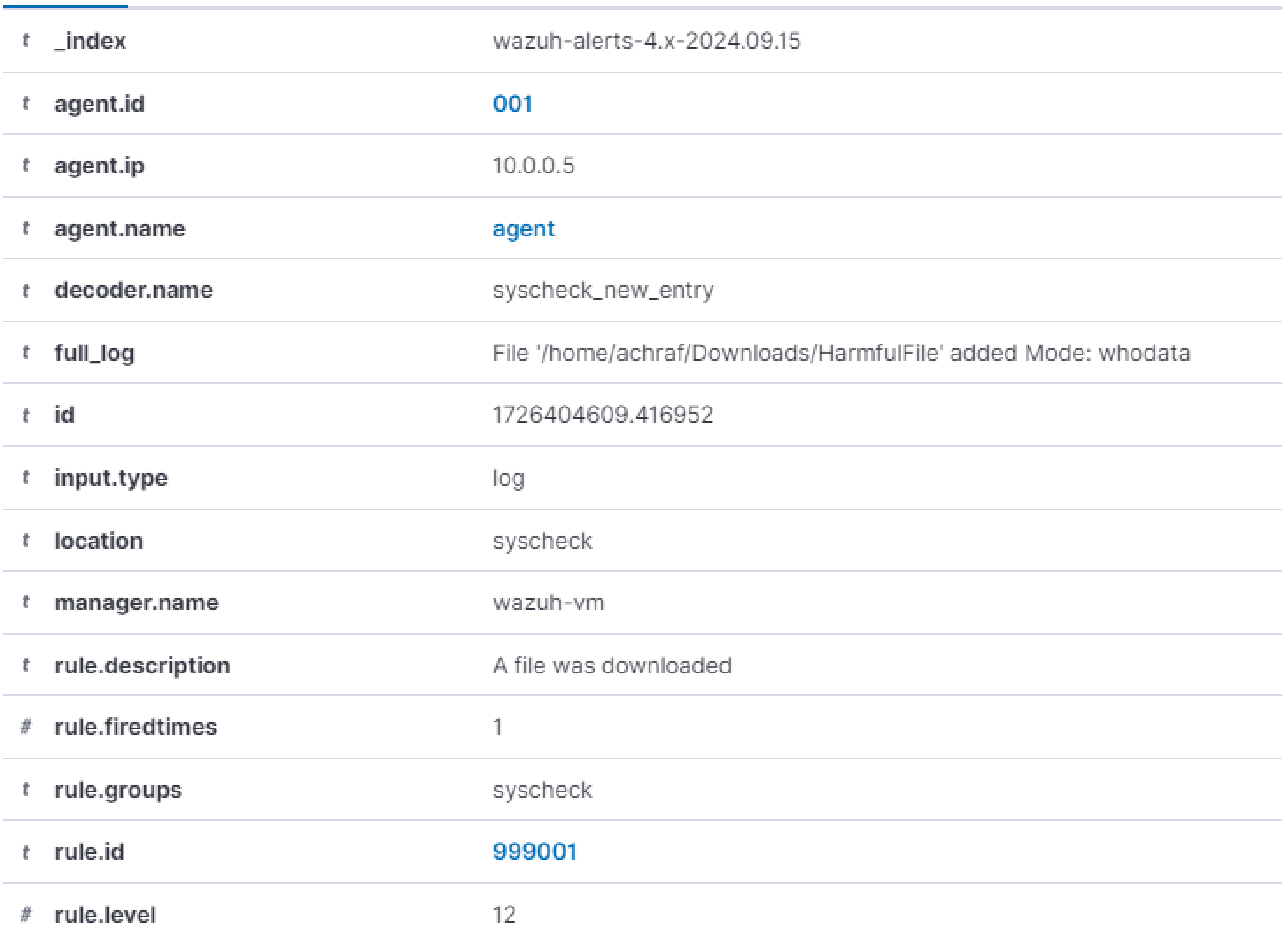
VirusTotal stores all the analyses it performs, allowing users to search for file hashes. By sending the hash to the VirusTotal engine, you can know if VirusTotal has already scanned that specific file, and you can analyze its report.

VirusTotal also provides an API that allows access to the information generated by VirusTotal without needing to utilize the HTML website interface. This API is subject to its Terms of Service, which we briefly discuss in the following section.

The Wazuh Integrator module allows Wazuh to connect to external APIs and alerting tools such as Slack, PagerDuty, VirusTotal, Shuffle, and Maltiverse. You can also configure the Integrator module to connect to other software. These integrations empower security administrators to enhance orchestration, automate responses, and fortify their defenses against cyber threats. Integrations get triggered upon the satisfaction of a rule. The rule created in this context is a child of two built-in FIM module rules. Rule 550 which alerts on the modification of a monitored file, and rule 554 that triggers whenever a file is created under a monitored directory would be the base of the integration-related rule. The following is the rule that will trigger the integration. This custom rule was reserved an xml file in the directory /var/ossec/etc/rules in the server.

```
<group name="syscheck">
<rule id="999001" level=12>
<if_sid>550,554</if_sid>
<field name="file">/home/achraf/Downloads/\w+</field>
<description>A file was downloaded</description>
</group>
```

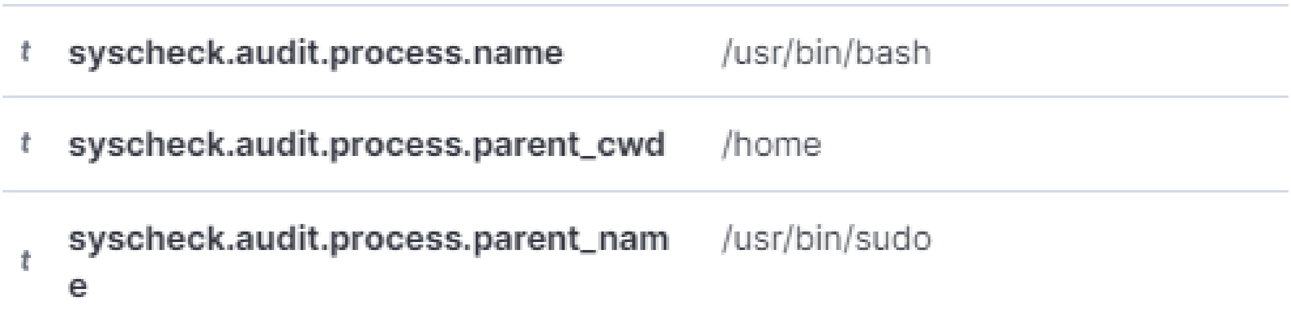
It matches only files under Downloads directory. To make sure that this configuration is the one that we want, after I restarted wazuh-manager, I created an empty file in Downloads folder and it indeed gave a positive result as in figure 22.



t	_index	wazuh-alerts-4.x-2024.09.15
t	agent.id	001
t	agent.ip	10.0.0.5
t	agent.name	agent
t	decoder.name	syscheck_new_entry
t	full_log	File '/home/achraf/Downloads/HarmfulFile' added Mode: whodata
t	id	1726404609.416952
t	input.type	log
t	location	syscheck
t	manager.name	wazuh-vm
t	rule.description	A file was downloaded
#	rule.firedtimes	1
t	rule.groups	syscheck
t	rule.id	999001
#	rule.level	12

Figure 22: The generated test alert for file creation

The alert contains also whodata information configured earlier (see figure 23).



t	syscheck.audit.process.name	/usr/bin/bash
t	syscheck.audit.process.parent_cwd	/home
t	syscheck.audit.process.parent_name	/usr/bin/sudo

Figure 23: Whodata attributes

Integrations' configuration need to be put in the xml file /var/ossec/etc/ossec.conf of the the server. The following configuration, dictates integrator module to run the built-in integration named virustotal upon an alert of id 999001.

```
<integration>
<name>virustotal</name>
<api_key>OUR_API_KEY</api_key>
<alert_format>json</alert_format>
<rule_id>999001</rule_id>
</integration>
```

The integration would query the database of virustotal using the provided api key. Along with the api key, the hash of the file included in the corresponded field in the alert 999001 would be sent in a HTTP GET request. VirusTotal will respond with an http response that get parsed by the integration which generates an alert. Testing the integration requires adding a file to the monitored folder. The following, shows the virustotal alert after creating a test-file named HarmfulFile10.

```
** Alert 1726406084.421980: - virustotal,
2024 Sep 15 13:14:44 (agent) 10.0.0.5->virustotal
Rule: 87104 (level 3) -> 'VirusTotal: Alert - /home/achraf/Downloads/HarmfulFile10 -
 No positives found'
{"virustotal": {"found": 1, "malicious": 0, "source": {"alert_id": "1726406082.421102
", "file": "/home/achraf/Downloads/HarmfulFile10", "md5": "0754ac3ddc26ee4ccfde05f81a2
eafac", "sha1": "0696d7a698116f51f228c04fecaa8c65400da057"}, "sha1": "0696d7a698116f51
f228c04fecaa8c65400da057", "scan_date": "2024-06-04 12:57:16", "positives":0, "total":
 64, "permalink": "https://www.virustotal.com/gui/file/d18c26e029b88f66c159ed502abb2a8
 6b3805eeea138098eba4ed5a763787686/detection/f-d18c26e029b88f66c159ed502abb2a86b3805ee
 ea138098eba4ed5a763787686-1717505836"},"integration": "virustotal"}
virustotal.found: 1
virustotal.malicious: 0
virustotal.source.alert_id: 1726406082.421102
virustotal.source.file: /home/achraf/Downloads/HarmfulFile10
virustotal.source.md5: 0754ac3ddc26ee4ccfde05f81a2eafac
virustotal.source.sha1: 0696d7a698116f51f228c04fecaa8c65400da057
virustotal.sha1: 0696d7a698116f51f228c04fecaa8c65400da057
virustotal.scan_date: 2024-06-04 12:57:16
virustotal.positives: 0
virustotal.total: 64
virustotal.permalink: https://www.virustotal.com/gui/file
/d18c26e029b88f66c159ed502abb2a86b3805eeea138098eba4ed
5a763787686/detection/f-d18c26e029b88f66c159ed502abb2a86b
3805eeea138098eba4ed5a763787686-1717505836
integration: virustotal
```

This alert of id 87104 is generated when VirusTotal finds no positive artifact that identify the file as malicious. For malicious files, the alert would match the rule of id 87105.

5.4 Active Response Module

To set an automated incident response when such malcious file get downloaded in the monitored folder, we passed through two steps:

1. Setting the agent
   The agent need to store a bash script that do the following:
   - Read STDIN to get the JSON message. The json message is the virusTotal alert that triggered the active response.
   - Parse the JSON message. In linux environment we used a linux utility named jq to access json elements in the message. The utility extracts the name of the malicious file from the alert. This information is present in the json element named as .parameters.alert.datavirustotal.source.file.
   - Delete that file.
2. Setting the server
   In the server side, the xml file /var/ossec/etc/ossec.conf was appended with the following:
   - A command section that names a specific action - the execution of the script located in the agent.
   - An active response block that indicates the conditions when and where the command would trigger.

The following is the active response script. It needs to be stored in /var/ossec/active_response/bin/, and is named as remove_file.

```
read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
LOG="/var/ossec/logs/active-responses.log"
rm -f $FILENAME
if [ $? -eq 0 ]; then
 echo "‘date ’+%Y/%m/%d %H:%M:%S’‘ $0: $INPUT_JSON Successfully removed threat" >> ${LOG_FILE}
else
 echo "‘date ’+%Y/%m/%d %H:%M:%S’‘ $0: $INPUT_JSON Error removing threat" >> ${LOG_FILE}
fi

exit 0;
```

The <command>block sets the script to run in response to a trigger.

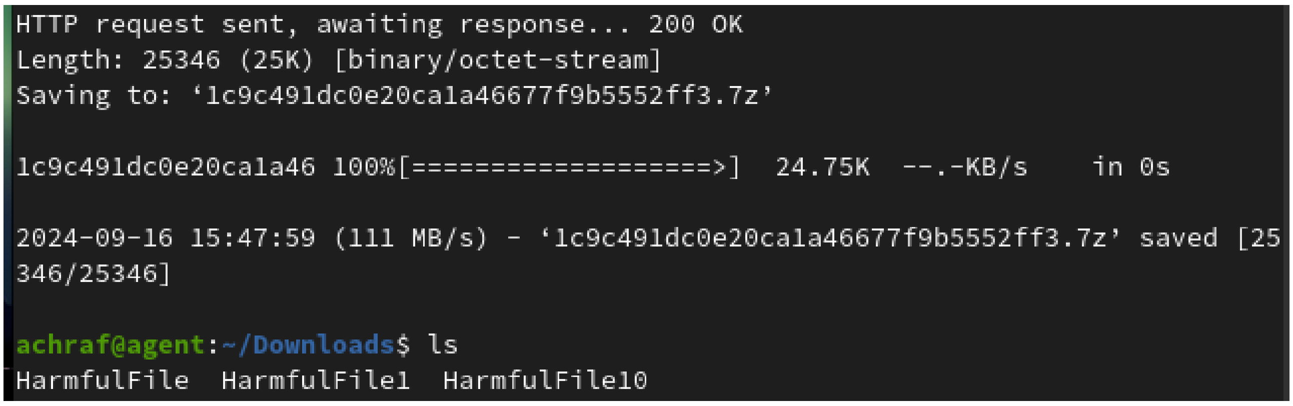
```
 <command>
    <name>file_deletion</name>
    <executable>remove_file</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>
```

The following <active-response>block defines when and where a command executes. In our case, the command executes in whatever agent that generates the virustotal alert. This can be specified by setting the location option to local. To specify the condition of execution to be an alert of specific rule id, we use the rules_id option and we set it to 87105.

```
  <active-response>
    <command>file_deletion</command>
    <location>local</location>
    <rules_id>87105</rules_id>
  </active-response>
```

5.5 Attack Emulation

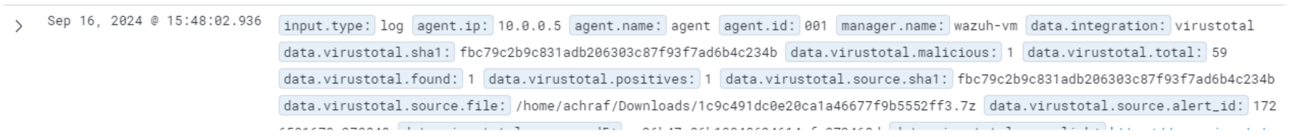
To test the validity of the use case configuration, using wget we downloaded a sample of a malware (see figure 24). The malware is a trojan widely involved in activities related to a threat actor named Goznym[12].



```
HTTP request sent, awaiting response... 200 OK
Length: 25346 (25K) [binary/octet-stream]
Saving to: ‘1c9c491dc0e20ca1a46677f9b5552ff3.7z’

1c9c491dc0e20ca1a46 100%[===================>]  24.75K  --.-KB/s    in 0s

2024-09-16 15:47:59 (111 MB/s) - ‘1c9c491dc0e20ca1a46677f9b5552ff3.7z’ saved [25
346/25346]

achraf@agent:~/Downloads$ ls
HarmfulFile  HarmfulFile1  HarmfulFile10
```

Figure 24: Malware sample download

Upon download the malware almost instantly get removed. We investigated the alerts in wazuh dashboard, and we found an alert of id 87105 as in figure 25. The deletion of the file also triggered an

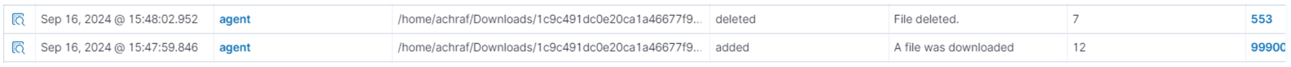


```
>  Sep 16, 2024 @ 15:48:02.936  input.type: log agent.ip: 10.0.0.5 agent.name: agent agent.id: 001 manager.name: wazuh-vm data.integration: virustotal
                                data.virustotal.sha1: fbc79c2b9c831adb206303c87f93f7ad6b4c234b data.virustotal.malicious: 1 data.virustotal.total: 59
                                data.virustotal.found: 1 data.virustotal.positives: 1 data.virustotal.source.sha1: fbc79c2b9c831adb206303c87f93f7ad6b4c234b
                                data.virustotal.source.file: /home/achraf/Downloads/1c9c491dc0e20ca1a46677f9b5552ff3.7z data.virustotal.source.alert_id: 172
```

Figure 25: Malicious file alert generated by virusTotal integration

alert shows in figure 26.

[12]https://www.europol.europa.eu/media-press/newsroom/news/goznym-malware-cybercriminal-network-dismantled-in-international-operation



🔍	Sep 16, 2024 @ 15:48:02.952	agent	/home/achraf/Downloads/1c9c491dc0e20ca1a46677f9...	deleted	File deleted.	7	553
🔍	Sep 16, 2024 @ 15:47:59.846	agent	/home/achraf/Downloads/1c9c491dc0e20ca1a46677f9...	added	A file was downloaded	12	99900

Figure 26: File deletion alert

6 Conclusion

Wazuh is a flexible XDR and SIEM tool. Flexibility in wazuh is in term of data collection and use case implementation. It is able to be integrated for any syslog-enabled system. Also, the modules it provides make it possible to realize any useful use-case for a security analyst.