Creation of SOC

Master CRYPTIS
Semestre 1

ASSR Project

Creation of Security Operation Center

Antoine MORIN
Affoua TOKOU
Bechir HEDHLI
Claudio ANTONIO
Fatima Ezzahrae ELFARA
Nhat BUI

29 janvier 2023

Faculté des Sciences & Techniques
Université de Limoges

Année universitaire 2022-2023

Table des matières

1 Architecture and components

1.1 Virtualized environment with Microstack

A virtual SOC is a secure web-based tool that allows you to easily monitor the security of your systems in real-time. This centralized command and control center enables together control of security operations, a better view into the security posture of your organization, and a one-stop- shop for all your security monitoring and incident response needs. By using a virtual SOC, administrators can prioritize security events by focusing on the incidents that have the most impact to your system.

To build our virtual environnement we choise to use Microstack which is an OpenStack in a snap which means that all OpenStack services and supporting libraries are packaged together in a single package which can be easily installed, upgraded or removed. MicroStack includes all key OpenStack components : Keystone, Nova, Neutron, Glance, and Cinder. Nevertheless we can only create a instance that has not graphical interface.

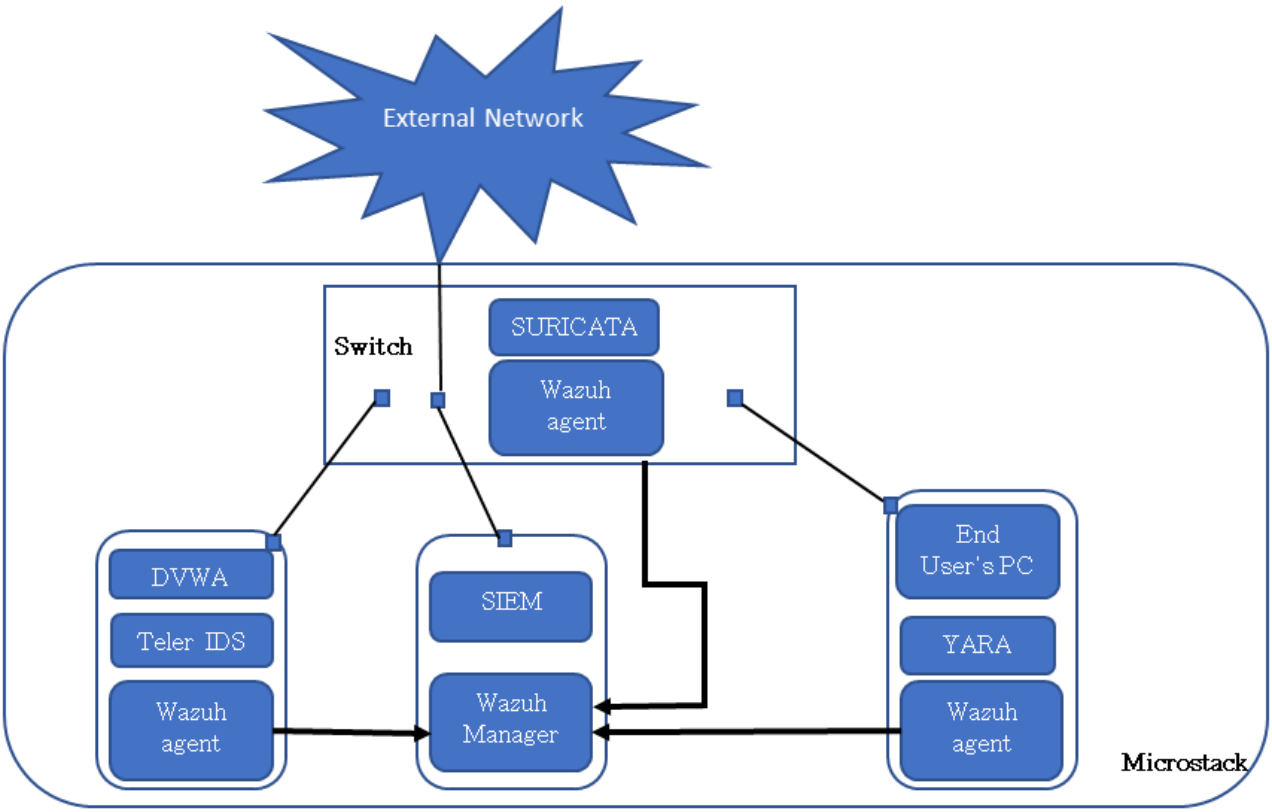


FIGURE 1 – :SOC Architecture

1.2 Components

This part consist to present briefly all components of our SOC. We justify in this the reason of our choice according to their advantages for our project.

1.2.1 Microstack

Microstack is a simple software that enable us to install quickly Openstack with all features. It provides a efficient way to deploy cloud infrastructure in a small environment, it is designed to be easy to set up and use.

what is The reasons of our choice ?

We tried to use Devstack for virtualization but we found a lot of problems using Devstack like: Once we shut down the physical computer, we lost all configuration of the virtualized architecture and in addition it takes a lot of resources to create an instance, so because of that we move to Microstack, which is more simple.

Additionally due to his design ,Microstack is a great choice for our project that need to deploy cloud infrastructure quickly.

Strength and weaknesses

Microstack is a good solution for our project architecture because it :

- can be deploy in a small environment.
- is designed to be lightweight and scalable, making it ideal for development, testing, and continuous integration.
- is designed to be easy to set up and use.

Despite many advantages of this solution, we can notice **some weak point** such as :

- it does not support certain features, such as volume storage, which is necessary for larger deployments.
- it does not have certain security measures in place, such as encryption and authentication, which can be important for enterprise deployments.

1.2.2 DVWA

Damn Vulnerable Web Application, shorter **DVWA**, is a PHP web application that uses MySQL database. Used for education purposes, help web developers to test their skills to better understand the processes of securing web applications.

What is The reasons of our choice ?

Our objective in using this application is to exploit its vulnerabilities of this one to be able to make some common attacks such as SQL injection; to test if our soc works as we hope.

Strenght and weaknesses of DVWA

DVWA has vulnerabilities like XSS, CSRF, SQL injection, file injection, upload flaws and more, which is great for researchers to learn and help others learn about these flaws.

What we can say about limit (for us) of this solution, is that it is an entreprise solution, so we cannot have access to all features for this project.

1.2.3 Wazuh

Wazuh is a security event monitoring platform used to collect, analyze and correlate data to provide threat detection, compliance management and incident response capabilities. It performs real-time analysis of security alerts generated by network devices, servers and applications.

What is The reasons of our choice ?

We choose Wazuh as a SIEM, because it most popular, easy to use and it also provides a complete and easy-to-read documentation.

Advantages of Wazuh

Wazuh has many advantages that we can regroup in to following points:

— a unique security monitoring platform that performs real-time analytic in real time and 100% open source;

— compliance reporting;

— a highly scalable solution thanks to its distributed architecture;

— Infrastructure monitoring (cloud, physical).

1.2.4 HIDS-Wazuh agent

The Wazuh platform consists of multiple components, including the Wazuh agent, which is a key component of the platform. so it's a software component that runs on individual hosts within a network to collect a wide range of information from the host, including system logs, system events, and network traffic for analysis by the Wazuh server.

What is The reasons of our choice ?

the Wazuh agent provides a powerful and flexible HIDS solution that can be tailored to meet a specific needs, this flexibility simplifies management and analysis.

Advantages of Wazuh agent

There are several advantages of using the Wazuh agent :

— Real-time monitoring of system activity, file integrity, and network connections, which allows for quick detection of potential security threats

— Can be installed on a wide range of operating systems, including Windows, Linux, and macOS..

— File integrity monitoring, which can help detect potential attacks or unauthorized changes;

— Customizable rules and alerts;

1.2.5 Suricata

Suricata is an open-source detection engine that can act as an intrusion detection system (IDS) and an intrusion prevention system (IPS).IT uses a rule set and signature language to detect and prevent threats.

What is The reasons of our choice ?

We chose to use Suricata as our NIDS because it is a popular solution compatible with Wazuh, the SIEM we are using. Suricata also offers a simple installation process, as well as an user-friendly rules management system through Suricata-Update. Furthermore, Suricata can use sets of rules made for Snort.

Additionally, The documentation of Suricata is well detailed and the configuration files are clear which helps greatly in the installation process. It supports multithreading which allows better performances and can be used in IPS mode which attempts to stop ongoing attacks by interacting with the firewall and creating new rules.

Weaknesses

The main problems with Suricata are that it generates a lot of alerts and that we were not able to make use of the IPS mode because our network was not setup to be accessible from other physical machines in the room.

1.2.6 Teler

Teler is an real-time intrusion detection and threat alert based on web log. It have many features that we can regroup in the followings points.

— Analysing: Analyze logs and identify suspicious activity in real-time.

— Alerting: provides alerting when a threat is detected, push notifications

— Monitoring

Advantages of Teler

Teler provides minimal configuration and flexible log formats (teler allows any custom log format string!).

Why we choose Teler?

Teler was designed to be a fast, terminal-based threat analyzer. Its core idea is to quickly analyse and hunt threats in real time.

1.2.7 YARA

YARA is a string pattern-matching tool used to confirm the identity of malware by comparing signatures in the code. It uses string signatures, allowing for closer examination of multiple strings of code. If a strain of malware is altered in some manner, it will not affect YARA's ability to identify it. This makes YARA a powerful tool for detecting the newest variants of malware

YARA vs Hashes

The use of string signatures is preferable to the use of hashes for malware detection. Indeed, when a code changes, the hash changes too. This would distort the results.

As opposed to hashes, YARA rules use a set of strings and a Boolean expression to scan and identify families of malware despite variations in the code. **Why we choose YARA?**

The main reason that we choose YARA is that if a strain of malware is altered in some manner, it will not affect YARA's ability to identify it. This makes YARA a powerful tool for detecting the newest variants of malware.

2 Deployment and Rules Configuration

2.1 Microstack

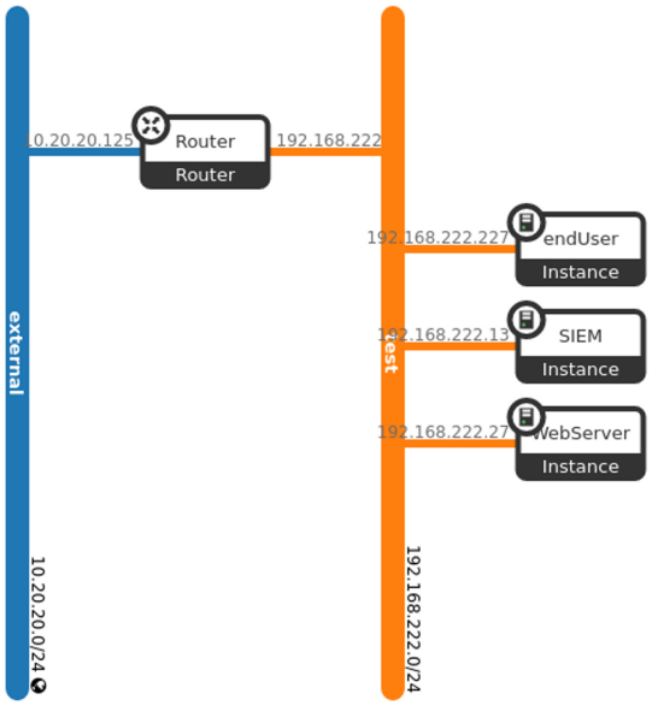


Figure 2 – Architecture under microstack

We decided to use Microstack for our virtual environment based on its advantages and then to deploy it, we follow these steps:

Install MicroStack using the following command:

```
sudo snap install microstack --classic --beta
```

Initialize MicroStack using the following command:

```
sudo microstack.init --auto
```

By the default Microstack provide two sub-networks the test one and the external that is connect to our physical machine.

```
mcca@mcca:~$ microstack.openstack network list
+--------------------------------------+----------+--------------------------------------+
| ID                                   | Name     | Subnets                              |
+--------------------------------------+----------+--------------------------------------+
| 54a3f477-6833-44fe-9827-04963b341838 | test     | 3e3bf1b1-c183-422e-b2e9-4dd0ec46886a |
| 676a3cd5-b1e6-46c9-9cee-c91476222211 | external | 095d2704-387e-4753-9c77-bdb13eeaf93c |
+--------------------------------------+----------+--------------------------------------+
```

Figure 3

In addition Microstack also provides some flavors,the resource needed to create an instance and a cirros image to create an instance on the environment but we can add and delete a new image and flavor whenever we want.

```
mcca@mcca:~$ microstack.openstack flavor list
+----+-----------+-------+------+-----------+-------+-----------+
| ID | Name      |   RAM | Disk | Ephemeral | VCPUs | Is Public |
+----+-----------+-------+------+-----------+-------+-----------+
| 1  | m1.tiny   |   512 |    1 |         0 |     1 | True      |
| 2  | m1.small  |  2048 |   20 |         0 |     1 | True      |
| 3  | m1.medium |  4096 |   20 |         0 |     2 | True      |
| 4  | m1.large  |  8192 |   20 |         0 |     4 | True      |
| 5  | m1.xlarge | 16384 |   20 |         0 |     8 | True      |
+----+-----------+-------+------+-----------+-------+-----------+
```

FIGURE 4

```
mcca@mcca:~$ microstack.openstack image list
+--------------------------------------+--------+--------+
| ID                                   | Name   | Status |
+--------------------------------------+--------+--------+
| b0ef8bd3-8096-4c6f-a6bd-ed5a93cfefcb | cirros | active |
| f2266eaa-aca4-4160-be65-a10081ee0278 | ubuntu | active |
+--------------------------------------+--------+--------+
```

FIGURE 5

To create an instance on Microstack, we are using the following command:

```
microstack.launch ubuntu --name soc --flavor m1.xlarge
```

Each instances in MicroStack environment are assigned to a private IP address that can be used to communicate with other instances within the same network. However, in order to access an instance from outside the network, we assign a floating IP address to it.

IP Address	Description	Mapped Fixed IP Address
10.20.20.224		SIEM 192.168.222.13
10.20.20.17		endUser 192.168.222.227
10.20.20.27		-
10.20.20.191		-
10.20.20.119		WebServer 192.168.222.27

FIGURE 6

Next we associate a security group, that is a virtual firewall that defines a set of rules for inbound and outbound network traffic, for each instance to determine which network traffic is allowed or blocked.

Once the instance is running, we can connect to it via an ssh connection using the following command:

```
ssh -i /home/mcca/snap/microstack/common/.ssh/id_microstack ubuntu@10.20.20.224
```

To enable internet on our Microstack environment, we implemented the following firewall rules on our physical machine.

Displaying 6 items

Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix
Egress	IPv4	Any	Any	0.0.0.0/0
Egress	IPv6	Any	Any	::/0
Ingress	IPv4	Any	Any	-
Ingress	IPv4	ICMP	Any	0.0.0.0/0
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0
Ingress	IPv6	Any	Any	-

Figure 7

```
#!/bin/bash
sudo iptables -t nat -A POSTROUTING -s 10.20.20.1/24 ! -d 10.20.20.1/24 -j
    MASQUERADE
sudo iptables -A FORWARD -i enp0s31f6 -o br-ex -j ACCEPT
sudo iptables -A FORWARD -o br-ex -i enp0s31f6 -j ACCEPT
sudo sysctl net.ipv4.ip_forward=1
```

The router that is linked to the external sub-network will have an interface on our physical machine, then it'll be the access point to our virtual environment.

Figure 8

However, sometimes when we shut down the physical machine it would lose that link, then let's fix that using the following commands.

```
#!/bin/bash
sudo ifconfig br-ex up
sudo ip route add 10.20.20.1 dev br-ex
sudo ip route add 10.20.20.0/24 via 10.20.20.1
sudo ip route add 192.168.222.0/24 via 10.20.20.1
```

2.2 DVWA

We decide to run DVWA on Docker, on Webserver instance. By running DVWA in a container, it can be isolated from other applications and services running on your system, which helps to prevent potential security issues. Docker also allows easy versioning of the DVWA setup, making it

easy to track changes and roll back to previous versions if necessary. Overall, installing DVWA on Docker provides a fast, easy, and secure way to set up a self-contained environment for learning and practicing web application security.

1. Pull image from dockerhub :

```
docker pull vulnerables/web-dvwa
```

2. Start to run the DVWA container:

```
docker container run -d -p 80:80 --name test -v /var/www/html:/var/www/html
    vulnerables/web-dvwa
```

   — **-p 80:80** This option maps the container's port 80 (the default port for HTTP) to the host's port 80. This allows the container's web server to be accessible from the host's web browser.

   — **-v /var/log/apache2:/var/log/apache2**: This option creates a bind mount that maps the host directory **/var/log/apache2** to the container directory **/var/log/apache2** . This allows the container's Apache web server logs to be persisted on the host file system. access.log file is important input to Teler Web IDS, so we have to map it to host system.

   The result:

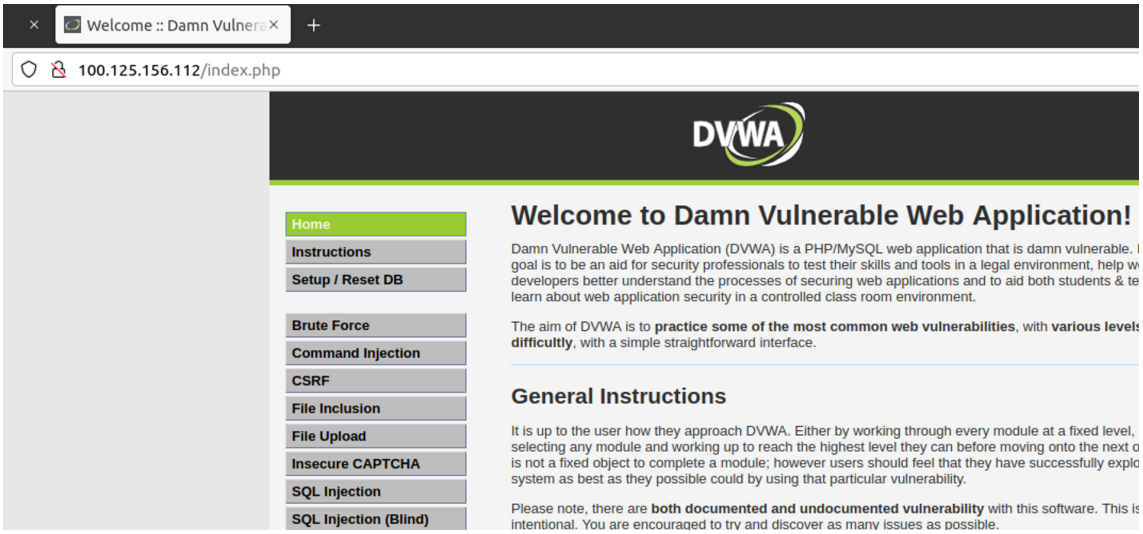


FIGURE 9 – DVWA Application

2.3 Wazuh SIEM

We install Wazuh SIEM on SIEM instance. Installing Wazuh is relatively easy because this one provides the Wazuh installation assistant too.
Download and run the Wazuh installation assistant.

```
curl -sO https://packages.wazuh.com/4.3/wazuh-install.sh && sudo bash ./wazuh-
    install.sh -a
```

Once the assistant finishes the installation, the output shows the access credentials and a message that confirms that the installation was successful.

Then we go Wazuh dashboard:

```
qman@ubuntu:~/Documents$ curl -sO https://packages.wazuh.com/4.3/wazuh-install.sh && sudo bas
h ./wazuh-install.sh -a
16/02/2023 03:51:56 INFO: Starting Wazuh installation assistant. Wazuh version: 4.3.10
16/02/2023 03:51:56 INFO: Verbose logging redirected to /var/log/wazuh-install.log
16/02/2023 03:52:17 INFO: Wazuh repository added.
16/02/2023 03:52:17 INFO: --- Configuration files ---
16/02/2023 03:52:17 INFO: Generating configuration files.
16/02/2023 03:52:18 INFO: Created wazuh-install-files.tar. It contains the Wazuh cluster key,
 certificates, and passwords necessary for installation.
16/02/2023 03:52:18 INFO: --- Wazuh indexer ---
16/02/2023 03:52:18 INFO: Starting Wazuh indexer installation.
16/02/2023 03:53:20 INFO: Wazuh indexer installation finished.
16/02/2023 03:53:21 INFO: Wazuh indexer post-install configuration finished.
16/02/2023 03:53:21 INFO: Starting service wazuh-indexer.
16/02/2023 03:53:45 INFO: wazuh-indexer service started.
16/02/2023 03:53:45 INFO: Initializing Wazuh indexer cluster security settings.
16/02/2023 03:53:50 INFO: Wazuh indexer cluster initialized.
16/02/2023 03:53:50 INFO: --- Wazuh server ---
16/02/2023 03:53:50 INFO: Starting the Wazuh manager installation.
16/02/2023 03:55:26 INFO: Wazuh manager installation finished.
16/02/2023 03:55:26 INFO: Starting service wazuh-manager.
16/02/2023 03:55:46 INFO: wazuh-manager service started.
16/02/2023 03:55:46 INFO: Starting Filebeat installation.
16/02/2023 03:56:00 INFO: Filebeat installation finished.
16/02/2023 03:56:01 INFO: Filebeat post-install configuration finished.
16/02/2023 03:56:01 INFO: Starting service filebeat.
16/02/2023 03:56:03 INFO: filebeat service started.
16/02/2023 03:56:03 INFO: --- Wazuh dashboard ---
16/02/2023 03:56:03 INFO: Starting Wazuh dashboard installation.
16/02/2023 03:56:49 INFO: Wazuh dashboard installation finished.
16/02/2023 03:56:49 INFO: Wazuh dashboard post-install configuration finished.
16/02/2023 03:56:49 INFO: Starting service wazuh-dashboard.
16/02/2023 03:56:50 INFO: wazuh-dashboard service started.
16/02/2023 03:57:17 INFO: Initializing Wazuh dashboard web application.
16/02/2023 03:57:19 INFO: Wazuh dashboard web application initialized.
16/02/2023 03:57:19 INFO: --- Summary ---
16/02/2023 03:57:19 INFO: You can access the web interface https://<wazuh-dashboard-ip>
    User: admin
    Password: p*hC*ANEhOLmsHTdfHY2KJxATZv0r17v
```

FIGURE 10 – Wazuh Installation

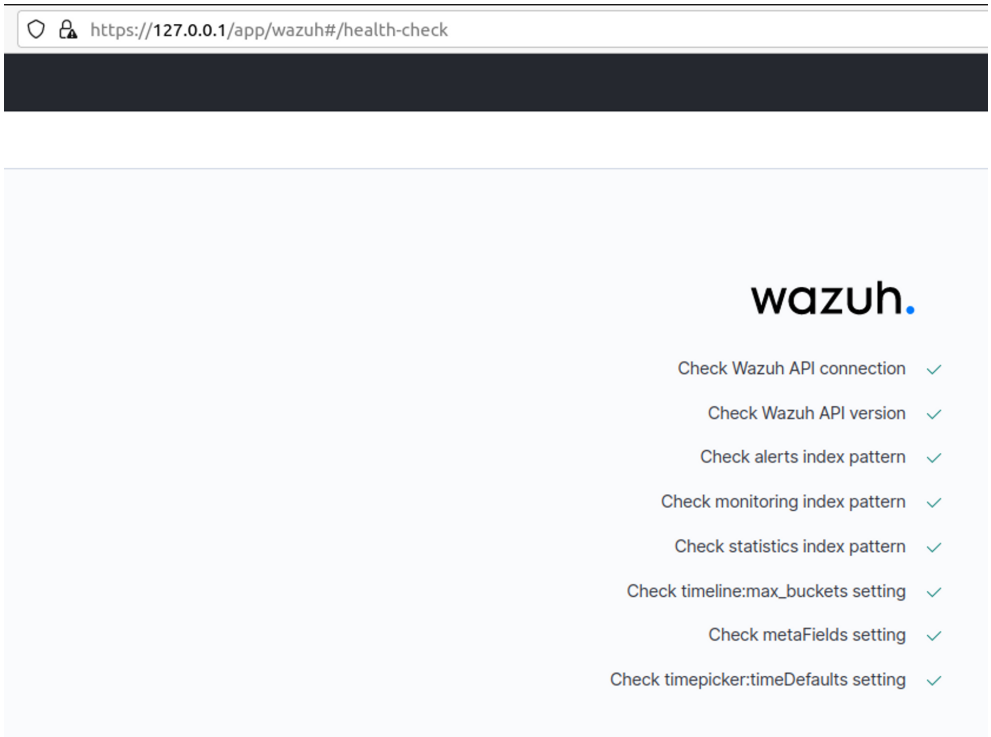


FIGURE 11 – Wazuh Dashboard checking

2.4 HIDS - Wazuh agent

Wazuh also provides feature to deploy Wazuh agent - EDR endpoint easily.

1. Go to module Agent

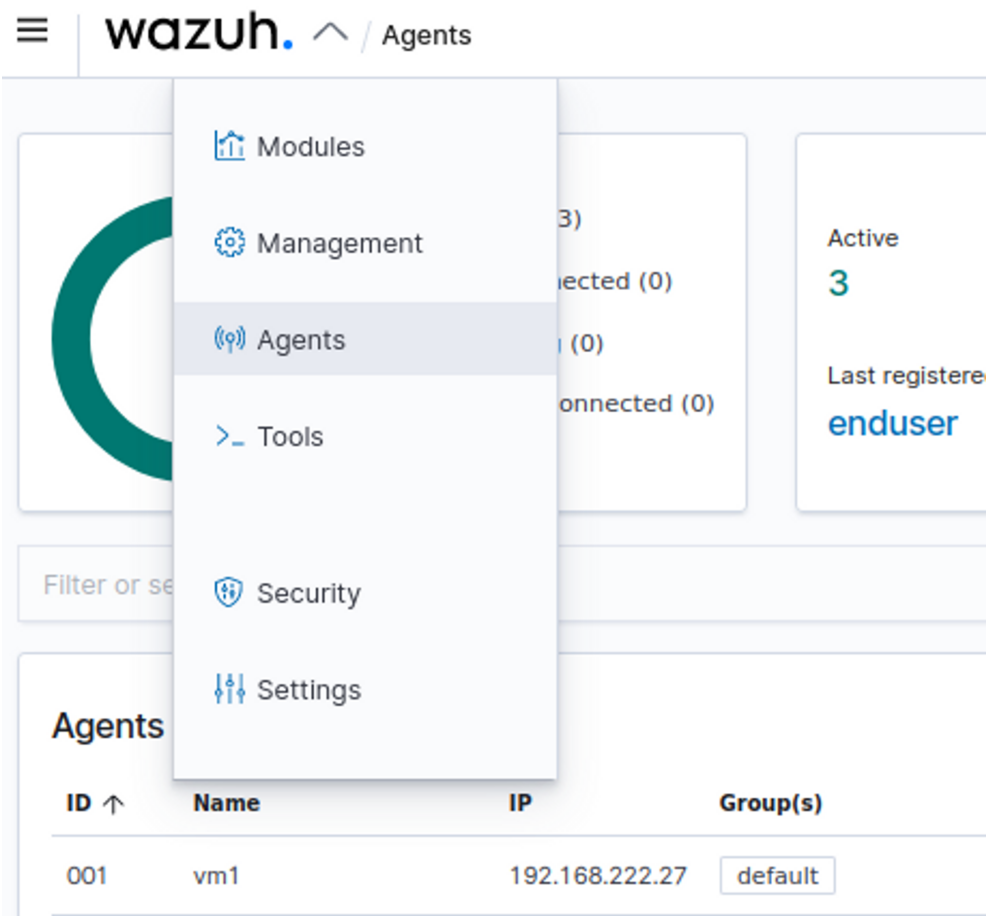


FIGURE 12 – Wazuh agent

2. We can see the Agent dashboard. Then, click the Deploy new agent button.

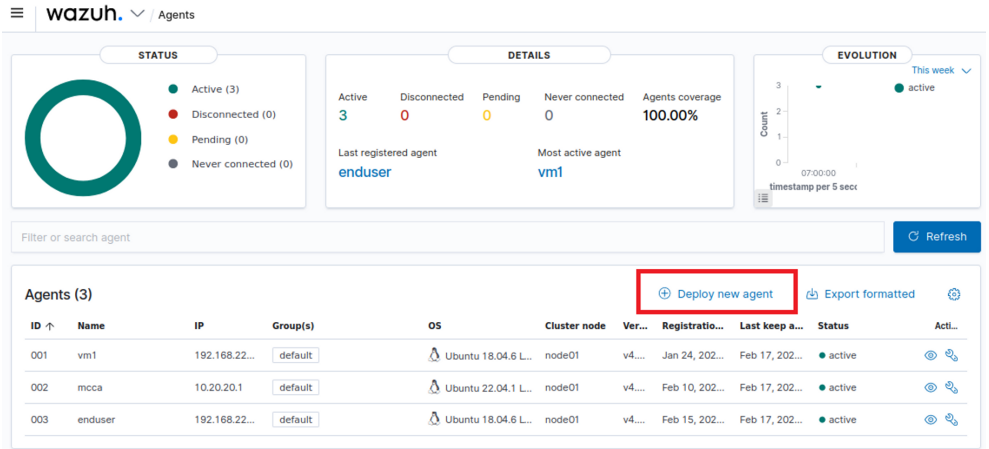


FIGURE 13 – Wazuh agent interface

3. Then, we go to the Deploy agent page. At here, we choose the version of OS to install on.
4. After choosing the options, Wazuh will generate the bash command to download and install Wazuh agent (for Windows, Wazuh generate the powershell command).

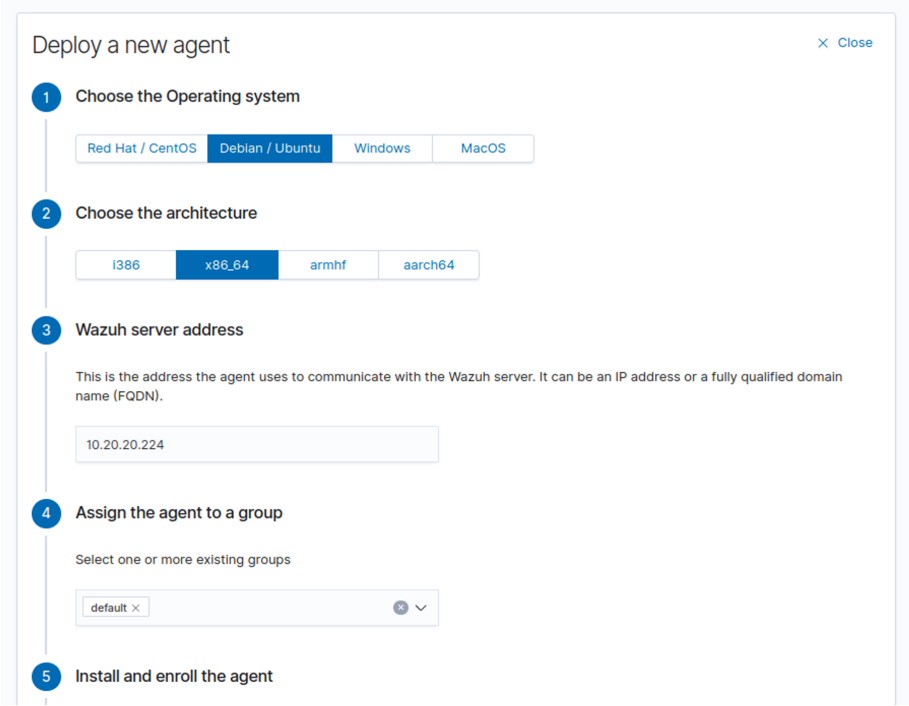


FIGURE 14 – Deploy agent

FIGURE 15 – Wazuh generation of bash command

5. Run the command on endpoint.

```
remnux@remnux:~$ curl -so wazuh-agent-4.3.10.deb https://packages.wazuh.com/4.x/apt/pool/main/w/wazu
h-agent/wazuh-agent_4.3.10-1_amd64.deb && sudo WAZUH_MANAGER='100.65.191.127' WAZUH_AGENT_GROUP='def
ault' dpkg -i ./wazuh-agent-4.3.10.deb
Selecting previously unselected package wazuh-agent.
(Reading database ... 207124 files and directories currently installed.)
Preparing to unpack ./wazuh-agent-4.3.10.deb ...
Unpacking wazuh-agent (4.3.10-1) ...
Setting up wazuh-agent (4.3.10-1) ...
Processing triggers for systemd (245.4-4ubuntu3.17) ...
remnux@remnux:~$ sudo systemctl daemon-reload
remnux@remnux:~$ sudo systemctl enable wazuh-agent
Synchronizing state of wazuh-agent.service with SysV service script with /lib/systemd/systemd-sysv-i
nstall.
Executing: /lib/systemd/systemd-sysv-install enable wazuh-agent
Created symlink /etc/systemd/system/multi-user.target.wants/wazuh-agent.service → /lib/systemd/syste
m/wazuh-agent.service.
remnux@remnux:~$ sudo systemctl start wazuh-agent
```

FIGURE 16 – Installation on endpoint

6. Check on Agent dashboard to see active Agent.

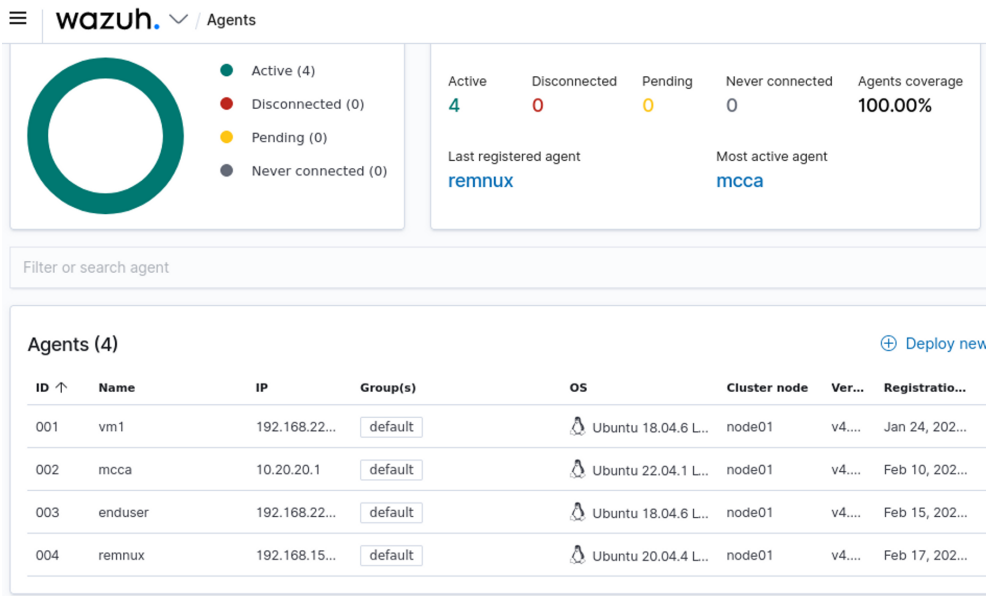


FIGURE 17 – Agent Dashboard

2.5 NIDS - Suricata

We installed Suricata on the physical machine hosting our virtual network in order to monitor the traffic going through the virtual interface. This gives us the option to use Suricata in IPS mode with the firewall installed on the physical machine. On the same machine, we installed a Wazuh agent which collects suricata's logs and sends them to the Wazuh server on the SIEM machine. In order to facilitate the installation of suricata on different machines for testing purposes, we wrote an installation script which automates the process.

```
#!/bin/bash
apt update
#Dependancies
apt-get install  libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev
    libnet1-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev \
                libcap-ng-dev libcap-ng0 make libmagic-dev \
                libnss3-dev libgeoip-dev liblua5.1-dev libhiredis-dev libevent-dev
    \
                rustc cargo jq -y &&
#IPS mode dependancies
apt-get install  libnetfilter-queue-dev libnetfilter-queue1  \
        libnetfilter-log-dev libnetfilter-log1\
        libnfnetlink-dev libnfnetlink0 -y &&
#Install Suricata
add-apt-repository ppa:oisf/suricata-stable -y &&
apt-get update  -y &&
apt-get install suricata -y &&
#replace suricata.yaml
mv /etc/suricata/suricata.yaml /etc/suricata/suricata.yaml.backup &&
cp ./config.yaml /etc/suricata/suricata.yaml &&
#add rulesets from oisf repo
sudo suricata-update update-sources &&
#activate rulesets
sudo suricata-update enable-source et/open &&
sudo suricata-update enable-source oisf/trafficid &&
sudo suricata-update enable-source sslbl/ssl-fp-blacklist &&
#update rules
sudo suricata-update &&
#restart the service
sudo systemctl restart suricata
```

Listing 1 – Suricata installation script

We set the br-ex virtual interface created by openstack for capture in our suricata.yaml configuration file and the network to protect as 10.20.20.0/24 to allow suricata to capture traffic coming in and out of our virtual network, and consider that the physical host is the gateway to the outside.
Finally, we make sure that Wazuh will read Suricata's logs at the right path, by adding this configuration into /var/ossec/etc/ossec.conf file.

```
<ossec_config>
  <localfile>
    <log_format>json</log_format>
    <location>/var/log/suricata/eve.json</location>
  </localfile>
</ossec_config>
```

And restart the Wazuh agent to apply the changes.

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

2.6 Web IDS - Teler

By integrating teler, a lightweight HTTP IDS, with Wazuh, we can further enhance its web detection capabilities, allowing for the detection of web exploits such as HTML injection attacks, directory traversal, and upload attacks.

1. Download the binary excutable file from teler repository, from the link below and extract it. https://github.com/kitabisa/teler/releases
2. Download the sample configuration fileand ensure it is in the same directory as the teler binary. Then, rename it to teler.yaml

```
wget -O teler.yaml https://raw.githubusercontent.com/kitabisa/teler/v2/teler.
    example.yaml
```

3. modify the log-format and logs parameter in the teler.yaml file to use the configuration below.

```
log_format: |
  $remote_addr - $remote_user [$time_local] "$request_method $request_uri
    $request_protocol" $status $body_bytes_sent "$http_referer" "
    $http_user_agent"

logs:
  file:
    active: true
    json: true
    path: "<PATH_TO_LOGFILE>/output.log" #e.g /var/log/teler/output.log
```

4. On endpoint machine, add the configuration block below to the Wazuh agent configuration file located at /var/ossec/etc/ossec.conf to monitor the output.log file generated by teler.

```
<localfile>
  <log_format>syslog</log_format>
  <location><PATH_TO_LOGFILE>/output.log</location>
</localfile>
```

After finish all of these configuration steps, restart the Wazuh agent.
We run this command on the endpoint to start teler IDS:

```
tail -f /var/log/apache2/access.log | ./teler -c teler.yaml
```

```
root@vm1:~/teler# tail -f /var/log/apache2/access.log | ./teler -c teler.yaml

        __    __
       / /___/ /__ ____
      / __/ -_) / -_) __/
      \__/\__/_/\__/_/
                    v2.0.0-dev.3

       infosec@kitabisa.com

[WRN] This tool is under development!
[WRN] Please submit a report if an error occurs.
[INF] Analyzing...
[INF] Listening dashboard on http://localhost:9080
[16/Feb/2023:11:33:32 +0000] [172.17.0.1] [Directory Bruteforce] /v
[16/Feb/2023:11:33:35 +0000] [172.17.0.1] [Bad IP Address] 172.17.0.1
[16/Feb/2023:11:33:37 +0000] [172.17.0.1] [Bad IP Address] 172.17.0.1
[16/Feb/2023:11:33:38 +0000] [172.17.0.1] [Bad IP Address] 172.17.0.1
[16/Feb/2023:11:33:40 +0000] [172.17.0.1] [Bad IP Address] 172.17.0.1
[16/Feb/2023:11:33:43 +0000] [172.17.0.1] [Bad IP Address] 172.17.0.1
[16/Feb/2023:11:33:45 +0000] [172.17.0.1] [Bad IP Address] 172.17.0.1
[16/Feb/2023:11:33:52 +0000] [172.17.0.1] [Common Web Attack: Detects common comment types]
 /vulnerabilities/sqli/?id=1%27+OR+1%3D1+%23&Submit=Submit
```

FIGURE 18 – Starting Teler

Integration into Wazuh

On the Wazuh server, we have to add the custom rules below to the /var/ossec/etc/rules/local-rules.xml file for parsing alerts from Teler.

```
<group name="teler,">
 <rule id="100012" level="10">
   <decoded_as>json</decoded_as>
   <field name="category" type="pcre2">Common Web Attack(: .*)?|CVE
    -[0-9]{4}-[0-9]{4,7}</field>
   <field name="request_uri" type="pcre2">\D.+|-</field>
   <field name="remote_addr" type="pcre2">\d+.\d+.\d+.\d+|::1</field>
   <mitre>
     <id>T1210</id>
   </mitre>
   <description>teler detected $(category) against resource $(request_uri) from $(
    remote_addr)</description>
   </rule>

   <rule id="100013" level="10">
   <decoded_as>json</decoded_as>
   <field name="category" type="pcre2">Bad (IP Address|Referrer|Crawler)</field>
   <field name="request_uri" type="pcre2">\D.+|-</field>
   <field name="remote_addr" type="pcre2">\d+.\d+.\d+.\d+|::1</field>
   <mitre>
     <id>T1590</id>
   </mitre>
   <description>teler detected $(category) against resource $(request_uri) from $(
    remote_addr)</description>
   </rule>

   <rule id="100014" level="10">
   <decoded_as>json</decoded_as>
   <field name="category" type="pcre2">Directory Bruteforce</field>
   <field name="request_uri" type="pcre2">\D.+|-</field>
   <field name="remote_addr" type="pcre2">\d+.\d+.\d+.\d+|::1</field>
   <mitre>
     <id>T1595</id>
   </mitre>
   <description>teler detected $(category) against resource $(request_uri) from $(
    remote_addr)</description>
   </rule>
</group>
```

The teler alerts on Wazuh:

2.7 YARA

Wazuh uses its File Integrity Monitoring (FIM) module to monitor file changes and trigger alerts when the monitored files are modified. Then, Wazuh leverage the YARA ruleset to scan files for malicious activities. The FIM and active response module of Wazuh are used to automatically execute YARA scans when a file change occurs on a monitored endpoint.
This diagram illustrates the flow of events between the different components:

2.7.1 Configuration of FIM

First, we need to configuration for FIM module to look out for changes in the important folders (/home , /root...) since they will be modified. Add this configuration to the file /var/ossec/etc/ossec.conf on Wazuh agent.

Tactic(s)	Description
Reconnaissance	Teler WIDS detected Bad IP Address against resource /vulnerabilities/brute/ from 10.20.20.1
Reconnaissance	Teler WIDS detected Bad IP Address against resource /vulnerabilities/brute/ from 10.20.20.1
Reconnaissance	Teler WIDS detected Bad IP Address against resource /security.php from 10.20.20.1
Reconnaissance	Teler WIDS detected Bad IP Address against resource /security.php from 10.20.20.1
Reconnaissance	Teler WIDS detected Bad IP Address against resource /security.php from 10.20.20.1
Reconnaissance	Teler WIDS detected Bad IP Address against resource /vulnerabilities/brute/ from 10.20.20.1
Reconnaissance	Teler WIDS detected Bad IP Address against resource /security.php from 10.20.20.1
Reconnaissance	Teler WIDS detected Bad IP Address against resource /security.php from 10.20.20.1

FIGURE 19 – Alerts on Wazuh

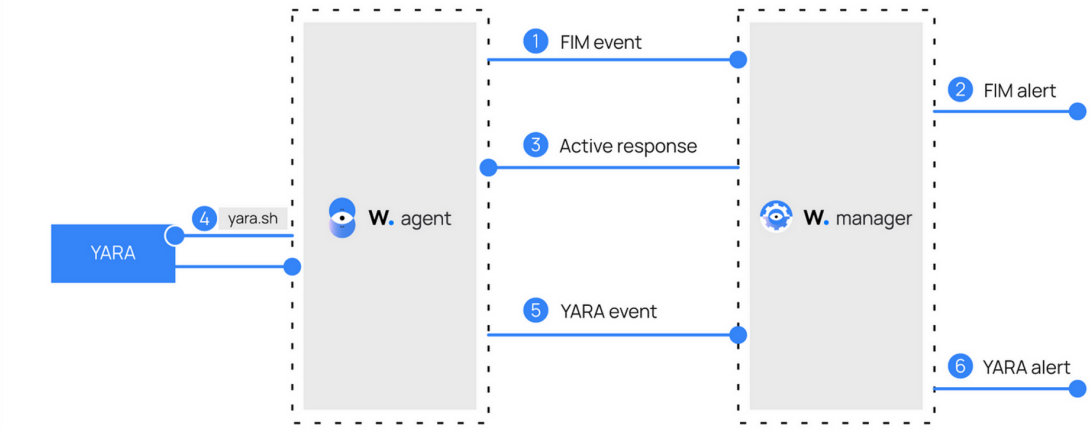


FIGURE 20 – Active response flow

```
<syscheck>
    <disabled>no</disabled>
    <frequency>60</frequency>
    <scan_on_start>yes</scan_on_start>
    <directories realtime="yes">/home,/root</directories>
</syscheck>
```

Then, we restart wazuh-agent service.

2.7.2 Config YARA on User Ubuntu enpoint

1. Install YARA, run this bash:

```
sudo curl -LO https://github.com/VirusTotal/yara/archive/v4.2.2.tar.gz
sudo tar -xvzf v4.2.2.tar.gz -C /usr/local/bin/ && rm -f v4.2.2.tar.gz
cd /usr/local/bin/yara-4.2.2
sudo ./bootstrap.sh && ./configure && make && sudo make install && make check
```

2. Download YARA detection rules:

```
sudo mkdir -p /home/ubuntu/yara/rules
sudo curl 'https://valhalla.nextron-systems.com/api/v1/get' \
```

```
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US,en;q=0.5' \
--compressed \
-H 'Referer: https://valhalla.nextron-systems.com/' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'DNT: 1' -H 'Connection: keep-alive' -H 'Upgrade-Insecure-Requests: 1' \
--data 'demo=demo&apikey
    =1111111111111111111111111111111111111111111111111111111111111111&format=
    text' \
-o /home/ubuntu/yara/rules/yara_rules.yar
```

3. Create a yara.sh script in the /var/ossec/active-response/bin/ directory. This is necessary for the Wazuh-YARA active response scans. Then, give it the execution permission.

```
#!/bin/bash
# Wazuh - Yara active response

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

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

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

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

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

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

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

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

if [[ $yara_output != "" ]]
then
    # Iterate every detected rule and append it to the LOG_FILE
    while read -r line; do
         echo "wazuh-yara: INFO - Scan result: $line" >> ${LOG_FILE}
         sleep 1
         # Removing file
         rm -f $FILENAME
         echo "wazuh-yara: INFO - Successfully removed threat: $line" >> ${
    LOG_FILE}
    done <<< "$yara_output"
fi
```

```

exit 0;
```

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

4. Install the Linux utility jq to allow the /var/ossec/active-response/bin/yara.sh script to process the JSON input:

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

Then, restart wazuh-agent service.

2.7.3 Config decoder and rules on Wazuh Server

1. To extract information from YARA scan results, add the following decoders to the /var/ossec/etc/decoders/local_decoder.xml file.

```
<decoder name="yara_decoder">
    <prematch>wazuh-yara:</prematch>
</decoder>

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

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

2. The rules detect FIM events in the monitored directory. They also alert when the YARA integration finds malware. Add the following rules to the /var/ossec/etc/rules/local_rules.xml file.

```
<group name="syscheck,">
    <rule id="100300" level="1">
        <if_sid>550</if_sid>
        <field name="file">/home/</field>
        <description>File modified in /home/* directory.</description>
    </rule>
    <rule id="100301" level="1">
        <if_sid>554</if_sid>
        <field name="file">/home/</field>
        <description>File added to /home/* directory.</description>
    </rule>
</group>

<group name="yara,">
  <rule id="108000" level="0">
    <decoded_as>yara_decoder</decoded_as>
    <description>Yara grouping rule</description>
  </rule>
  <rule id="108001" level="12">
    <if_sid>108000</if_sid>
    <match>wazuh-yara: INFO - Scan result: </match>
    <description>File "$(yara_scanned_file)" is a positive match. Yara rule: $
    (yara_rule)</description>
  </rule>
</group>
```

3. Then, we configures the active response module to trigger after the rule 100300 and 100301 are fired. Add this configuration to the Wazuh server /var/ossec/etc/ossec.confconfiguration file:

```
<ossec_config>
  <command>
    <name>yara_linux</name>
    <executable>yara.sh</executable>
    <extra_args>-yara_path /usr/local/bin -yara_rules /home/ubuntu/yara/rules/
    yara_rules.yar</extra_args>
    <timeout_allowed>no</timeout_allowed>
  </command>

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

After configuration, restart wazuh-manager service.
Finish, the result will show in Incident Case section.

2.8 Virustotal Module on Wazuh

In this case, we will combine HIDS Wazuh agent and Virustotal Module of Wazuh server to verify if the file is malicious or not.
First, We have to get API key from Virustotal to use Module Virustotal:

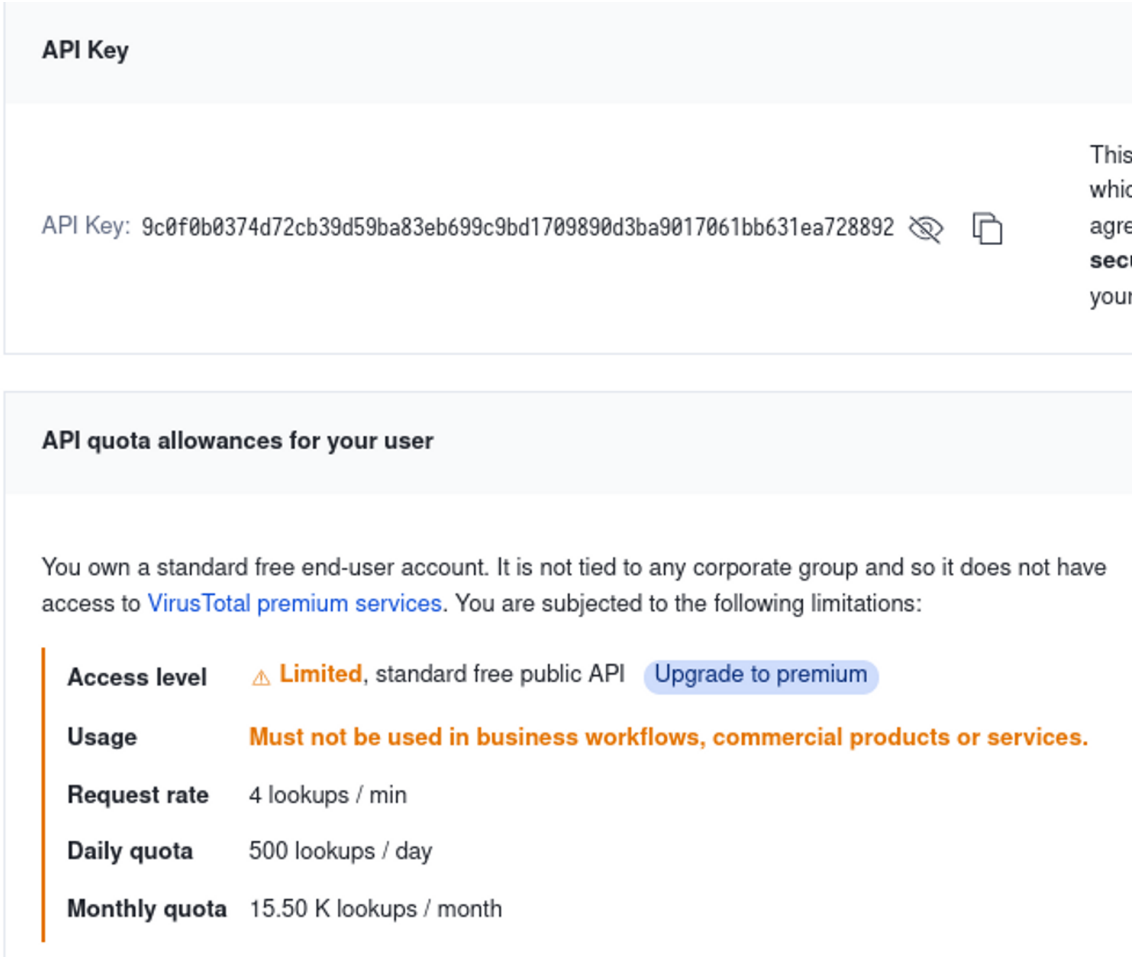


Figure 21 – API Key

2.8.1 Config monitoring on Wazuh Server

1. We config in the file vim /var/ossec/etc/ossec.conf following:

```
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key>${VIRUSTOTAL_API_KEY}</api_key>
    <rule_id>100200,100201,100300,100301</rule_id>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```

2. Config FIM rules - we do the same like the YARA configuration part. But, We will set rules to detect /root folder in this case. Add the following custom rules to /var/ossec/etc/rules/local_rules.xml

```
<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">
    <!-- Rules for Linux systems -->
    <rule id="100200" level="7">
        <if_sid>550</if_sid>
        <field name="file">/root</field>
        <description>File modified in /root directory.</description>
    </rule>
    <rule id="100201" level="7">
        <if_sid>554</if_sid>
        <field name="file">/root</field>
        <description>File added to /root directory.</description>
    </rule>
</group>
```

2.8.2 Config active response on Wazuh server

Once VirusTotal identifies a file as a threat, Wazuh will trigger an active response to remove the file from the system.

1. Configuration command for active response, add this configuration into /var/ossec/etc/ossec.conf file.

```
<ossec_config>

    <command>
        <name>remove-threat</name>
        <executable>remove-threat.sh</executable>
        <timeout_allowed>no</timeout_allowed>
    </command>

    <active-response>
        <disabled>no</disabled>
        <command>remove-threat</command>
        <location>local</location>
        <rules_id>87105</rules_id>
    </active-response>

</ossec_config>
```

   Active response is triggered by rule 87105 which is tripped when VirusTotal identifies a file as malicious.

2. Add rule to monitoring the result of active response in the file var/ossec/etc/rules/local_rules.xml . These rules trigger when a malicious file is removed by active response or if an error occurred removing the file.

```
<group name="virustotal,">
  <rule id="100092" level="12">
    <if_sid>657</if_sid>
    <match>Successfully removed threat</match>
    <description>$(parameters.program) removed threat located at $(parameters.
    alert.data.virustotal.source.file)</description>
  </rule>

  <rule id="100093" level="12">
    <if_sid>657</if_sid>
    <match>Error removing threat</match>
    <description>Error removing threat located at $(parameters.alert.data.
    virustotal.source.file)</description>
  </rule>
</group>
```

After config, we have to restart wazuh-manager service on Wazuh server.
Then, we enable Virustotal module on Wazuh dashboard.

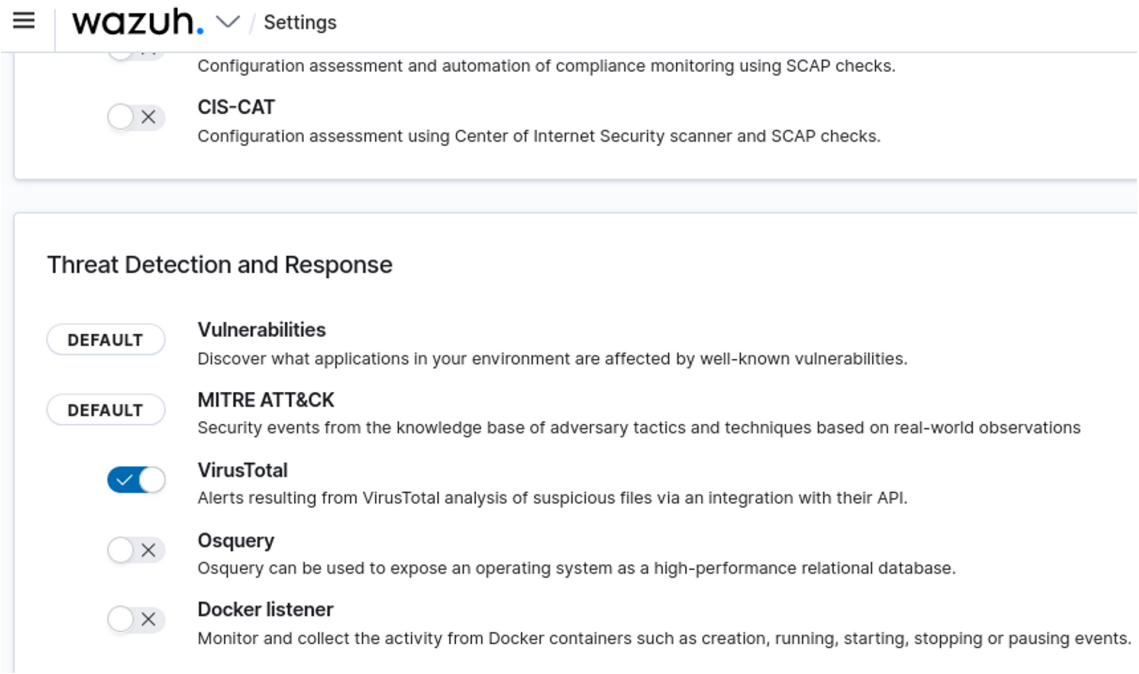


FIGURE 22 – Wazuh Dashboard

2.8.3 Configuration on Wazuh agent

1. Config to file integrity monitoring settings in /var/ossec/etc/ossec.conf to monitor /root in real time.

```
<syscheck>
  <directories whodata="yes">/root</directories>
</syscheck>
```

2. Create active response script at /var/ossec/active-response/bin/remove-threat.sh

```
#!/bin/bash

LOCAL=`dirname $0`;
cd $LOCAL
cd ../

PWD=`pwd`
```

```
read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
COMMAND=$(echo $INPUT_JSON | jq -r .command)
LOG_FILE="${PWD}/../logs/active-responses.log"

#------------------------ Analyze command -------------------------#
if [ ${COMMAND} = "add" ]
then
 # Send control message to execd
 printf '{"version":1,"origin":{"name":"remove-threat","module":"active-response"},"command":"check_keys", "parameters":{"keys":[]}}\n'

 read RESPONSE
 COMMAND2=$(echo $RESPONSE | jq -r .command)
 if [ ${COMMAND2} != "continue" ]
 then
   echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Remove threat active response aborted" >> ${LOG_FILE}
   exit 0;
 fi
fi

# Removing file
rm -f $FILENAME
if [ $? -eq 0 ]; then
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Successfully removed threat" >> ${LOG_FILE}
else
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Error removing threat" >> ${LOG_FILE}
fi

exit 0;
```

3. Install the Linux utility jq to allow the /var/ossec/active-response/bin/remove-threat.sh script to process the JSON input, and set execution to the remove-threat.sh file:

```
sudo apt-get install -y jq
chmod 750 /var/ossec/active-response/bin/remove-threat.sh
chown root:wazuh /var/ossec/active-response/bin/remove-threat.sh
```

Then, restart the wazuh-agent service.
Finish, the result will show in Incident Case section.

2.9 Configuration to dettect DDOS

In this case, we config the rule to detect DDOS attack by GoldenEye tool.

1. Add rule in the file /var/ossec/etc/rules/local_rules.xml on Wazuh server.

```
<group name="custom_active_response_rules,">
  <rule id="100200" level="12">
    <if_sid>86600</if_sid>
    <field name="event_type">^alert$</field>
    <match>ET DOS Inbound GoldenEye DoS attack</match>
    <description>GoldenEye DoS attack has been detected. </description>
    <mitre>
      <id>T1498</id>
    </mitre>
  </rule>
</group>
```

2. Config the active response to this alert
   Config in this config file '/var/ossec/etc/ossec.conf'
   Wazuh includes an out-of-box 'firewall-drop'script that adds the IP address extracted from an alert to the monitored endpoints firewall block list. Configure the 'firewall-drop'active response on the Wazuh server using the following steps:

```
<ossec_config>
  <command>
    <name>firewall-drop</name>
    <executable>firewall-drop</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>

<active-response>
    <command>firewall-drop</command>
    <location>local</location>
    <rules_id>100200</rules_id>
    <timeout>180</timeout>
  </active-response>
</ossec_config>
```

3 Incident Case

3.1 Network Scanning, Attack and Detection

3.1.1 Nmap scanning

Do the reconnaissance step using Nmap tool to scan the hosts 10.20.20.119 to identify open ports, services running on those ports, operating system information, and vulnerabilities that can be exploited.

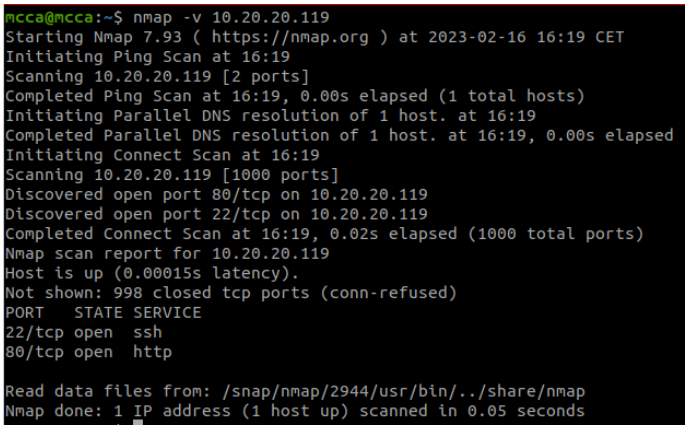


FIGURE 23

When we scan using nmap the web app, our NIDS Suricata will pick up alerts and send them to SIEM Wazuh as we can see below.

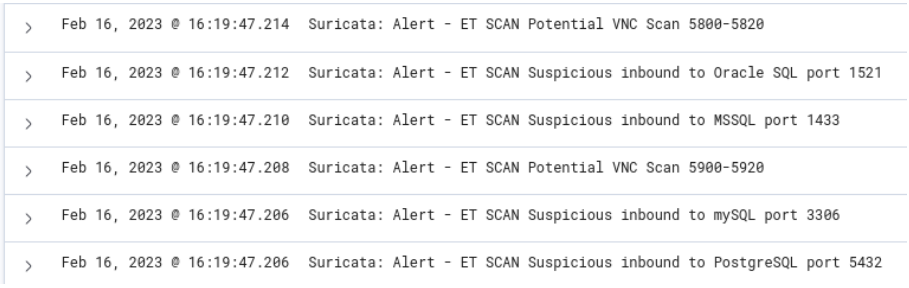


FIGURE 24

Here we can see more details about the alert.

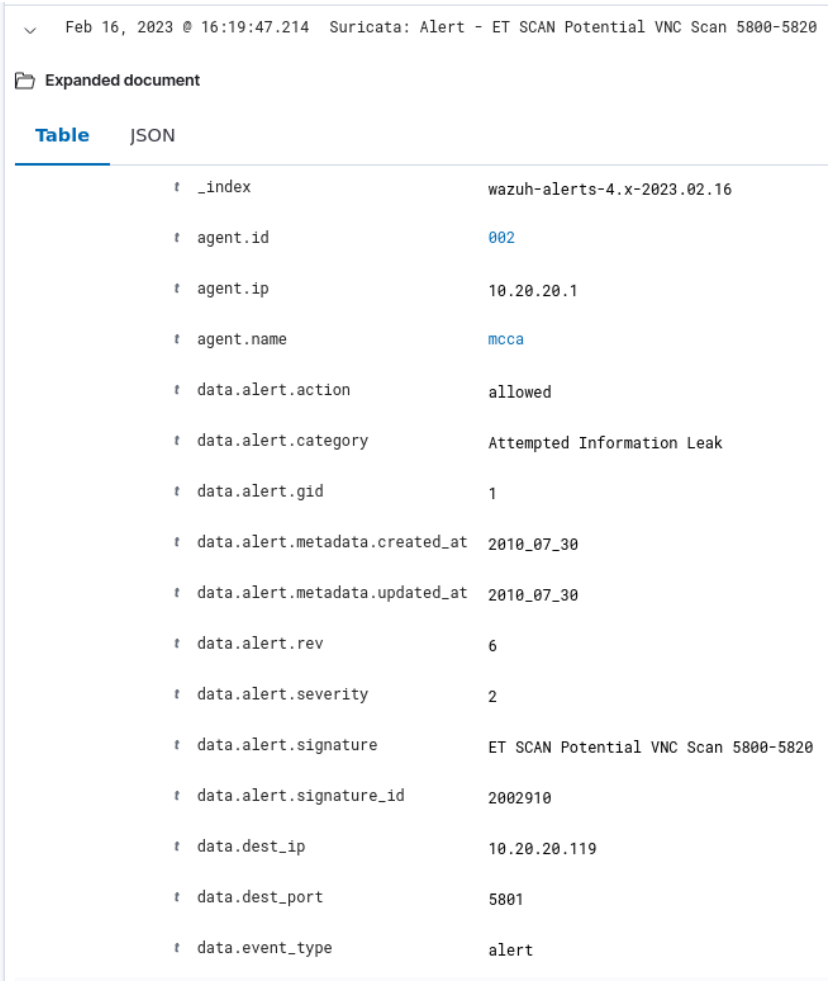


FIGURE 25 – Phishing Attack

To respond to Nmap scanning attack, there are several steps we can take:

— Identify Source: Identify the source of the scan. This can be done by checking our SIEM log alerts, network logs, and firewall logs to determine the scanner's IP address.

— Block the source: Once the source of the scan has been identified, block the IP address to prevent further scanning attempts. This can be done by adding the IP address to a firewall rule or by configuring a network access control system.

— Implement security measures: Implement security measures to prevent future Nmap scanning attacks. This can include network segmentation, network access controls, and monitoring tools that can detect and prevent network scanning.

3.1.2 Nikto scanning

Using Nikto to scan the web server to identify vulnerabilities, misconfigurations, and security weaknesses in web servers and web applications.

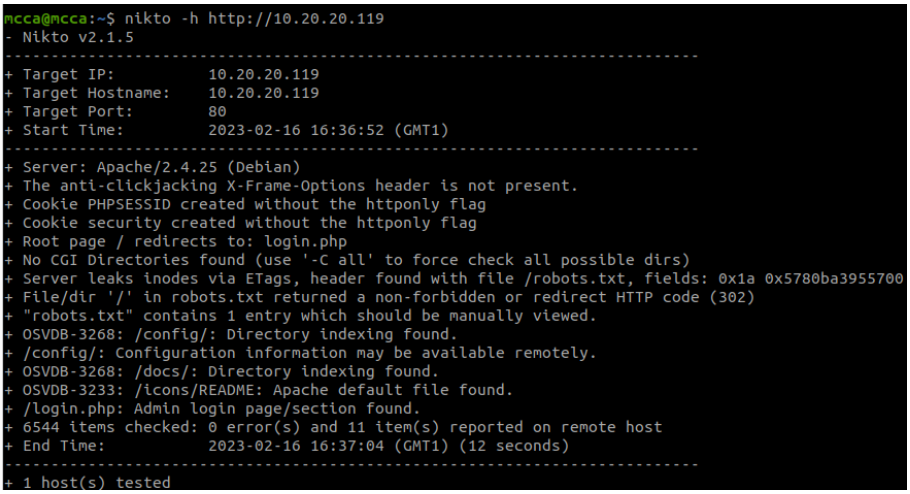


FIGURE 26

As we can see here, our SIEM detects this as a web application attack.

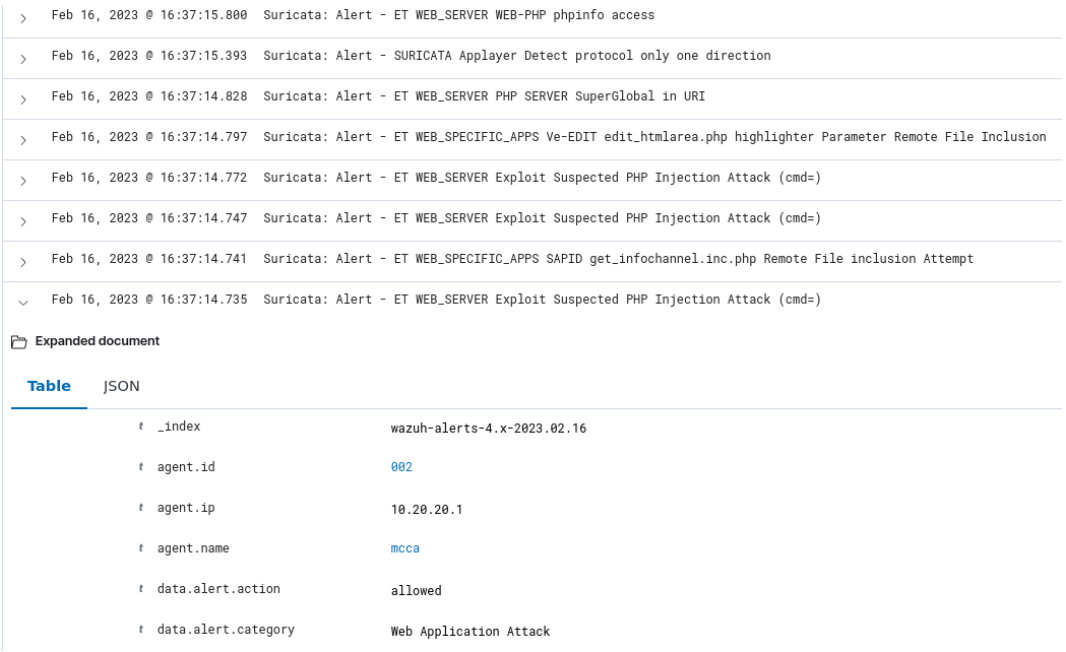


FIGURE 27

And we find the nikto as user agent in this alert.

t data.event_type	alert
t data.flow.bytes_toclient	14677
t data.flow.bytes_toserver	7065
t data.flow.pkts_toclient	27
t data.flow.pkts_toserver	29
t data.flow.start	2023-02-16T16:37:03.165368+0100
t data.flow_id	535182144669176.000000
t data.http.hostname	10.20.20.119
t data.http.http_content_type	text/html
t data.http.http_method	GET
t data.http.http_user_agent	Mozilla/5.00 (Nikto/2.1.5) (Evasions:None) (Test:006019)
t data.http.length	314
t data.http.protocol	HTTP/1.1

FIGURE 28

To respond to Nikto attack, there are several steps we can take:

— Identify the Attack: Use the web server logs or our SIEM to identify the Nikto scan.

— Evaluate the Scan Results: Review the Nikto scan results to identify any vulnerabilities that were detected.

— Patch Vulnerabilities: Once the vulnerabilities are identified, apply the necessary patches or mitigations to fix the issues. This may involve updating software, configuring access controls, or removing unnecessary services.

3.1.3 Ddos attack

Using GoldenEye to test the resilience of web servers to DoS attacks. It is designed to generate a large volume of traffic and requests to a target web server, simulating the effects of a DoS attack on the server.

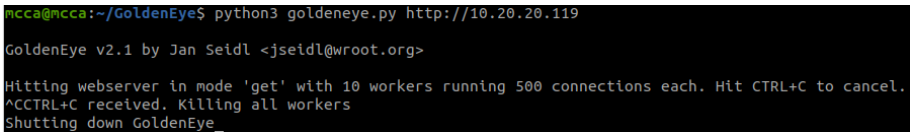


FIGURE 29

b As we can see the SIEM detects GoldenEye's DDOS attack.

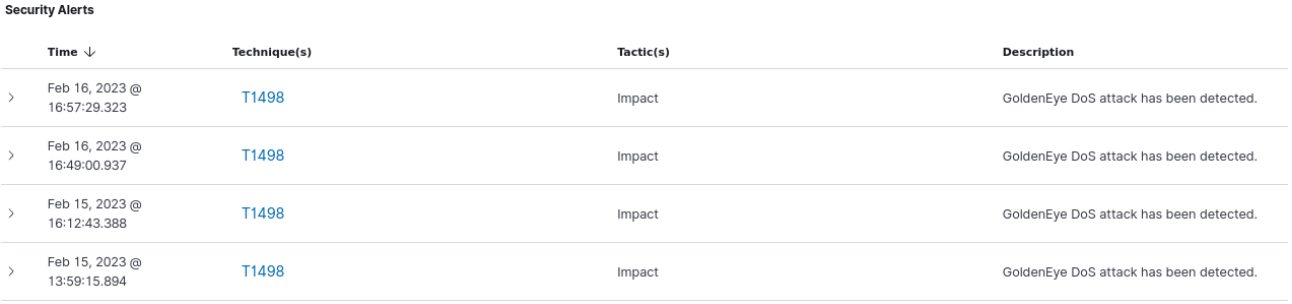


FIGURE 30

And then it tries to automatically respond to the attack.

data.parameters.program	active-response/bin/firewall-drop
data.version	1
decoder.name	ar_log_json
full_log	2023/02/15 13:59:15 active-response/bin/firewall-drop: {"version":1,"origin":{"name":"node01","m {"timestamp":"2023-02-15T12:59:15.894+0000","rule":{"level":12,"description":"GoldenEye DoS at Service"]},"firedtimes":1,"mail":true,"groups":["custom_active_response_rules"]},"agent":{"id":"002 \":\"2023-02-15T13:59:14.060468+0100\",\"flow_id\":1987658638906161,\"in_iface\":\"br-ex\",\"e \",\"metadata\":{\"flowbits\":[\"ET.formdata\",\"http.dottedquadhost\"]},\"tx_id\":0,\"alert\":{\"actior \":\"Detection of a Denial of Service Attack\",\"severity\":2,\"metadata\":{\"created_at\":[\"2014_03 \":\"/?vstDMO1J=VKbNtI27xHYQAku3E5&N42A=RxwolnxfUQwyHkAj&5F5QE=vn6slQD\",\"http_us \":\"text/html\",\"http_refer\":\"http://10.20.20.119/n6hq6e?krT607hhLV=MdAHBv0WeebBA4MNW Xb6h7=AYbN6gxhccXWeYm\",\"http_method\":\"GET\",\"protocol\":\"HTTP/1.1\",\"status\":302,\"r \":879,\"bytes_toclient\":685,\"start\":\"2023-02-15T13:59:07.873265+0100\"}}","decoder":{"name {"srcip":"10.20.20.1","timestamp":"2023-02-15T13:59:14.060468+0100","flow_id":"19876586389(ex","event_type":"alert","src_ip":"10.20.20.1","src_port":"57386","dest_ip":"10.20.20.119","dest_pc {"action":"allowed","gid":"1","signature_id":"2018208","rev":"3","signature":"ET DOS Inbound Golde ["2014_03_05"],"updated_at":["2020_04_28"]}},"http":{"hostname":"10.20.20.119","url":"/?vstDMO 6.0; Windows NT 5.1; .NET CLR 1.3.7093; WOW64)","http_content_type":"text/html","http_refer":"h u0StIlUv4=2qorXLTQm8ir&U2L=nyT87DRyw2N0V3PA6G&4tnt=YB7&Xb6h7=AYbN6gxhccXWeYm" {"pkts_toserver":"4","pkts_toclient":"3","bytes_toserver":"879","bytes_toclient":"685","start":"2023
id	1676465957.810095
input.type	log
location	/var/ossec/logs/active-responses.log
manager.name	soc
rule.description	Host Blocked by firewall-drop Active Response

FIGURE 31

To respond to Ddos attack, there are several steps we can take:

— Detect the attack: By consulting our SIEM logs.

— Create a whitelist of the source IPs and protocols you must allow if prioritizing traffic during an attack.

— Confirm DNS time-to-live (TTL) settings for the systems that might be attacked. Lower the TTLs, if necessary, to facilitate DNS redirection if the original IPs get attacked.

— Harden the configuration of network, OS, and application components that may be targeted by DDoS.

3.2 Web Attack and Detection

3.2.1 SQL Injection attack

In this case We are going to do SQL Injection in order to extract all passwords with payload into a web application's database backend.

```
' UNION SELECT user, password FROM users#
```

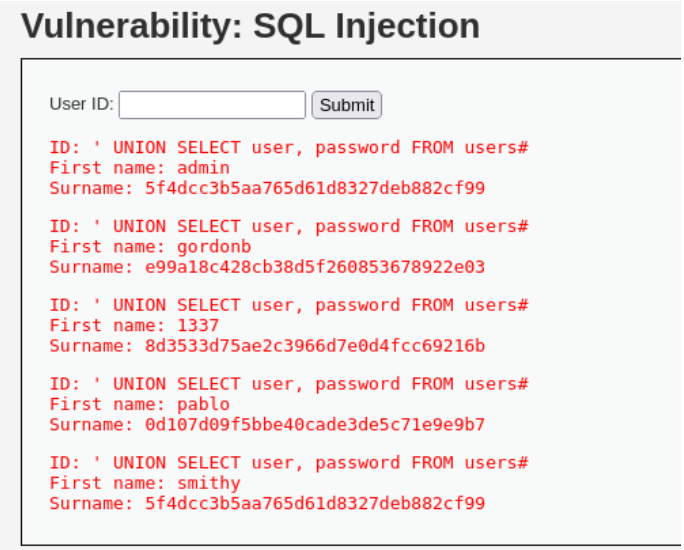


FIGURE 32

As we can see the SIEM detects SQL Injection attack.

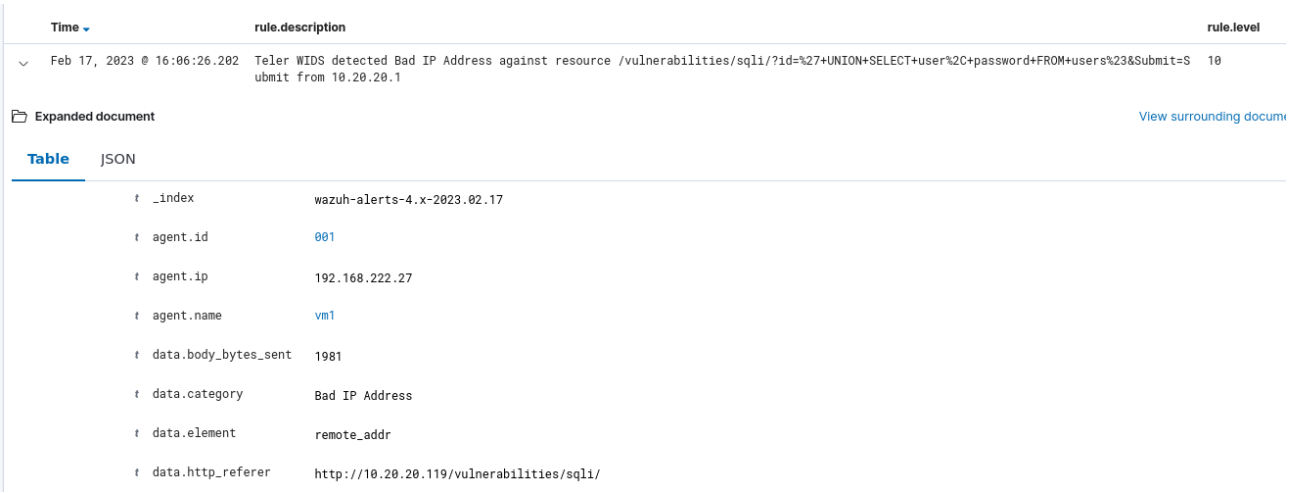


FIGURE 33

To respond to a SQL injection attack, there are several steps we can take:

— Identify and isolate the affected system: As soon as you suspect a SQL injection attack, isolate the affected system to prevent further damage.

— Analyze and contain the attack: Analyze the attack and determine how the attacker is gaining access to the system. Once you have identified the entry point, you can take steps to contain the attack.

— Fix the vulnerability: Patch the vulnerability that allowed the SQL injection attack to take place. This may involve modifying the code of the affected application, tightening access controls, or other security measures.

— Contain the attack: Take immediate action to contain the attack to prevent further damage. This may involve blocking the source IP address, disconnecting affected systems from the network, or stopping the affected processes.

3.2.2 Webshell upload

In this case we are going to add a web shell into this input to see if we can gain remote access to a web server and execute arbitrary commands.In this case we are going to add a web shell into this input to see if we can gain remote access to a web server and execute arbitrary commands.

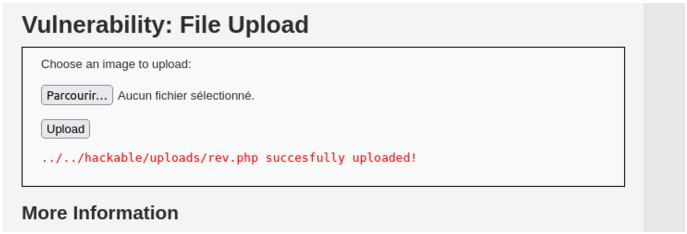


FIGURE 34

As we can see we have access to the web server and then we can execute some commands.

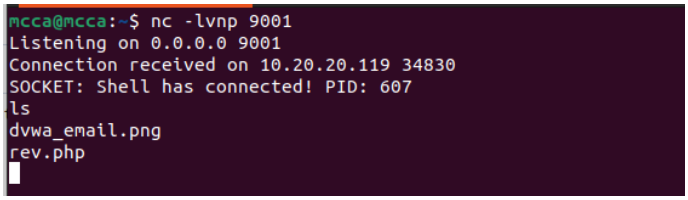


FIGURE 35

And our wazuh SIEM shows us the detection of this attack by our OSSEC HDIS.

FIGURE 36

Responding to a webshell upload attack involves several steps:

— Hardening the Web Server : The configuration of web servers and web applications should be strong enough to protect from web shell and other threats. User input validation can prevent local or remote file inclusion vulnerabilities.

— Limited Access: It is necessary to follow the limited access policy of the Web Shell installation to the web server. For example, the number of ports that have access to the web server can be kept as low as possible to avoid the possibility of the web shell being installed from unknown ports.

— Remove the Unused Web Server Functions : It is essential to disable unnecessary web functions used to communicate with the webserver scripts, which can be done by shutting off default configuration in any products or technologies.

3.2.3 File Inclusion

In this attack, we want to exploit vulnerabilities in a web application that includes and executes remote files on the server. We're going to inject a file path into a parameter of a web application that accepts user input, in this case a page parameter.

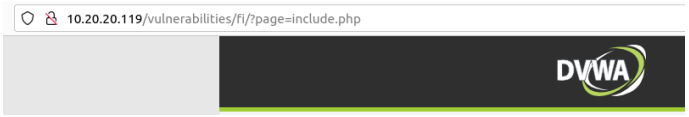


FIGURE 37

We use directory traversal techniques to specify the path to the "../../etc/passwd" file. in order to check if the web application is not properly validating or sanitizing user input. as we can see we have access to this file.

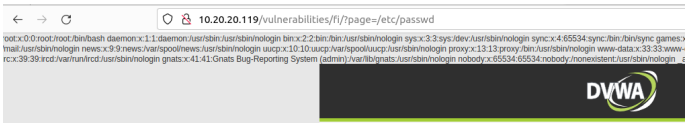
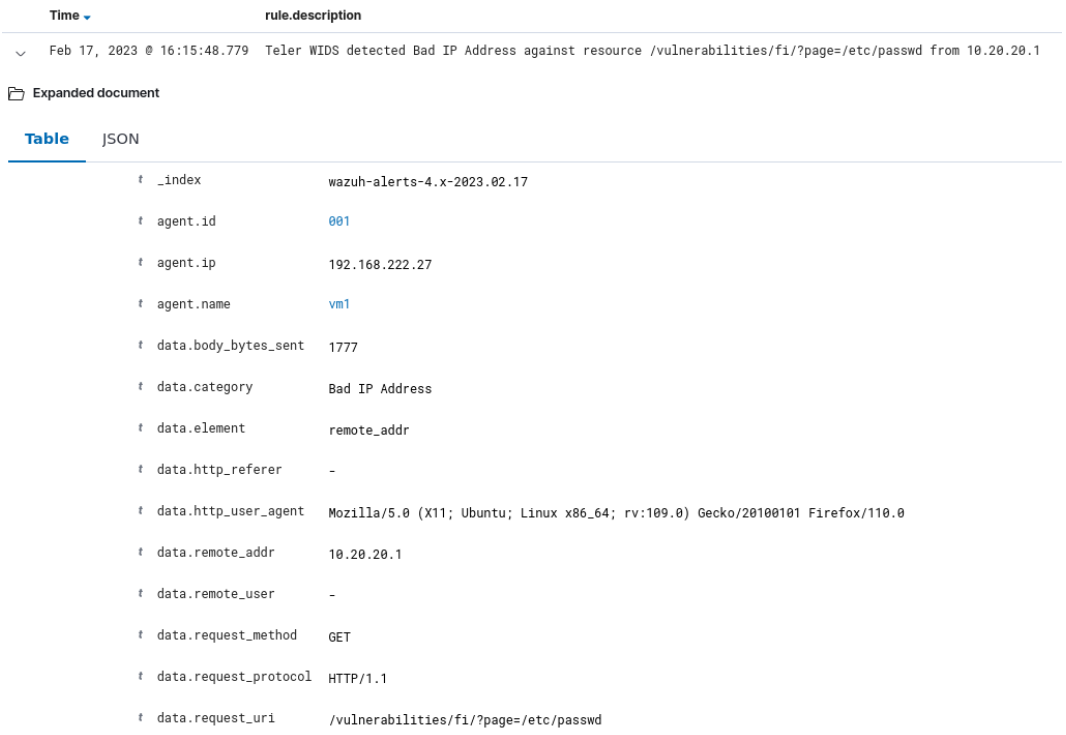


FIGURE 38

We can see the SIEM detects the file inclusion attack .



Time	rule.description
Feb 17, 2023 @ 16:15:48.779	Teler WIDS detected Bad IP Address against resource /vulnerabilities/fi/?page=/etc/passwd from 10.20.20.1

Expanded document

Table JSON

_index	wazuh-alerts-4.x-2023.02.17
agent.id	001
agent.ip	192.168.222.27
agent.name	vm1
data.body_bytes_sent	1777
data.category	Bad IP Address
data.element	remote_addr
data.http_referer	-
data.http_user_agent	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/110.0
data.remote_addr	10.20.20.1
data.remote_user	-
data.request_method	GET
data.request_protocol	HTTP/1.1
data.request_uri	/vulnerabilities/fi/?page=/etc/passwd

FIGURE 39

To respond to File Inclusion, there are several steps we can take:

— ID assignation : save your file paths in a secure database and give an ID for every single one, this way users only get to see their ID without viewing or altering the path.

— Whitelisting : use verified and secured whitelist files and ignore everything else.

— Use databases : don't include files on a web server that can be compromised, use a database instead.

— Better server instructions : make the server send download headers automatically instead of executing files in a specified directory.

3.3 Malware Detection

Attack

In this case, we will assume that the user is attacked by social engineering. Attackers send emails containing malicious download links and trick users into downloading malicious code. By using YARA and the Virustotal module, we will detect and respond to this incident.

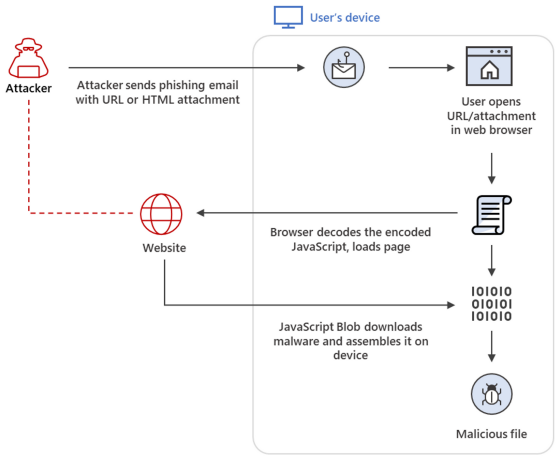


FIGURE 40

To do the demos, we downloaded some malware samples to endUser machine: https://github.com/InQuest/malware-samples

3.3.1 Malware detection with YARA rule

Detection

Since we are monitoring the */home* folder, We will download some malware that has been detected by configured rules.

Feb 16, 2023 @ 07:18:00.035	File "/home/ubuntu/malware/webshell" is a positive match. Yara rule: Webshell_Worse_Linux_Shell_php_RID3323	12	108001
Feb 16, 2023 @ 07:17:58.032	File "/home/ubuntu/malware/vpn_filter" is a positive match. Yara rule: MAL_ELF_VPNFilter_3_RID2D6C	12	108001

Table JSON Rule

@timestamp	2023-02-16T15:17:58.032Z
_id	pPjNWoYBYPSvssXuwlg9
agent.id	003
agent.ip	192.168.222.227
agent.name	enduser
data.log_type	INFO
data.yara_rule	MAL_ELF_VPNFilter_3_RID2D6C
data.yara_scanned_file	/home/ubuntu/malware/vpn_filter
decoder.name	yara_decoder
full_log	wazuh-yara: INFO - Scan result: MAL_ELF_VPNFilter_3_RID2D6C /home/ubuntu/malware/vpn_filter
id	1676560678.11972366
input.type	log
location	/var/ossec/logs/active-responses.log
manager.name	soc

FIGURE 41

Response

This diagram show the how Wazuh and YARA work together to response to a malware incident case.

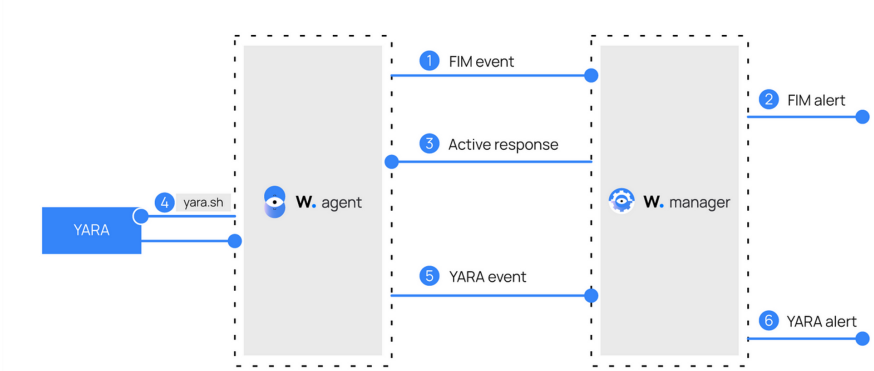


FIGURE 42

After getting FIM alert, Wazuh will trigger the active response to run YARA to scan (yara.sh file). If YARA detect a malware, It will detete and send the result alert (YARA event) to Wazuh.

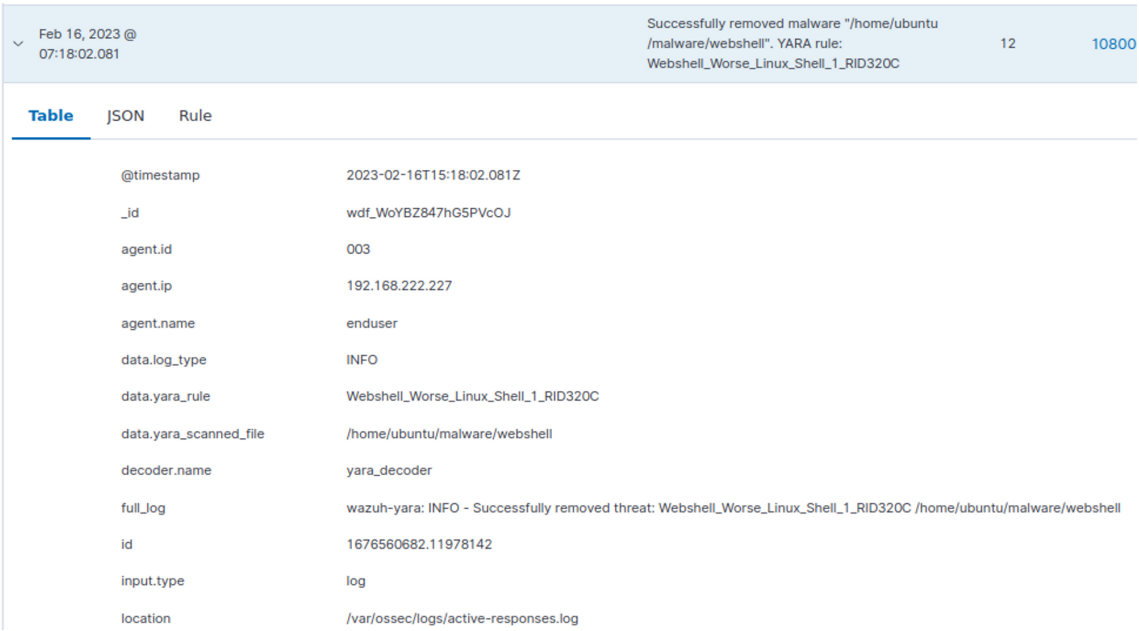


FIGURE 43

3.3.2 Malware detection with Virustotal module

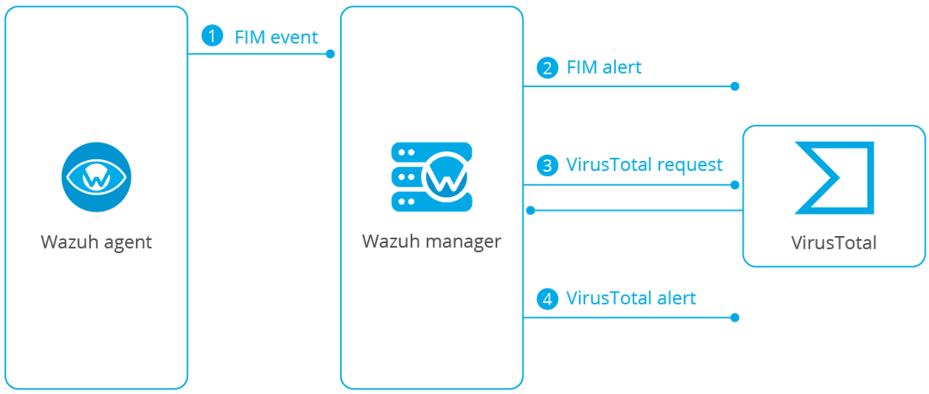


FIGURE 44

Detection

We downloaded some malware into */home* folder. These malwares can not be detected by YARA, because we didn't config enough rules for many malware. But, we can verify them by Virustotal.

However, because of the limitation of the Free API, we will only be able to check 4 files as malware in a minute. (4 lookups/min)

We got the alerts on dashboard.

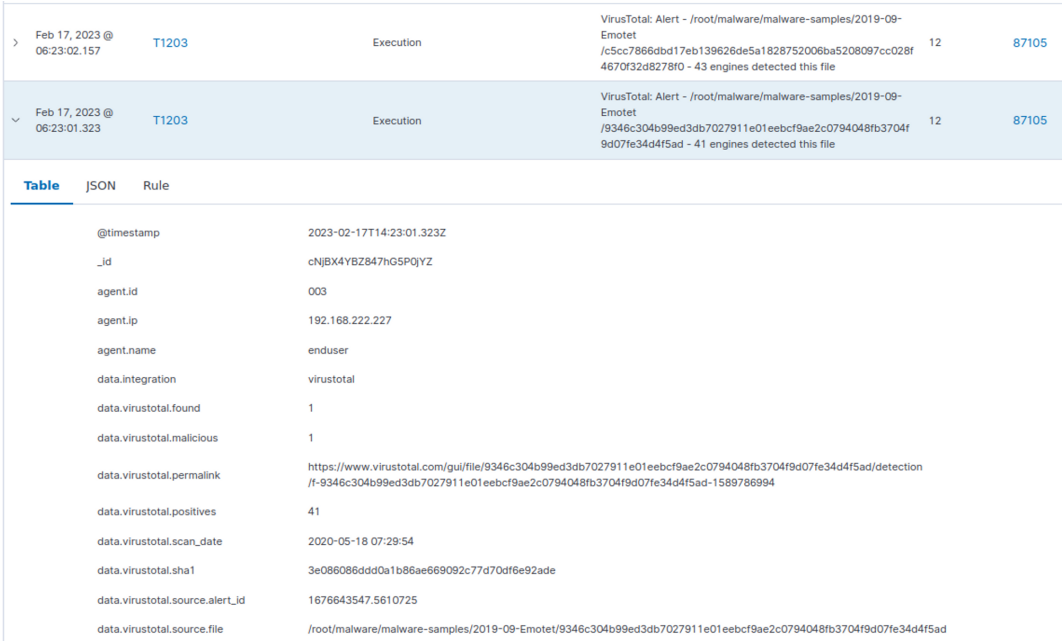


FIGURE 45

Wazuh also prove Virustotal dashboard for analytic.

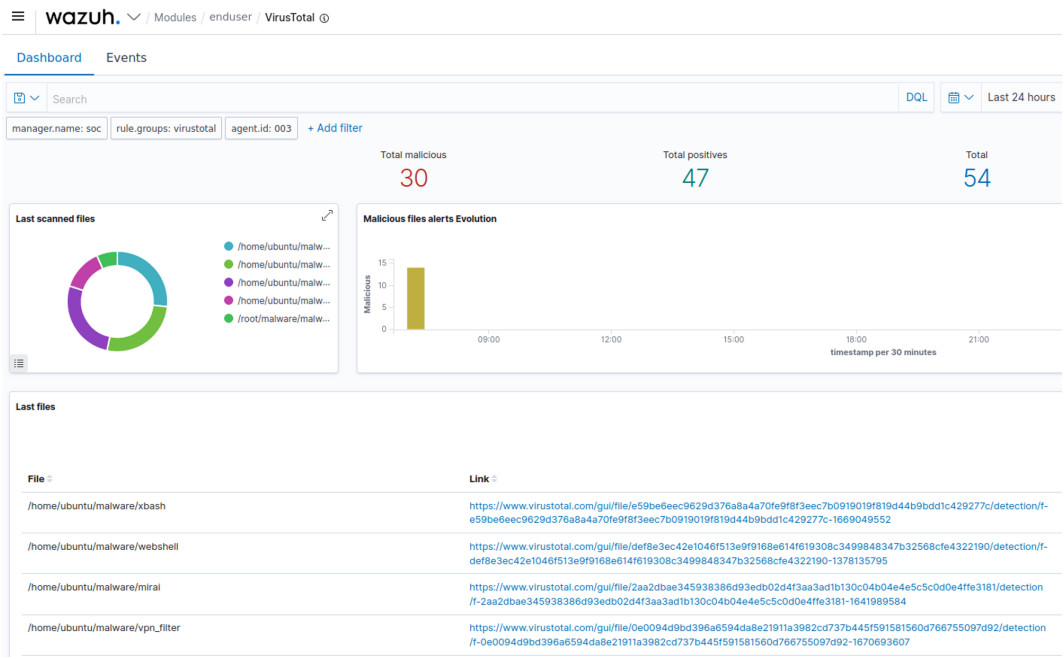


FIGURE 46

Response

The response step is similar to the YARA case, Wazuh receiving the Virustotal warning is positive will trigger the active response. Wazuh-agent will stop running the remove-threat.sh script to remove files that are confirmed to be malicious, then send a warning back to Wazuh manage.

We can see the result in Wazuh dashboard.

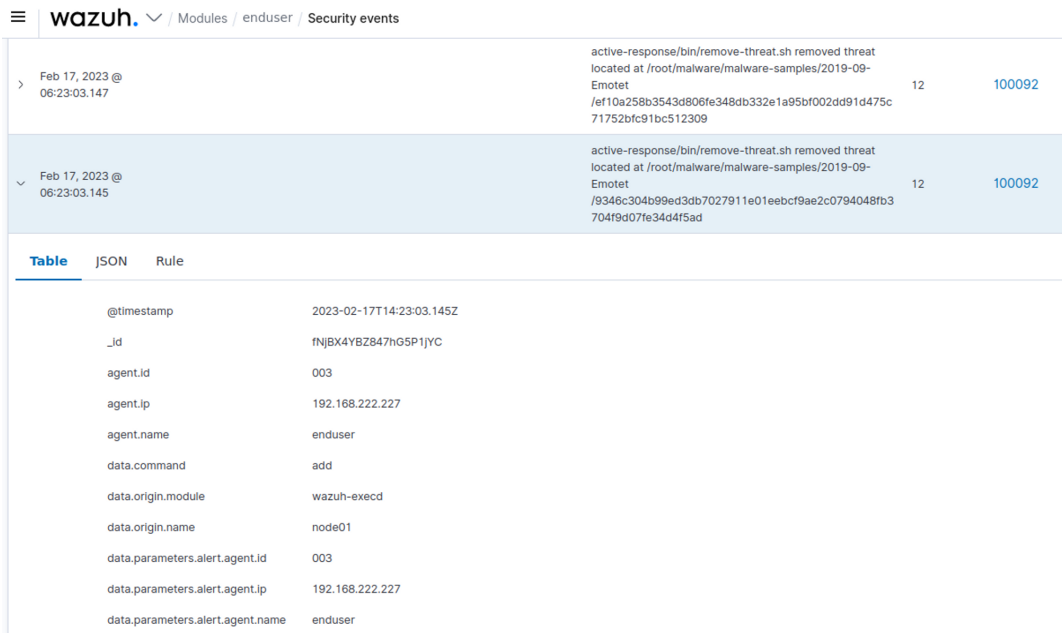


FIGURE 47

More details:

data.parameters.alert.rule.description	VirusTotal: Alert - /root/malware/malware-samples/2019-09-Emotet/9346c304b99ed3db7027911e01eebcf9ae2c0794048fb3704f9d07fe34d4f5ad - 41 engines detected this file
data.parameters.alert.rule.firedtimes	14
data.parameters.alert.rule.gdpr	IV_35.7.d
data.parameters.alert.rule.groups	virustotal
data.parameters.alert.rule.id	87105
data.parameters.alert.rule.level	12
data.parameters.alert.rule.mail	true
data.parameters.alert.rule.mitre.id	T1203
data.parameters.alert.rule.mitre.tactic	Execution
data.parameters.alert.rule.mitre.technique	Exploitation for Client Execution
data.parameters.alert.rule.pci_dss	10.6.1, 11.4
data.parameters.alert.timestamp	2023-02-17T14:23:01.323+0000
data.parameters.extra_args	
data.parameters.program	active-response/bin/remove-threat.sh
data.version	1
decoder.name	ar_log_json
full_log	2023/02/17 14:23:01 active-response/bin/remove-threat.sh: {"version":1,"origin":{"name":"node01","module":"wazuh-execd"},"command":"add","parameters":{"extra_args":[],"alert":{"timestamp":"2023-02-17T14:23:01.323+0000","rule":{"level":12,"description":"VirusTotal: Alert - /root/malware/malware-samples/2019-09-Emotet/9346c304b99ed3db7027911e01eebcf9ae2c0794048fb3704f9d07fe34d4f5ad - 41 engines detected this file","id":"87105","mitre":{"id":["T1203"],"tactic":["Execution"],"technique":["Exploitation for Client Execution"]},"firedtimes":14,"mail":true,"groups":["virustotal"],"pci_dss":["10.6.1","11.4"],"gdpr":["IV_35.7.d"]},"agent":{"id":"003","name":"enduser","ip":"192.168.222.227"},"manager":{"name":"soc"},"id":"1676643781.6198051","decoder":{"name":"json"},"data":{"virustotal":{"found":"1","malicious":"1","source":{"alert_id":"1676643547.5610725","file":"/root/malware/malware-samples/2019-09-Emotet/9346c304b99ed3db7027911e01eebcf9ae2c0794048fb3704f9d07fe34d4f5ad","md5":"2955221dcddc1214dbf94d55a39491eb","sha1":"3e086086ddd0a1b86ae669092c77d70df6e92ade","scan_date":"2020-05-18 07:29:54","positives":"41","total":"61","permalink":"https://www.virustotal.com/gui/file/9346c304b99ed3db7027911e01eebcf9ae2c0794048fb3704f9d07fe34d4f5ad/detection/f-9346c304b99ed3db7027911e01eebcf9ae2c0794048fb3704f9d07fe34d4f5ad-1589786994"},"integration":"virustotal"},"location":"virustotal"},"program":"active-response/bin/remove-threat.sh"}} Successfully removed threat
id	1676643783.6214798
input.type	log
location	/var/ossec/logs/active-responses.log

FIGURE 48

4 Conclusion & Improvement

In summary, we have constructed a simple Security Operation Center. The system is created on a virtualization platform to surveil a web server and user's computer against malicious activities and is capable of generating logs and alerts. In addition, we can also perform quick responses to danger warnings.

Improvement: We encountered some difficulties, for example in monitoring web server files on Docker containers. We want to improve some aspects, such as:

- Configure to monitor details about the containers running on the server, as well as monitor the files on it. This can determine if the webshell is being uploaded to the website.
- Additional SOAR (Security Orchestration, Automation and Response) can be installed, such as TheHive, for better security case management and coordination.

5 Referrence

- https://opendev.org/x/microstack
- https://www.cyber.nj.gov/informational-report/yara-effective-tool-to-detect-malware
- https://yara.readthedocs.io/en/v3.4.0/
- https://www.redpacketsecurity.com/teler-real-time-http-intrusion-detection/
- https://www.funinformatique.com/dvwa-testez-vos-competences-en-test-dintrusion/
- https://www.youtube.com/watch?v=4HJgzNx9BpM
- https://documentation.wazuh.com/current/proof-of-concept-guide/integrate-network-ids-suricata.html
- https://documentation.wazuh.com/current/proof-of-concept-guide/detect-malware-yara-integration.html
- https://documentation.wazuh.com/current/user-manual/capabilities/active-response/ar-use-cases/wazuh-with-yara.html
- https://documentation.wazuh.com/current/user-manual/capabilities/active-response/ar-use-cases/removing-malware.html
- https://kifarunix.com/detecting-malicious-files-with-wazuh-and-virustotal/