2023

Home-Lab
Phase-1 Report

ROHAN GOSWAMI

ABSTRACT

As a Analyst And Security Engineer our main job is to make sure Incident should not occur in first place, So to achieve that we use Different tools like Wazuh, Hive IR, Deceptive Bytes, FortiGate and for user communication we have Slack. So Wazuh is an opensource security platform that provides intrusion detection, log management, and security analytics capabilities. It is designed to help organizations detect, analyze, and respond to security incidents in real-time. This training will provide an overview of the Wazuh architecture, its features, and its integration capabilities with other security tools. Participants will learn how to configure and use Wazuh as a SIEM tool to collect, analyze, and respond to security events. They will also learn how to use Wazuh to monitor and analyze network traffic, identify security threats, and investigate security incidents.

Regards,

Rohan Goswami

Contents

Table of Figure

CHAPTER 1
1.0 INTRODUCTION

Here in this Home Lab Phase-1, we will be see how an organization maintain it's security posture by using different solutions, By using Proactive Approach we will we will do threat hunt will create hypothesis and prove if our hypothesis is right or wrong, by integrating it with Threat-Intelligence we can improve Security Posture,

For that we will,

Plan >> Simulate >> Detect >> Alert Creation >> Resimulate >> Document the Process of Attack & Detect Scenario.

For that, We are using filebeat to send the alerts and data to our Wazuh indexer.

Now Wazuh indexer will indexes that data and using Wazuh dashboard we will able to visualise the data.

Now if any alerts seem to be suspicious, we will have to investigate it further for that we have Hive tool which will help us manage incidents or alerts.

We will have our Hive integrated with Cortex which will help us automate the IOC which given to us.

For user communication & confirmation regarding events done by them or not for that, we have Slack.

Now our Wazuh will be integrated with different data sources like Virus total, AWS, Active-response, CDB Lists, Sysmon, Windows Defender, Hive, Cortex, Guard-duty.

CHAPTER 2
2.0 PROJECT SCOPE

A project scope for a SOC (Security and Operation Centre) analyst typically includes the following elements:

- **Objectives:**
  The main objective and goal of the report, is to improve the **Client's** security posture, monitoring, detecting and responding to cyber threats in a proper manner, and protecting AWS services and sensitive information.

- **Deliverables:**
  The main objective and goal of the report, is to improve the Client's security posture, monitoring, detecting and responding to cyber threats in a proper manner, and protecting AWS services and sensitive information.

- **Budget:**
  The estimated cost of the project, including any expenses related to personal, equipment, and other resources.

- **Resources:**
  The equipment, tools, and other resources that will be required to complete the project, such as security tools and technologies, IR teams.

So overall, it is important that the project scope is very well and clearly defined, understood and agreed upon by all stakeholder. This will ensure that the project stays up-to-date, is completed on time and within budget, and meets the organization's requirement.

CHAPTER 3
3.0 HARDWARE & SOFTWARE REQUIREMENT

3.1 Hardware & Software requirements for Wazuh:

Minimum Hardware Requirement for Wazuh:

Processor	2.0 GHz
RAM	8 GB
HDD	50 GB

Table 3.1.1 Hardware Requirements for Wazuh

Minimum Software Requirement for Wazuh:

Operating System	Here for demo purpose, I have taken: • 4 Windows OS(Agent) • 1 RedHat Linux (Agent) • 2 Ubuntu OS (Manager (Master-Worker))
Programming Language	-
Other Tools & Tech	Sysmon installed, Event viewer, Virtual Box or VMware.

Table 3.1.2 Hardware Requirements for Wazuh

3.2 Hardware & Software requirements for Hive, Cortex & MISP:

Hardware requirements for Hive & Cortex:

Processor	4 GHz
RAM	16 GB
HDD	160 GB

Table 3.2.1 Hardware Requirements for Hive & Cortex

Software requirements for Hive & Cortex:

Operating System	2-Ubuntu OS
Programming Language	-
Other Tools & Tech	Elastic version 7.11.2 Virtual box or Other than this no preq needed

Table 3.2.2 Software Requirements for Hive & Cortex

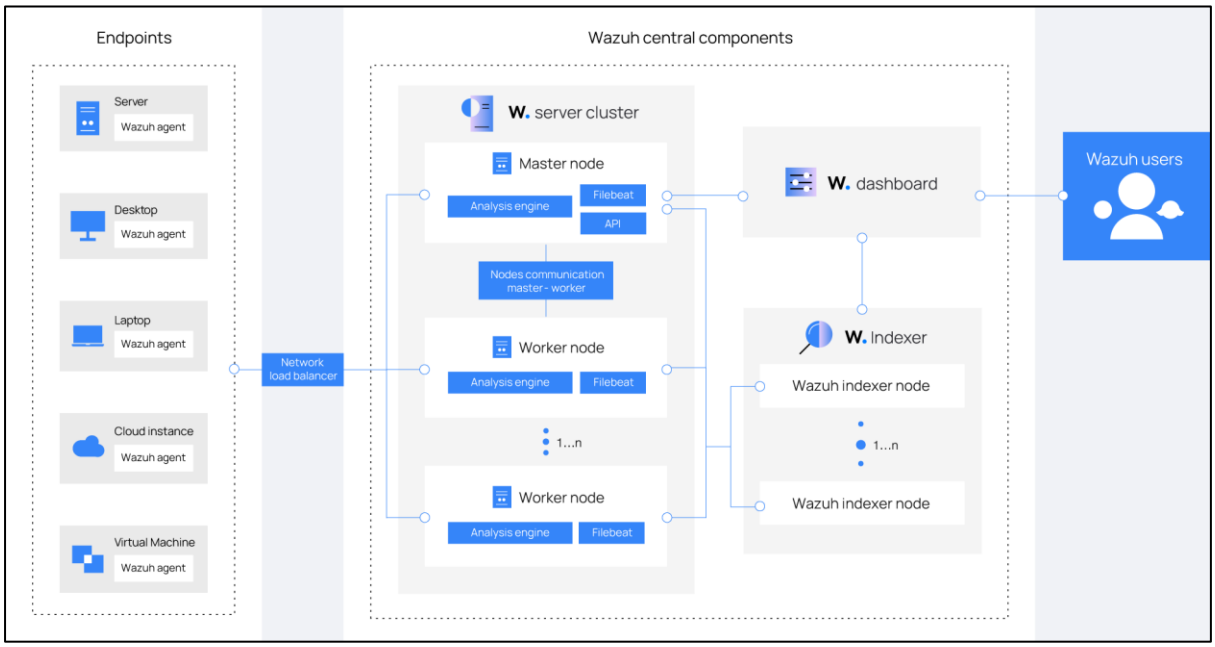
CHAPTER 4
4.0 PROJECT PLAN

List of Tasks:

- Blocking Malicious IP Trying to Login (Blocking SSH Attack).
- Detect Blacklisted IP login using CDB List.
- Installation of TheHive and Cortex (Integration of Hive and Cortex).
- Integration of TheHive with Wazuh (SIEM).
- Integration of Cortex with Different Threat Intel (Virus total, AbuseIPDB, IP void, IBMxForce).
- Integration of Wazuh with Virus total to Detect Malware.
- Integration of AWS with Wazuh.
- Monitoring Different AWS Services (EC2, S3, CT, IAM, Console Login, Root Login Detection).

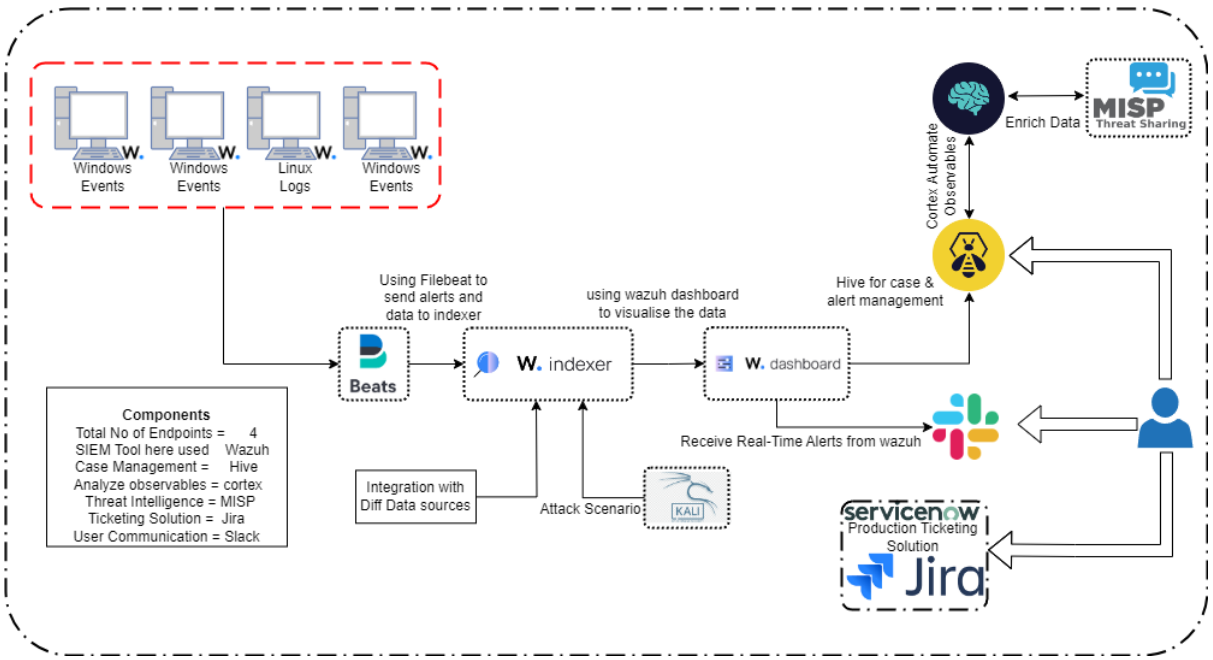
CHAPTER 5
5.0 PROCESS MODEL

5.1 Wazuh Architecture:



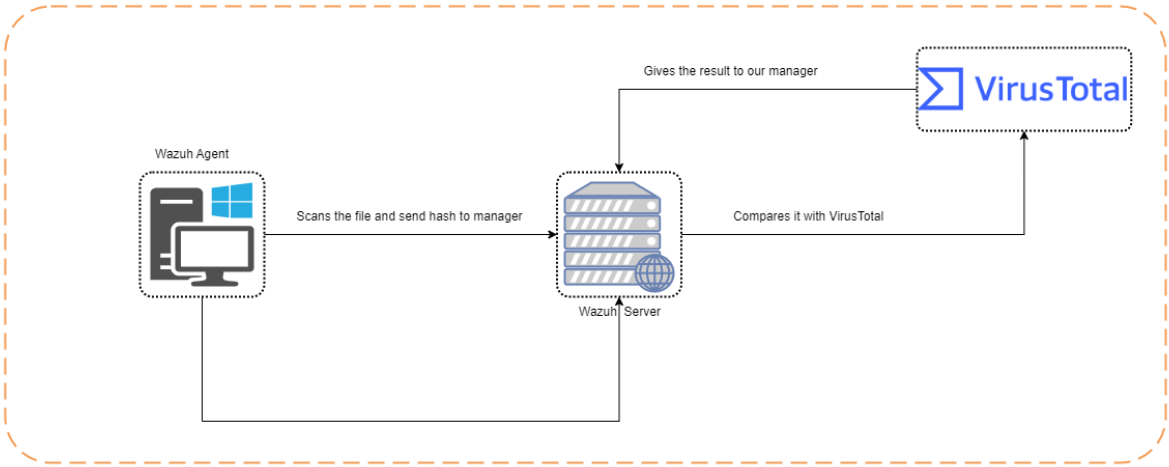
(Figure 1 wazuh architecture)

5.2 Project Architecture:



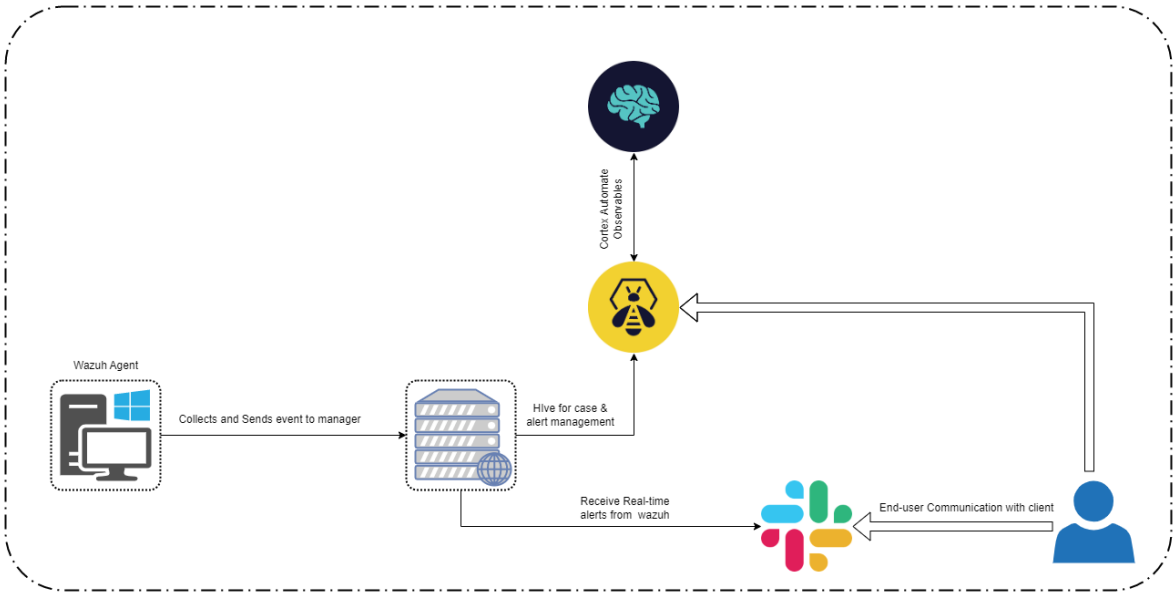
(Figure 2 Architecture of project)

5.3 Integration of Wazuh with Virus Total:



(Figure 3 Integration of Wazuh with Virus Total)

5.4 Integration of Wazuh with Hive, Cortex and Slack:



(Figure 4 Integration of Wazuh with Hive)

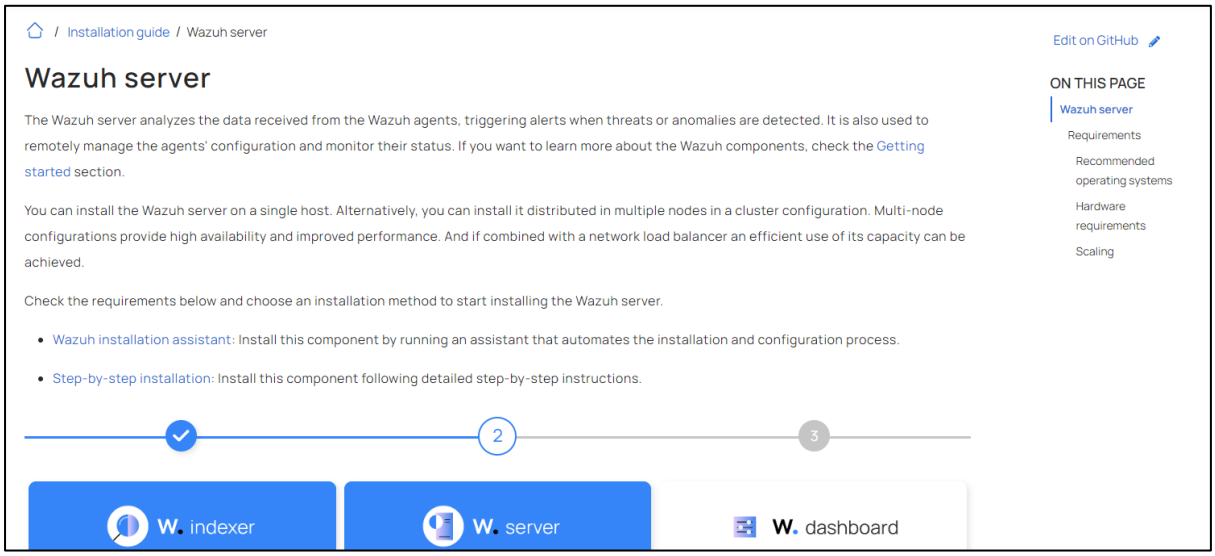
Chapter 6
6.0 IMPLEMENTATION DETAILS

Now let's show the step-by-step installation of Wazuh and all other stuff.

6.1 Wazuh installation

Download the Wazuh iso file from the official website

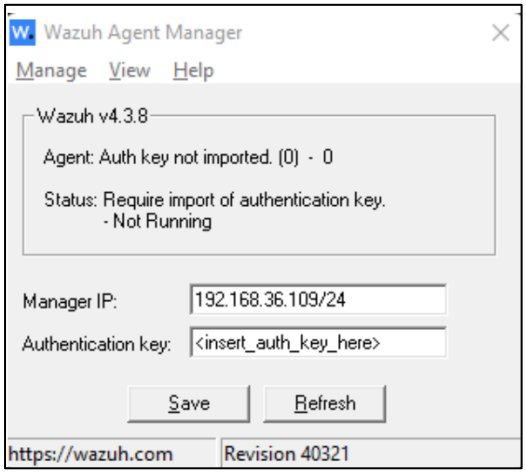
Website: https://documentation.wazuh.com/current/installation-guide/wazuh-server/index.html



(Figure 5 Wazuh installation)

Now after installing & Deploying Wazuh Server

2. Setup the Wazuh in local environment First we have to get the IP address of the Wazuh OS



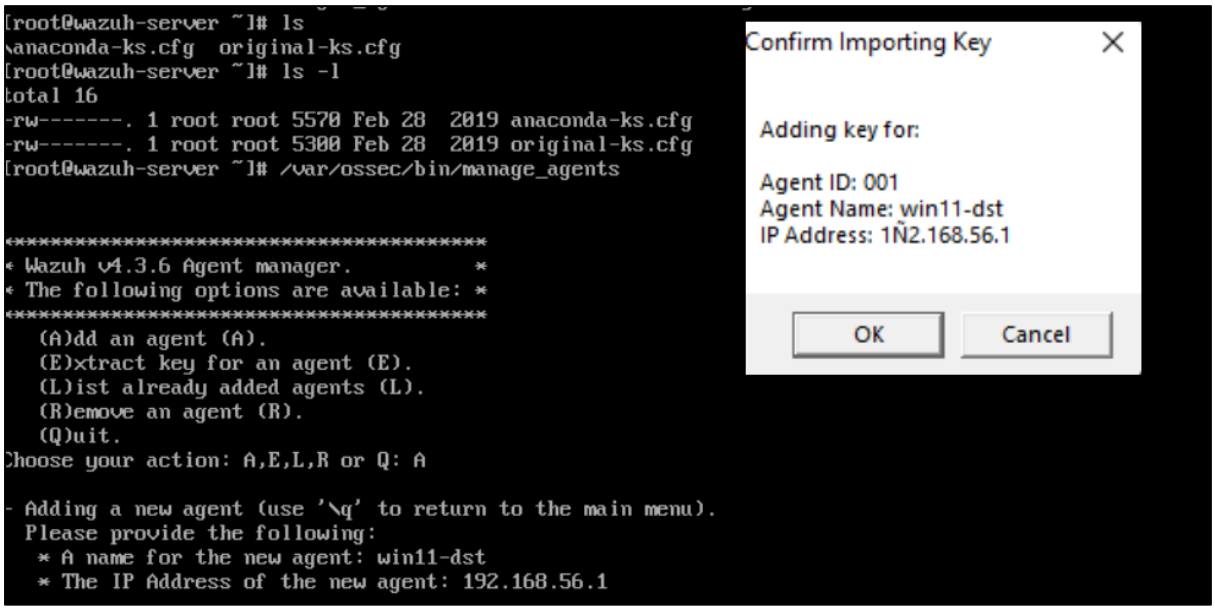
(Figure 6 Authentication key configuration)

Now we have to generate the auth key and insert it for register the user.

To generate auth key, we need to add the agent in our Wazuh-manager and after that extract key for that particular agent.

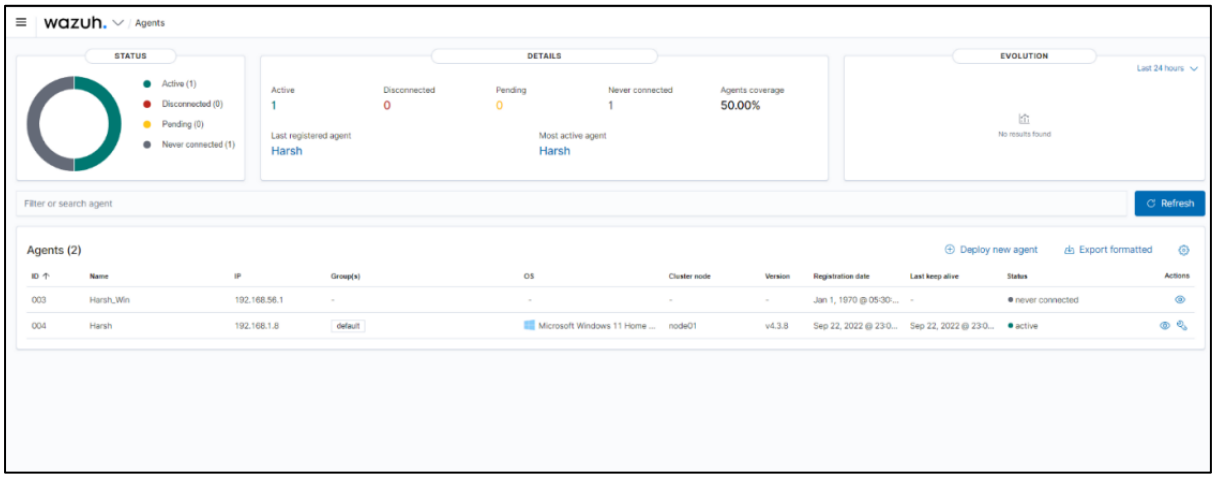
(Figure 7 extracting authentication key)

Now you can see the Agent key given to us, now what we have to do is to import that key in our agent file after adding key into our agent file, you can restart the agent service



(Figure 8 importing authentication key)

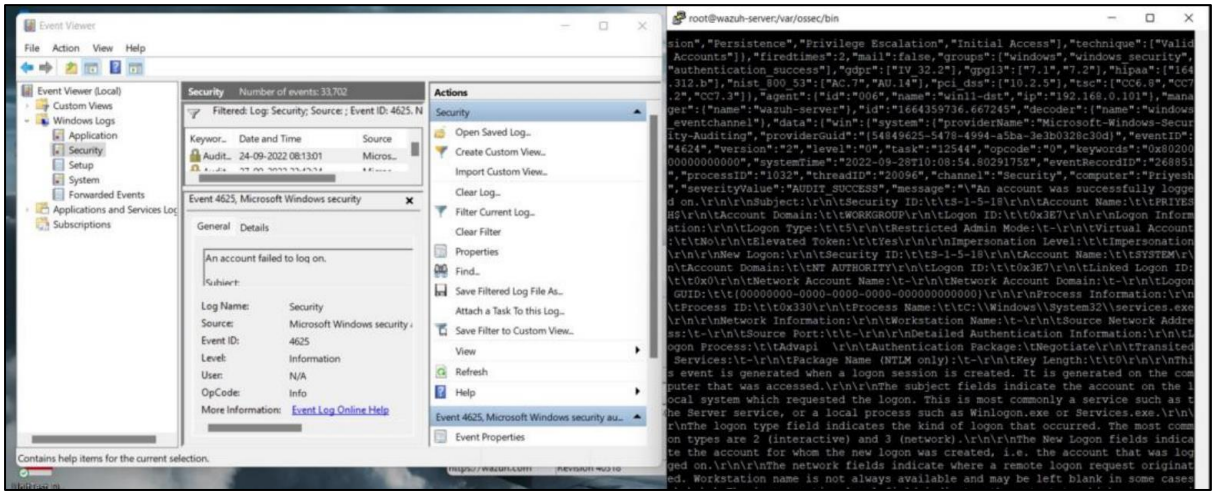
Once you successfully restarted the agent, you can see that the agent is loaded and begin active in our system as shown in figure.



(Figure 9 agent status)

Now let's see what happen if we enter wrong or right password for authentication,

So, from one side I will enter wrong password for remote login and we will see what happens.



(Figure 10 Authentication raw logs)

Here you can see in raw logs that someone was trying to login remotely but it gets unsuccessful.

6.2 Blocking SSH brute-force attack with active response

Step 1: Open the Wazuh server **/var/ossec/etc/ossec.conf** file and verify that a **<command>** block called firewall-drop.

```
<command>
  <name>disable-account</name>
  <executable>disable-account</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>restart-wazuh</name>
  <executable>restart-wazuh</executable>
</command>

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


<command>
  <name>host-deny</name>
  <executable>host-deny</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<command>
  <name>route-null</name>
  <executable>route-null</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

(Figure 11 Verifying firewall-drop command)

Step 2: Add the **<active-response>** block below to the Wazuh server **/var/ossec/etc/ossec.conf** configuration file:

```
<command>
  <name>netsh</name>
  <executable>netsh.exe</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>

<active-response>
<command>firewall-drop</command>
<location>local</location>
<rules_id>200009</rules_id>
<timeout>60</timeout>
</active-response>
```

(Figure 12 Adding active response block in conf)

Step 3: Restart the Wazuh manager service to apply the changes:

Command:

sudo systemctl restart Wazuh-manager

Step 4: Test the configuration,

Perform the steps below to perform an SSH brute-force attack,

Ping the RHEL endpoint to confirm there is network connectivity between the attacker and the victim endpoints:

Ping 192.168.0.102

```
C:\Users\Rohan>ping 192.168.0.102

Pinging 192.168.0.102 with 32 bytes of data:
Reply from 192.168.0.102: bytes=32 time<1ms TTL=64
Reply from 192.168.0.102: bytes=32 time<1ms TTL=64
Reply from 192.168.0.102: bytes=32 time<1ms TTL=64
Reply from 192.168.0.102: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.0.102:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

(Figure 13 checking victim machine status)

So now you can see in our Wazuh that we have detected Particular IP Address is trying to Login into your system.

	Time	rule.description	rule.level	rule.id
›	Apr 3, 2023 @ 09:08:53.768	Host Unblocked by firewall-drop Active Response	3	652
›	Apr 3, 2023 @ 09:07:51.714	Host Blocked by firewall-drop Active Response	3	651
›	Apr 3, 2023 @ 09:07:51.699	sshd: brute force attempt from	10	200009
›	Apr 3, 2023 @ 09:07:51.659	unix_chkpwd: Password check failed.	5	5557
›	Apr 3, 2023 @ 09:07:47.634	sshd: authentication failed.	5	5760
›	Apr 3, 2023 @ 09:07:45.630	unix_chkpwd: Password check failed.	5	5557
›	Apr 3, 2023 @ 09:07:43.628	sshd: authentication failed.	5	5760
›	Apr 3, 2023 @ 09:07:41.668	PAM: User login failed.	5	5503
›	Apr 3, 2023 @ 09:07:41.625	unix_chkpwd: Password check failed.	5	5557
›	Apr 3, 2023 @ 09:06:47.537	sshd: Timeout while logging in.	4	5704
›	Apr 3, 2023 @ 09:06:03.445	Host Unblocked by firewall-drop Active Response	3	652
›	Apr 3, 2023 @ 09:05:01.409	Host Blocked by firewall-drop Active Response	3	651
›	Apr 3, 2023 @ 09:05:01.364	sshd: brute force attempt from	10	200009
›	Apr 3, 2023 @ 09:04:59.362	unix_chkpwd: Password check failed.	5	5557
›	Apr 3, 2023 @ 09:04:55.357	sshd: authentication failed.	5	5760
›	Apr 3, 2023 @ 09:04:53.357	unix_chkpwd: Password check failed.	5	5557
›	Apr 3, 2023 @ 09:04:53.355	sshd: authentication failed.	5	5760

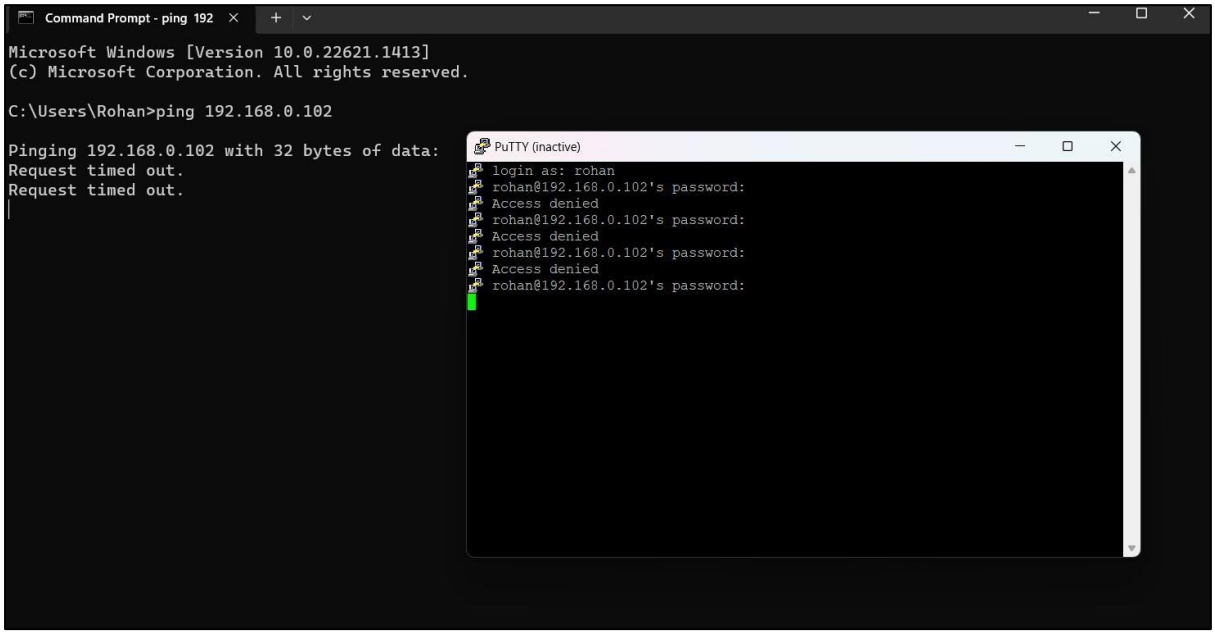
(Figure 14 Wazuh Events for Active-Response)

Now with the Help of active-response we will block this Particular IP who was trying to Login into our system,

	Time	rule.description	rule.level	rule.id
›	Apr 3, 2023 @ 09:08:53.768	Host Unblocked by firewall-drop Active Response	3	652
›	Apr 3, 2023 @ 09:07:51.714	Host Blocked by firewall-drop Active Response	3	651
›	Apr 3, 2023 @ 09:07:51.699	sshd: brute force attempt from	10	200009
›	Apr 3, 2023 @ 09:07:51.659	unix_chkpwd: Password check failed.	5	5557

(Figure 15 Active-Response Events)

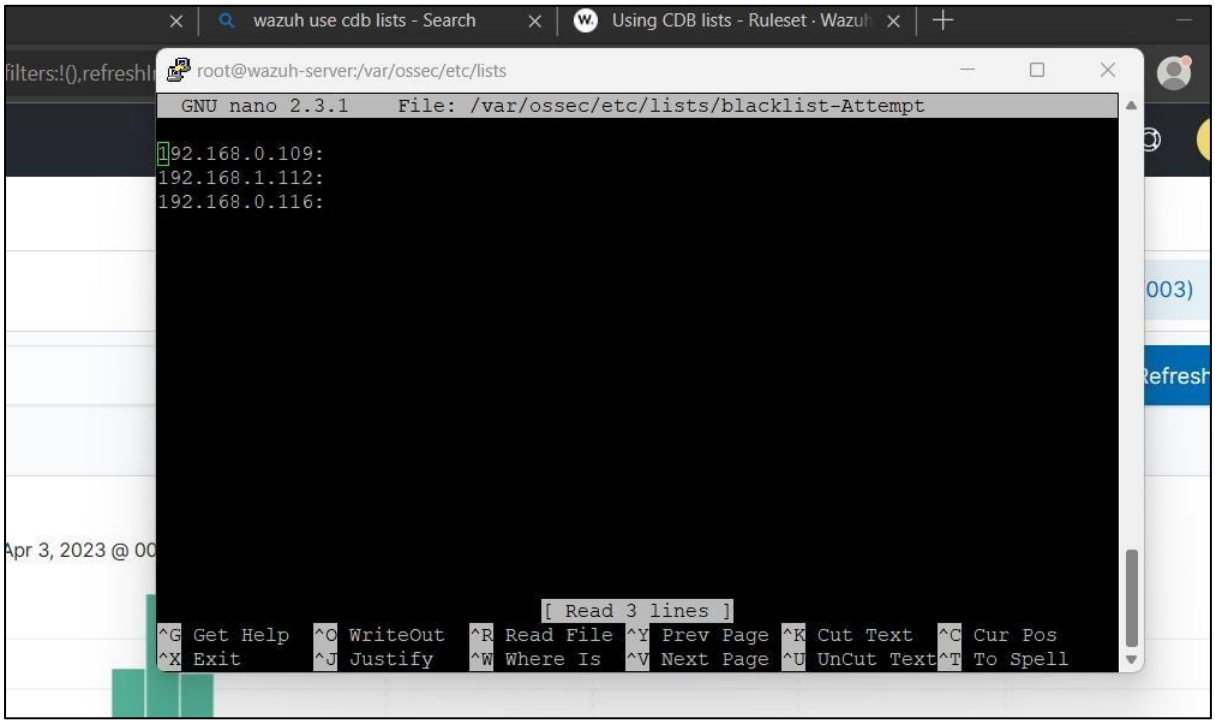
If you see here if it tries to get Login again it will block the IP which was trying to login.



(Figure 16 Blocking Attacker's IP)

6.3 Detect if any Blacklisted IP trying to Login using CDB Lists.

To Detect if any Blacklist IP tries to login, it will detect that give us alert in our Wazuh.



(Figure 17 Adding Blacklisted IP to a List)

Here we have some IPs in our CDBlist, which will help us to create a rule by using CDBlist.

```
[root@wazuh-server lists]# nano /var/ossec/etc/lists/blacklist-Attempt
[root@wazuh-server lists]# ls -llh
total 28K
drwxrwx---. 2 wazuh wazuh   73 Nov 11 14:22 amazon
-rw-rw----. 1 wazuh wazuh  107 Nov 11 13:11 audit-keys
-rw-rw----. 1 wazuh wazuh 2.3K Nov 11 14:22 audit-keys.cdb
-rw-r--r--. 1 wazuh wazuh   45 Apr  3 16:38 blacklist-Attempt
-rw-rw----. 1 wazuh wazuh 2.2K Apr  3 16:40 blacklist-Attempt.cdb
-rw-rw----. 1 wazuh wazuh  892 Nov 11 13:11 security-eventchannel
-rw-rw----. 1 wazuh wazuh 6.4K Nov 11 14:22 security-eventchannel.cdb
[root@wazuh-server lists]# chown wazuh:wazuh blacklist-Attempt
[root@wazuh-server lists]#
```

(Figure 18 Giving ownership to Wazuh)

We have to give this. cdb ownership to our Wazuh.

```
<ruleset>
  <!-- Default ruleset -->
  <decoder_dir>ruleset/decoders</decoder_dir>
  <rule_dir>ruleset/rules</rule_dir>
  <rule_exclude>0215-policy_rules.xml</rule_exclude>
  <list>etc/lists/audit-keys</list>
  <list>etc/lists/amazon/aws-eventnames</list>
  <list>etc/lists/blacklist-Attempt</list>
  <list>etc/lists/security-eventchannel</list>
```

(Figure 19 Adding it to a conf ruleset)

Here to display this alert in our SIEM tool we need to give this file's path to our configuration file.

```
<rule id="200009" level="10" frequency="3" timeframe="120" ignore="60">
  <if_matched_sid>5760</if_matched_sid>
  <description>sshd: brute force attempt from $(data.srcip)</description>
  <mitre>
    <id>T1110</id>
  </mitre>
  <group>authentication_failures,gdpr_IV_35.7.d,gdpr_IV_32.2,hipaa_164.312.b,nist_800_53_SI.4,nist_800_53_AU.14,nist_800_53_AC.7,pci_dss_11.4,pci_dss_10.2.4,pci_dss_10.2.5
      ,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
</rule>
<rule id="200010" level="12">
    <if_sid>5715</if_sid>
    <list field="srcip" lookup="address_match_key">etc/lists/blacklist-Attempt</list>
    <description>BlackListed IP Attempted to Login</description>
    <options>no_full_log</options>
</rule>
```

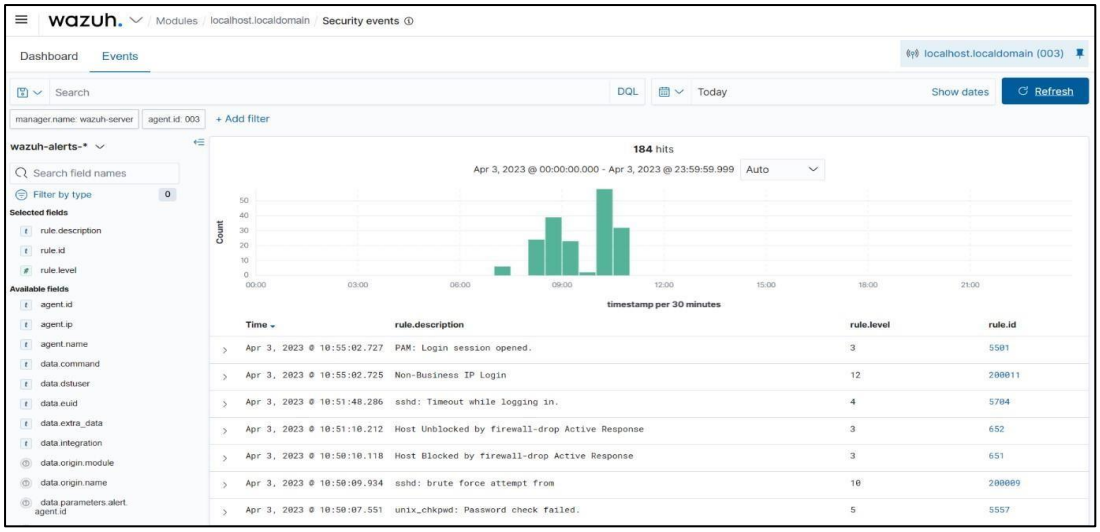
(Figure 20 creating a rule to generate an event)

Then after that we can create a rule which will trigger if any of the IP listed will try to login into SSH then it will generate an alert.

```
<rule id="200009" level="10" frequency="3" timeframe="120" ignore="60">
  <if_matched_sid>5760</if_matched_sid>
  <description>sshd: brute force attempt from $(data.srcip)</description>
  <mitre>
    <id>T1110</id>
  </mitre>
  <group>authentication_failures,gdpr_IV_35.7.d,gdpr_IV_32.2,hipaa_164.312.b,nist_800_53_SI.4,nist_800_53_AU.14,nist_800_53_AC.7,pci_dss_11.4,pci_dss_10.2.4,pci_dss_10.2.5
      ,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
</rule>
<rule id="200010" level="12">
    <if_sid>5715</if_sid>
    <list field="srcip" lookup="address_match_key">etc/lists/blacklist-Attempt</list>
    <description>BlackListed IP Attempted to Login</description>
    <options>no_full_log</options>
</rule>
<rule id="200011" level="12">
    <if_sid>200009</if_sid>
    <list field="srcip" lookup="address_match_key">etc/lists/blacklist-Attempt</list>
    <description>Multiple Failed Login Attempt from blacklisted IP</description>
    <options>no_full_log</options>
</rule>
```

(Figure 21 created another rule to trigger an event)

Now after creating a rule, now we can detect this event in our Wazuh.



(Figure 22 Event Generated)

6.4 Installation of The Hive in Ubuntu Box:

This installs the OpenJDK 8 runtime environment without any user prompts.
It allows any commands or scripts executed in the current shell session to use the correct Java installation.

```
apt-get install -y openjdk-8-jre-headless
```

```
root@rohan:/home/rohan# apt-get install -y openjdk-8-jre-headless
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java java-common
Suggested packages:
  default-jre fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei
The following NEW packages will be installed:
  ca-certificates-java java-common openjdk-8-jre-headless
0 upgraded, 3 newly installed, 0 to remove and 86 not upgraded.
Need to get 30.8 MB of archives.
After this operation, 104 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 java-common all 0.72build2 [6,782 B]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 openjdk-8-jre-headless amd64 8u362-ga-0ubuntu1~22.04 [30.8 MB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 ca-certificates-java all 20190909ubuntu1.1 [12.0 kB]
Fetched 30.8 MB in 12s (2,571 kB/s)
Selecting previously unselected package java-common.
(Reading database ... 201343 files and directories currently installed.)
Preparing to unpack .../java-common_0.72build2_all.deb ...
Unpacking java-common (0.72build2) ...
Selecting previously unselected package openjdk-8-jre-headless:amd64.
Preparing to unpack .../openjdk-8-jre-headless_8u362-ga-0ubuntu1~22.04_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u362-ga-0ubuntu1~22.04) ...
Selecting previously unselected package ca-certificates-java.
Preparing to unpack .../ca-certificates-java_20190909ubuntu1.1_all.deb ...
Unpacking ca-certificates-java (20190909ubuntu1.1) ...
Setting up java-common (0.72build2) ...
Setting up openjdk-8-jre-headless:amd64 (8u362-ga-0ubuntu1~22.04) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java to provide /usr/bin/java (java) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/jjs to provide /usr/bin/jjs (jjs) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/keytool to provide /usr/bin/keytool (keytool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/pack200 to provide /usr/bin/pack200 (pack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/rmid to provide /usr/bin/rmid (rmid) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/unpack200 to provide /usr/bin/unpack200 (unpack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/lib/jexec to provide /usr/bin/jexec (jexec) in auto mode
Setting up ca-certificates-java (20190909ubuntu1.1) ...
head: cannot open '/etc/ssl/certs/java/cacerts' for reading: No such file or directory
Adding debian:Hellenic_Academic_and_Research_Institutions_ECC_RootCA_2015.pem
Adding debian:ePKI_Root_Certification_Authority.pem
Adding debian:emSign_ECC_Root_CA_-_G3.pem
Adding debian:CA_Disig_Root_R2.pem
Adding debian:GlobalSign_Root_CA_-_R3.pem
Adding debian:Hellenic_Academic_and_Research_Institutions_RootCA_2015.pem
Adding debian:NAVER_Global_Root_Certification_Authority.pem
Adding debian:Microsec_e-Szigno_Root_CA_2009.pem
Adding debian:Certum_Trusted_Network_CA_2.pem
Adding debian:Baltimore_CyberTrust_Root.pem
Adding debian:emSign_ECC_Root_CA_-_C3.pem
Adding debian:Buypass_Class_3_Root_CA.pem
Adding debian:QuoVadis_Root_CA_1_G3.pem
```

(Figure 23 Installing OpenJDK Env)

It allows any commands or scripts executed in the current shell session to use the correct Java installation.

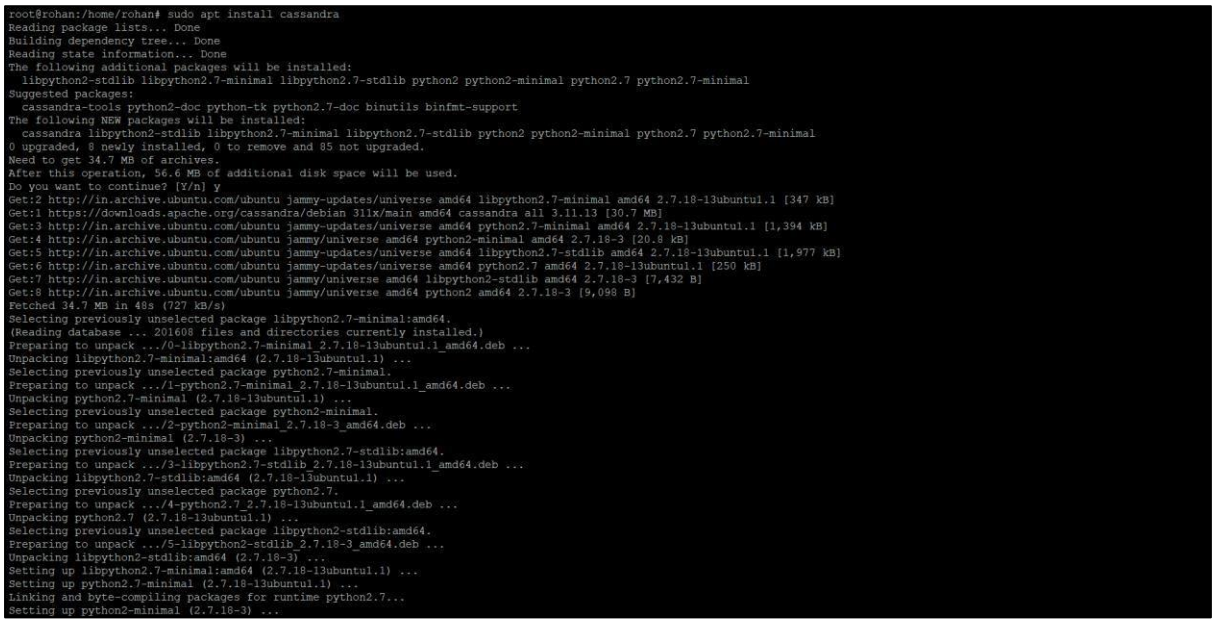
```
echo JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64" >> /etc/environment
export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
```

```
done.
done.
root@rohan:/home/rohan# echo JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64" >> /etc/environment
root@rohan:/home/rohan# export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
root@rohan:/home/rohan# curl -fsSL https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
Command 'curl' not found, but can be installed with:
apt install curl
gpg: no valid OpenPGP data found.
root@rohan:/home/rohan# apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl
The following packages will be upgraded:
  libcurl4
1 upgraded, 1 newly installed, 0 to remove and 85 not upgraded.
Need to get 194 kB/484 kB of archives.
After this operation, 459 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.10 [194 kB]
Fetched 194 kB in 2s (89.2 kB/s)
(Reading database ... 201601 files and directories currently installed.)
Preparing to unpack .../libcurl4_7.81.0-1ubuntu1.10_amd64.deb ...
Unpacking libcurl4:amd64 (7.81.0-1ubuntu1.10) over (7.81.0-1ubuntu1.7) ...
Selecting previously unselected package curl.
Preparing to unpack .../curl_7.81.0-1ubuntu1.10_amd64.deb ...
Unpacking curl (7.81.0-1ubuntu1.10) ...
Setting up libcurl4:amd64 (7.81.0-1ubuntu1.10) ...
Setting up curl (7.81.0-1ubuntu1.10) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
root@rohan:/home/rohan# curl -fsSL https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

(Figure 24 Installing Packages)

This command installs the latest version of Cassandra available from the configured repositories.

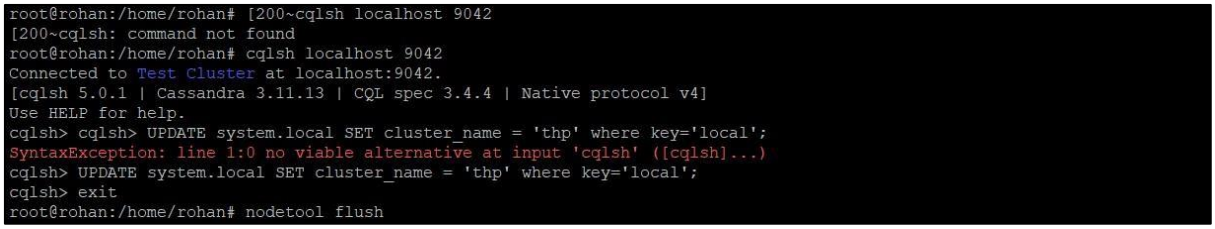
```
sudo apt update sudo apt
install Cassandra
```



(Figure 25 Installing Cassandra)

Change the cluster name as thp. Then run the command cqlsh:

```
cqlsh localhost 9042
cqlsh> UPDATE system.local SET cluster_name = 'thp' where key='local';
nodetool flush
```



(Figure 26 Change cluster name)

Add content to the yaml file of cassandra:

#Content from yaml:

```
cluster_name: 'thp' listen_address: '192.168.0.116' # address for nodes rpc_address: '192.168.0.116' # address for clients seed_provider:
class_name: org.apache.cassandra.locator.SimpleSeedProvider       parameters:
          # Ex: "<ip1>,<ip2>,<ip3>"
seeds: '192.168.0.116' # self for the first node data_file_directories:   -
'/var/lib/cassandra/data'  commitlog_directory: '/var/lib/cassandra/commitlog'
saved_caches_directory: '/var/lib/cassandra/saved_caches' hints_directory:   - '/var/lib/cassandra/hints'
```

```
root@rohan:/home/rohan# nodetool flush
root@rohan:/home/rohan# /etc/cassandra/cassandra.yaml
bash: /etc/cassandra/cassandra.yaml: Permission denied
root@rohan:/home/rohan# sudo /etc/cassandra/cassandra.yaml
sudo: /etc/cassandra/cassandra.yaml: command not found
root@rohan:/home/rohan# nano /etc/cassandra/cassandra.yaml
root@rohan:/home/rohan# systemctl daemon-reload
root@rohan:/home/rohan# service cassandra restart
root@rohan:/home/rohan# mkdir /opt/thp/thehive/index
mkdir: cannot create directory '/opt/thp/thehive/index': No such file or directory
root@rohan:/home/rohan# mkdir -p /opt/thp/thehive/files
root@rohan:/home/rohan# echo 'deb https://deb.thehive-project.org release main' | sudo tee -a /etc/apt/sources.l
ist.d/thehive-project.list
deb https://deb.thehive-project.org release main
root@rohan:/home/rohan# sudo apt-get update
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease
Get:5 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:6 https://deb.thehive-project.org release InRelease [2,582 B]
Err:6 https://deb.thehive-project.org release InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 3D99BB18562CB
C1C
Hit:1 https://downloads.apache.org/cassandra/debian 311x InRelease
Reading package lists... Done
W: GPG error: https://deb.thehive-project.org release InRelease: The following signatures couldn't be verified b
ecause the public key is not available: NO_PUBKEY 3D99BB18562CBC1C
E: The repository 'https://deb.thehive-project.org release InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
W: http://www.apache.org/dist/cassandra/debian/dists/311x/InRelease: Key is stored in legacy trusted.gpg keyring
 (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
root@rohan:/home/rohan# sudo apt-get install thehive4
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package thehive4
root@rohan:/home/rohan# apt-get install thehive4
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package thehive4
root@rohan:/home/rohan# curl https://raw.githubusercontent.com/TheHive-Project/TheHive/master/PGP-PUBLIC-KEY | s
udo apt-key add -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0Warning: apt-key is deprecated. Ma
nage keyring files in trusted.gpg.d instead (see apt-key(8)).
100  3136  100  3136    0     0   6411      0 --:--:-- --:--:-- --:--:--  6400
OK
```

(Figure 27 Adding contexts to Cassandra file)

Now install thehive in your System:

```
root@rohan:/home/rohan# sudo apt-get install thehive4
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  thehive4
0 upgraded, 1 newly installed, 0 to remove and 85 not upgraded.
Need to get 202 MB of archives.
After this operation, 202 MB of additional disk space will be used.
Get:1 https://deb.thehive-project.org release/main amd64 thehive4 all 4.1.24-1 [202 MB]
Fetched 202 MB in 5min 24s (623 kB/s)
Selecting previously unselected package thehive4.
(Reading database ... 202512 files and directories currently installed.)
Preparing to unpack .../thehive4_4.1.24-1_all.deb ...
Unpacking thehive4 (4.1.24-1) ...
Setting up thehive4 (4.1.24-1) ...
Creating system group: thehive
Creating system user: thehive in thehive with thehive daemon-user and shell /bin/false
1+0 records in
1+0 records out
1024 bytes (1.0 kB, 1.0 KiB) copied, 4.6831e-05 s, 21.9 MB/s
root@rohan:/home/rohan# mkdir -p /opt/thp/thehive/files
root@rohan:/home/rohan# chown -R thehive:thehive /opt/thp/thehive/files
root@rohan:/home/rohan# mkdir /opt/thp/thehive/index
mkdir: cannot create directory '/opt/thp/thehive/index': File exists
root@rohan:/home/rohan# cd /opt/thp/thehive/index
```

(Figure 28 Adding File system and installing hive)

TheHive configuration file **(application.conf)** must be modified and updated with the following lines in order to use the Cassandra database:

```
db {
  provider: janusgraph
  janusgraph {
    storage {
      backend: cql
      hostname: ["Ubuntu's IP"] # seed node ip addresses
      #username: "<cassandra_username>"       # login to connect to database (if configured in Cassandra)
      #password: "<cassandra_passowrd"
      cql {
        cluster-name: thp       # cluster name
        keyspace: thehive           # name of the keyspace
        local-datacenter: datacenter1   # name of the datacenter where TheHive runs (relevant only on multi datacenter setup)
        # replication-factor: 2 # number of replica
        read-consistency-level: ONE
        write-consistency-level: ONE
      }
    }
  }
}
```

(Figure 29 Adding Following content to Conf file)

Filesystem:

Verify the folder's permissions:

```
chown -R thehive:thehive /opt/thp/thehive/files
```

These lines should be added to the application.conf file of Hive:

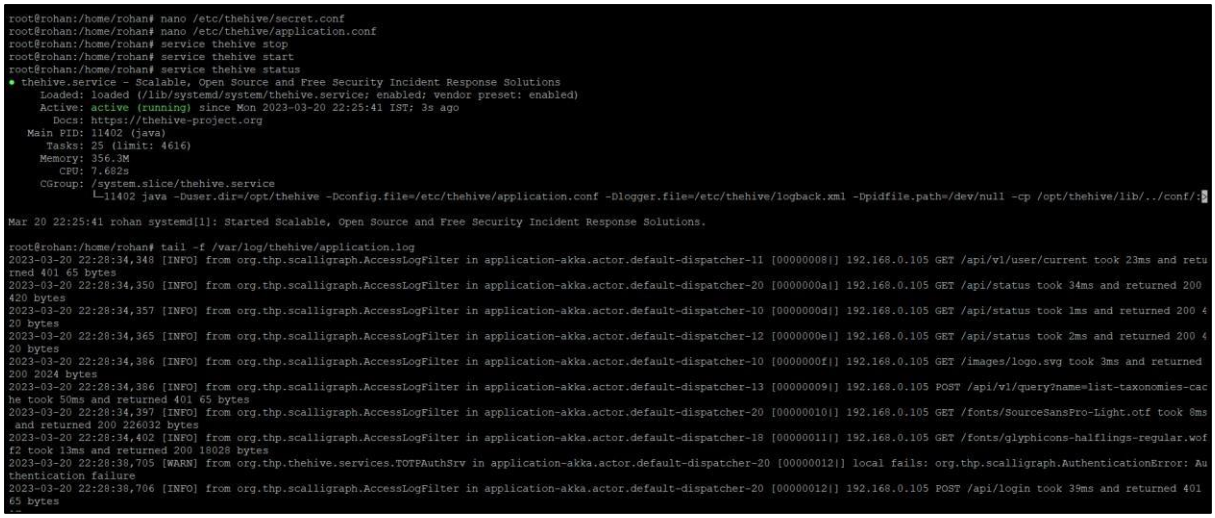
```
## Storage configuration
storage {
provider = localfs
localfs.location = /opt/thp/thehive/files
}
```

Now install thehive in your System:

```
root@rohan:/home/rohan# sudo apt-get install thehive4
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  thehive4
0 upgraded, 1 newly installed, 0 to remove and 85 not upgraded.
Need to get 202 MB of archives.
After this operation, 202 MB of additional disk space will be used.
Get:1 https://deb.thehive-project.org release/main amd64 thehive4 all 4.1.24-1 [202 MB]
Fetched 202 MB in 5min 24s (623 kB/s)
Selecting previously unselected package thehive4.
(Reading database ... 202512 files and directories currently installed.)
Preparing to unpack .../thehive4_4.1.24-1_all.deb ...
Unpacking thehive4 (4.1.24-1) ...
Setting up thehive4 (4.1.24-1) ...
Creating system group: thehive
Creating system user: thehive in thehive with thehive daemon-user and shell /bin/false
1+0 records in
1+0 records out
1024 bytes (1.0 kB, 1.0 KiB) copied, 4.6831e-05 s, 21.9 MB/s
root@rohan:/home/rohan# mkdir -p /opt/thp/thehive/files
root@rohan:/home/rohan# chown -R thehive:thehive /opt/thp/thehive/files
root@rohan:/home/rohan# mkdir /opt/thp/thehive/index
mkdir: cannot create directory '/opt/thp/thehive/index': File exists
root@rohan:/home/rohan# cd /opt/thp/thehive/index
```

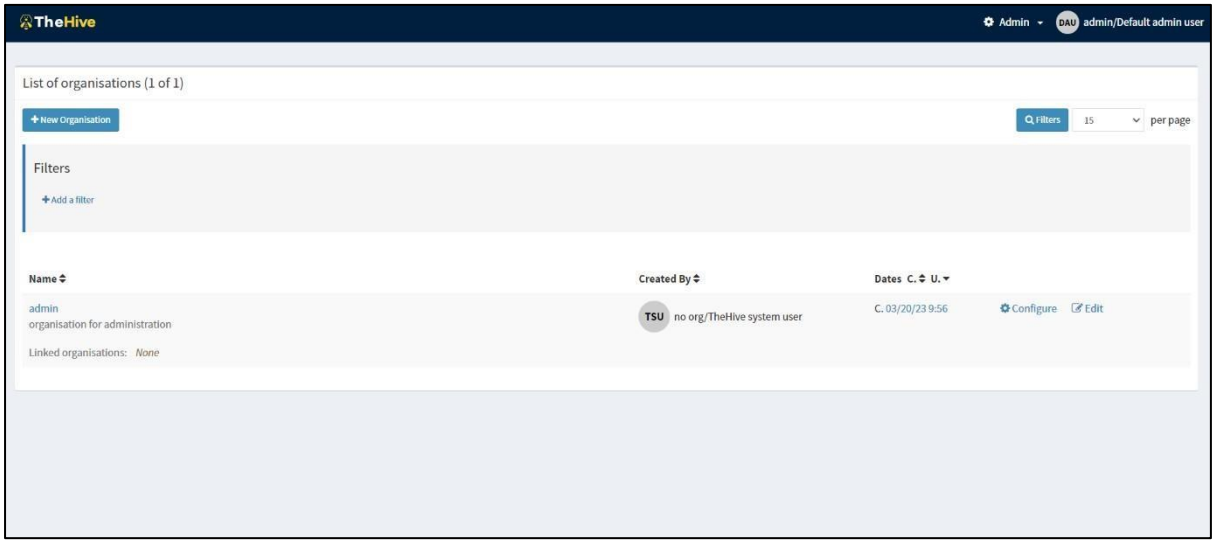
(Figure 30 Changing ownership and installing)

Now after installing hive check if is active or deactive.



(Figure 31 Tailing application.log check active status)

Now our Hive is active check if we can access the console or not?



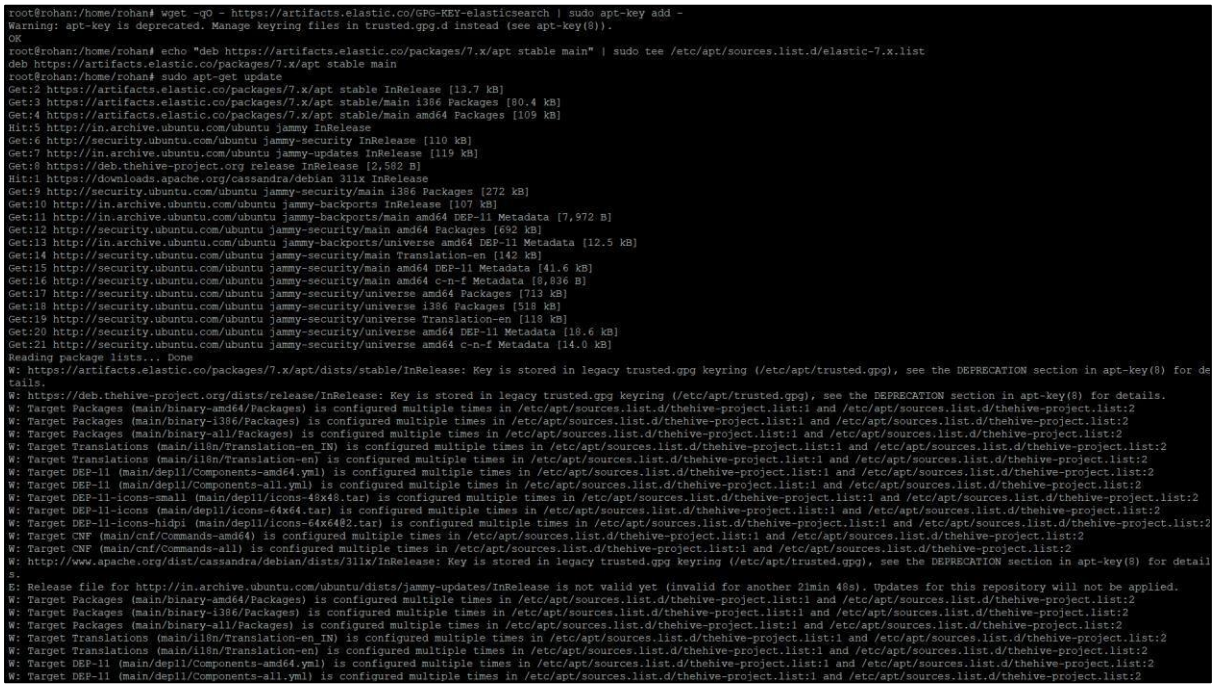
(Figure 32 Hive Console)

6.5 Installation of Cortex:

Now to install cortex, you need to first install elastic search and then after that you can install,

Now to install Elasticsearch in our ubuntu box,

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch |  sudo gpg --dearmor -o
/usr/share/keyrings/elasticsearch-keyring.gpg  echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg]
stable main" |  sudo tee /etc/apt/sources.list.d/elastic-7.x.list  sudo apt install elasticsearch
```

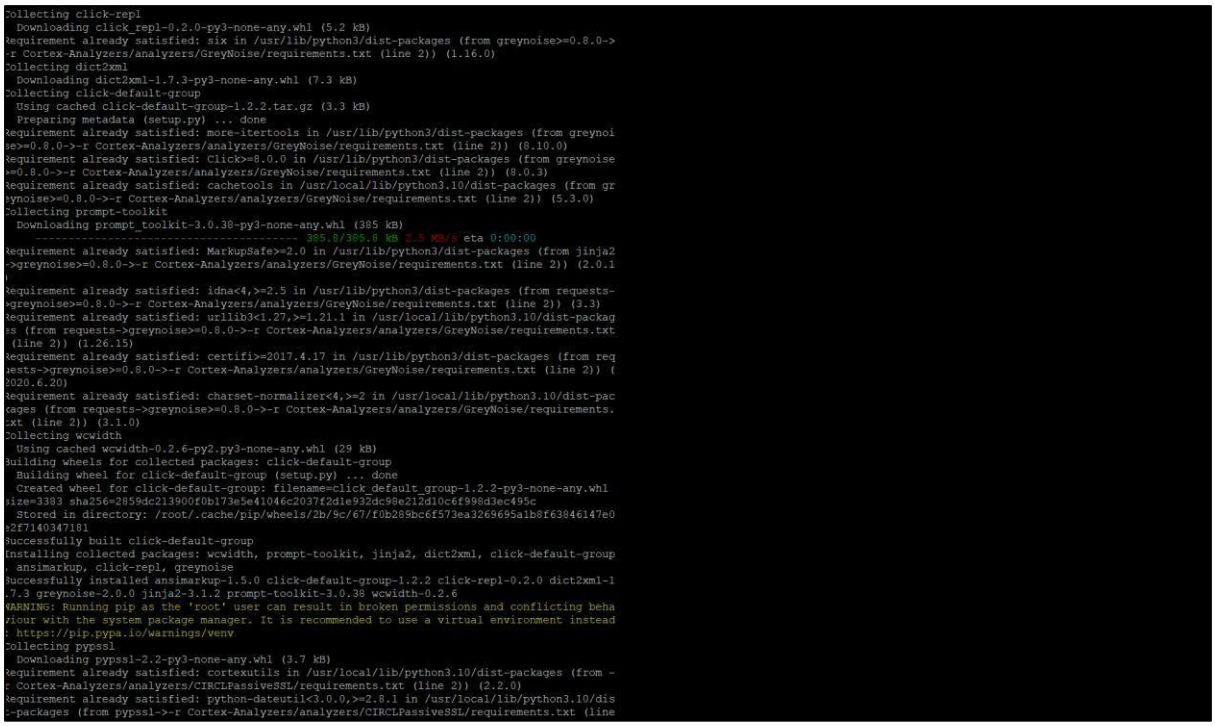


(Figure 33 getting GPG key and checking elastic version)

Now run Elasticsearch installation in our box,

(Figure 34 Installation of Elasticsearch)

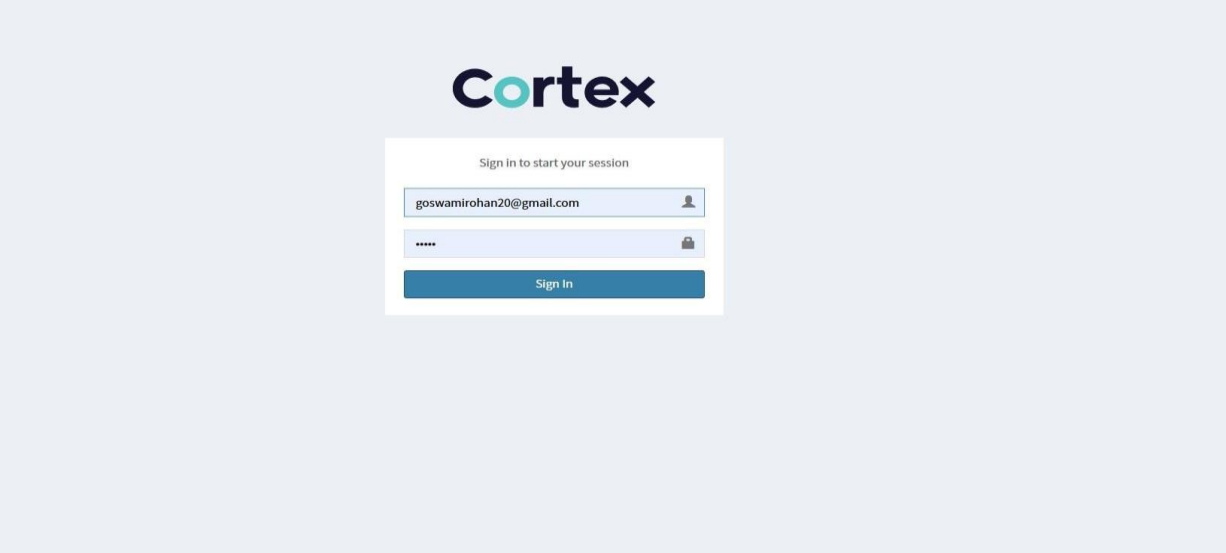
After running it, you need to then install the cortex in our system,



(Figure 35 Now Installing Cortex and analyzers)

Our packages repository hosts all of the published packages. Along with binary packages (zip archive), we now support Debian and RPM packages. Our GPG key, 562CBC1C, is used to sign each and every package.
0CD5 AC59 DE5C 5A8E 0EE1 3849 3D99 BB18 562C BC1C is its fingerprint.

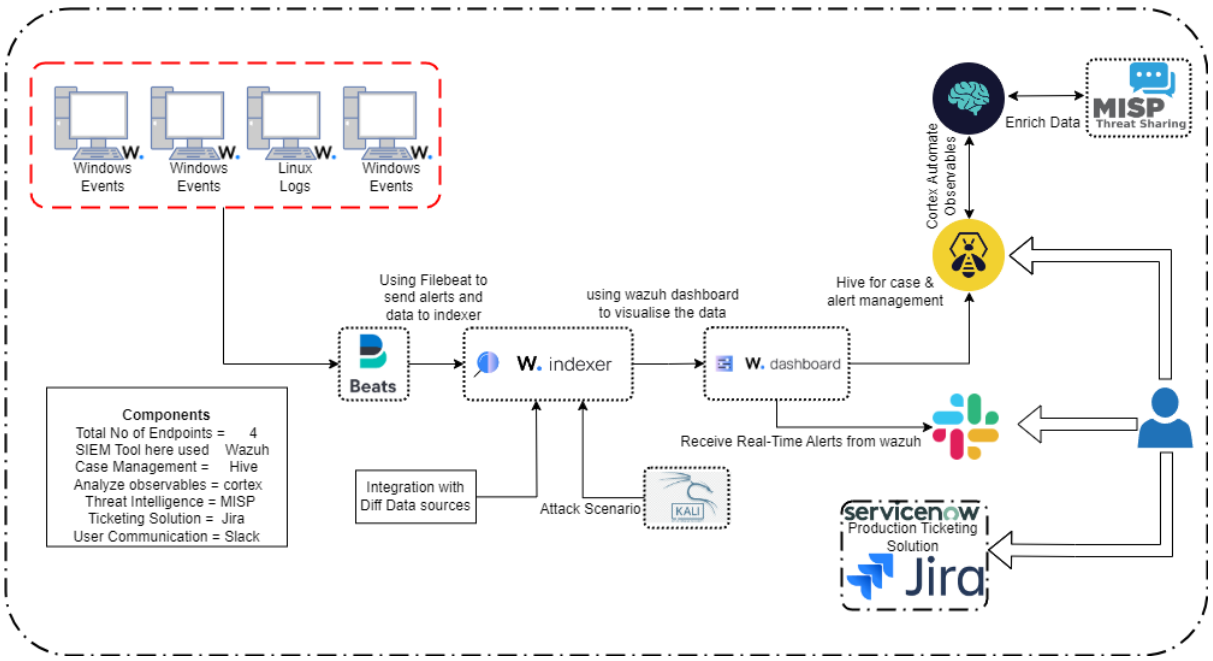
Check if our cortex is active or not?
Systemctl status cortex
After running this command, you can see cortex console after that,



(Figure 36 Cortex Console for signin)

As our requirements are already installed now let's move towards the architecture part.

6.6 Project Architecture:



(Figure 37 Our Lab Architecture)

Workflow of Architecture:

There are total 4 agents are installed in Windows and Linux Machine.

Wazuh agent will be installed which will collect the events or logs and send it to our File beat.

We are using filebeat to send the alerts and data to our Wazuh indexer.

Now Wazuh indexer will indexes that data and using Wazuh dashboard we will able to visualise the data.

Now if any alerts seem to be suspicious, we will have to investigate it further for that we have Hive tool which will help us manage incidents or alerts.

We will have our Hive integrated with Cortex which will help us automate the IOC which given to us.

For user communication & confirmation regarding events done by them or not for that, we have Slack.

Now our Wazuh will be integrated with different data sources like Virus total, AWS, Active-response, CDB Lists, Sysmon, Windows Defender, Hive, Cortex, Guard-duty.

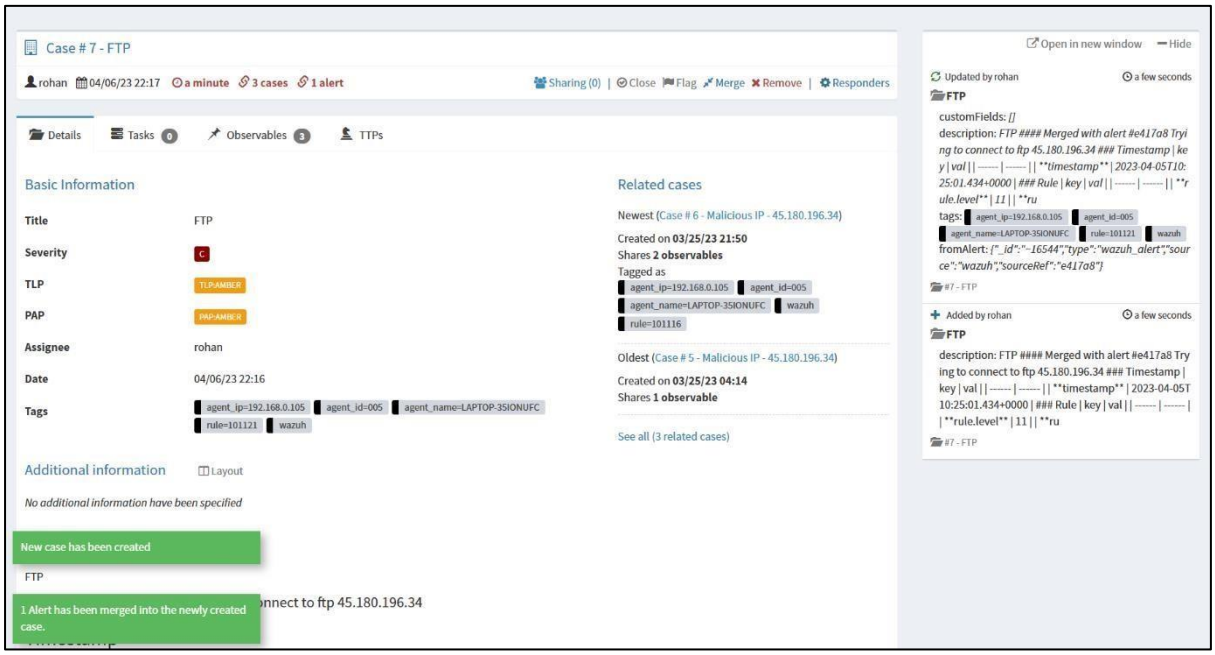
Let's take an example to demonstrate how we can identify if an individual attempting to run the FTP command has a malicious IP address. We'll explore how we can detect and evaluate the IP address's reputation.

Step 1: First you can see that in our SIEM we have observed that an individual attempting to run the FTP command which has a malicious IP.

(Figure 38 FTP Event Triggered)

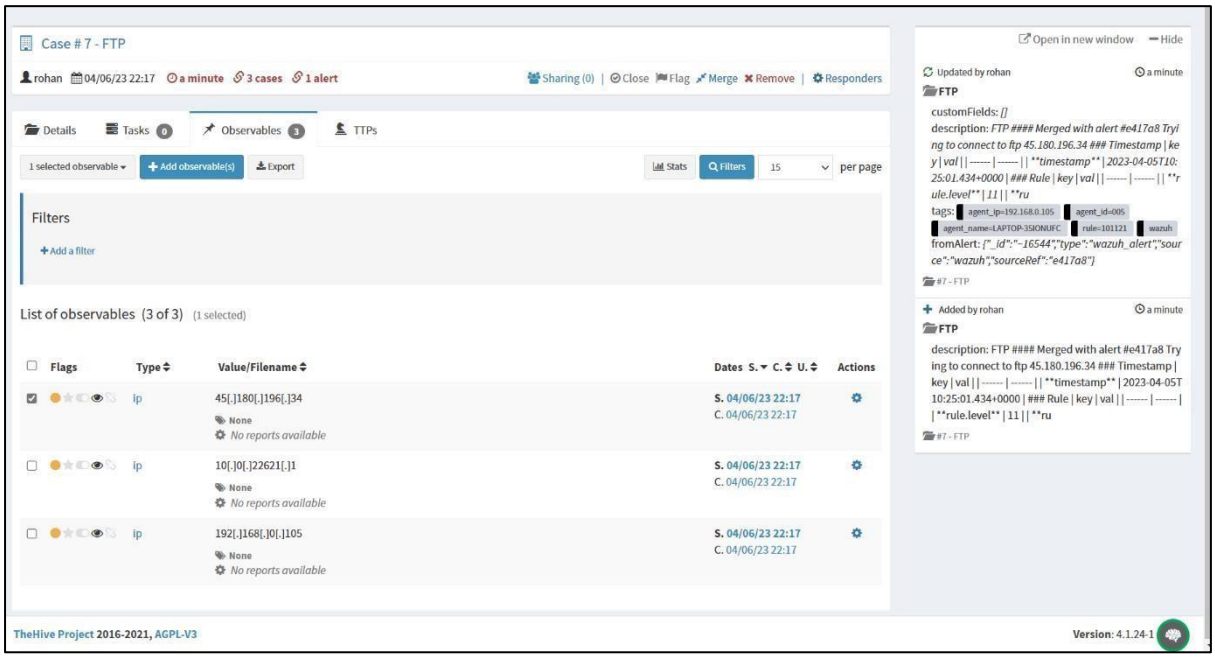
Step 2: For this we have already got this alert automatically in our Hive which is case management tool.

Step 3: Now let's create a case regarding this alert.



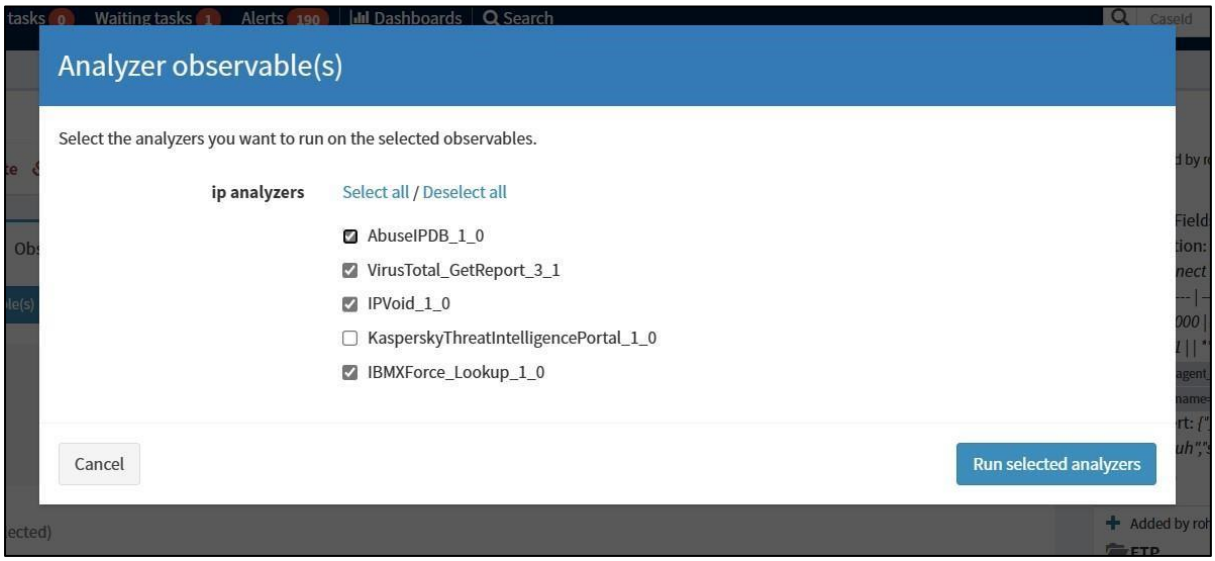
(Figure 39 Creating Case on Hive)

Step 4: Let's go to the observables section there we can run our analyzer's from there.



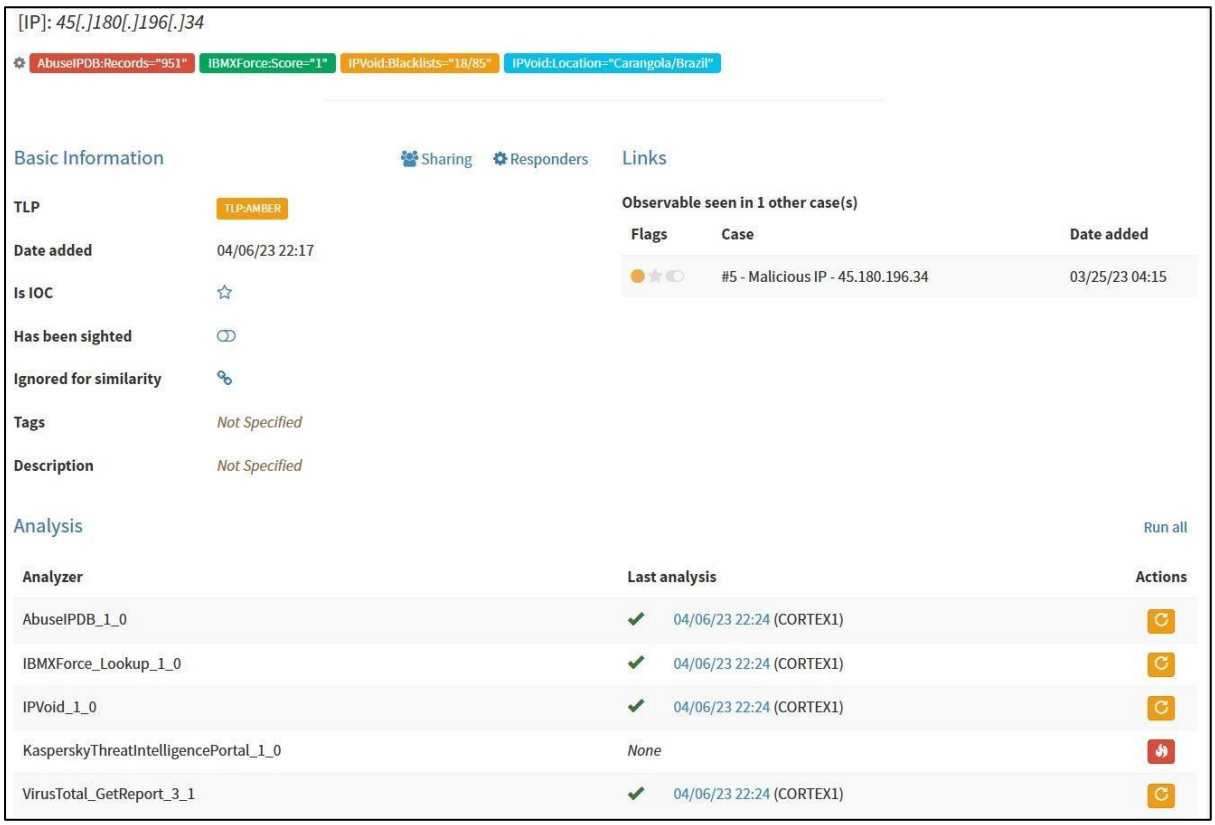
(Figure 40 Running analyzer from observables)

Step 5: Now basically what I have done is integrate different threat intel with our cortex tool so that we don't have to go to particular website and search for that.



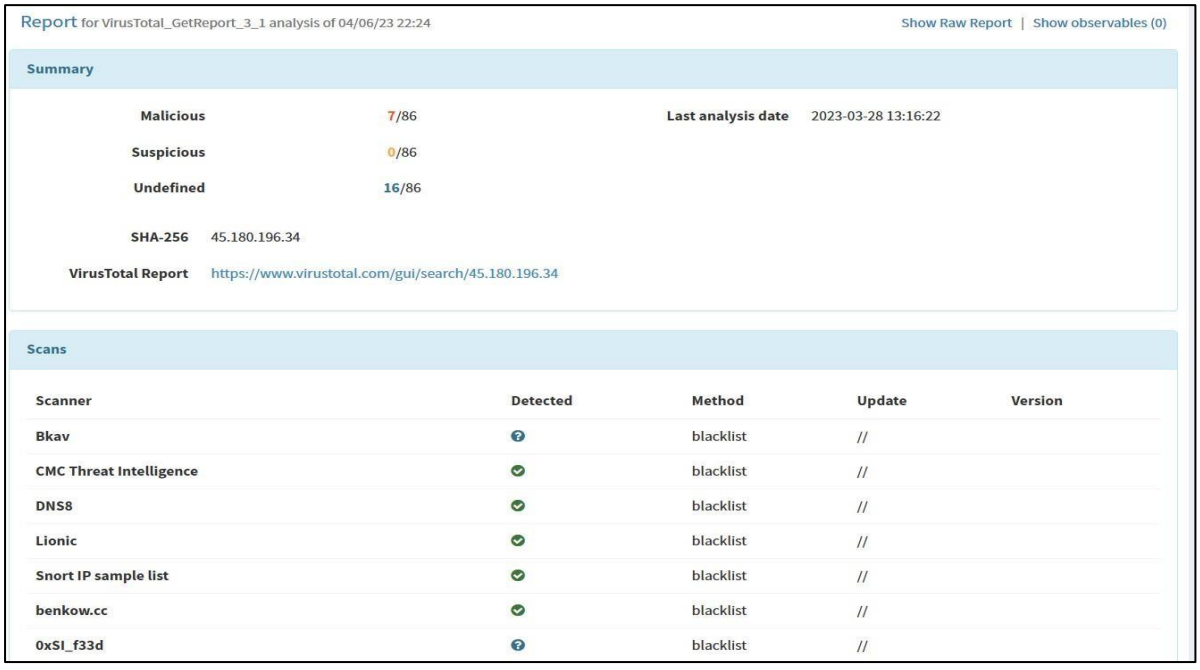
(Figure 41 Selecting analyzers to run on observables)

Step 6: Now you can see the results which I want to find out and we can see that almost all the all the 3rd party vendor are saying it is trying to communicate with malicious IP.



(Figure 42 Output from selected analyzers)

Step 7: You can see detail report in the below figure:



(Figure 43 Report of Analyzers)

6.7 Summary of Hypothesis:

We have observed that an attacker was trying to perform an SSH Brute-Force attack on our Linux Server, so to detect an SSH Brute-Force we need to create a rule which will detect brute force attack in our SIEM tool and also, we will see how Qualys Tool help us here in this Hypothesis.

H1: Multiple authentication failure followed by a success.

H2: Detect a Brute-force attack.

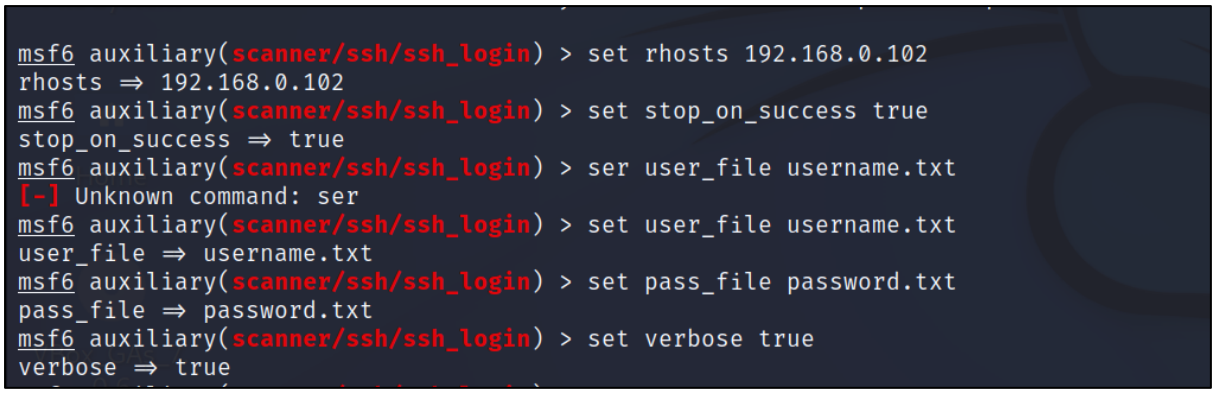
Attacker's Perspective:

First of all, the attacker was able to find out that our Linux server's SSH Port is open.

Using NMAP he was able to get some information,

(Figure 44 Information Gathering)

After getting to know that Port 22 is open, the attacker was trying to perform Brute-force on one of our servers.



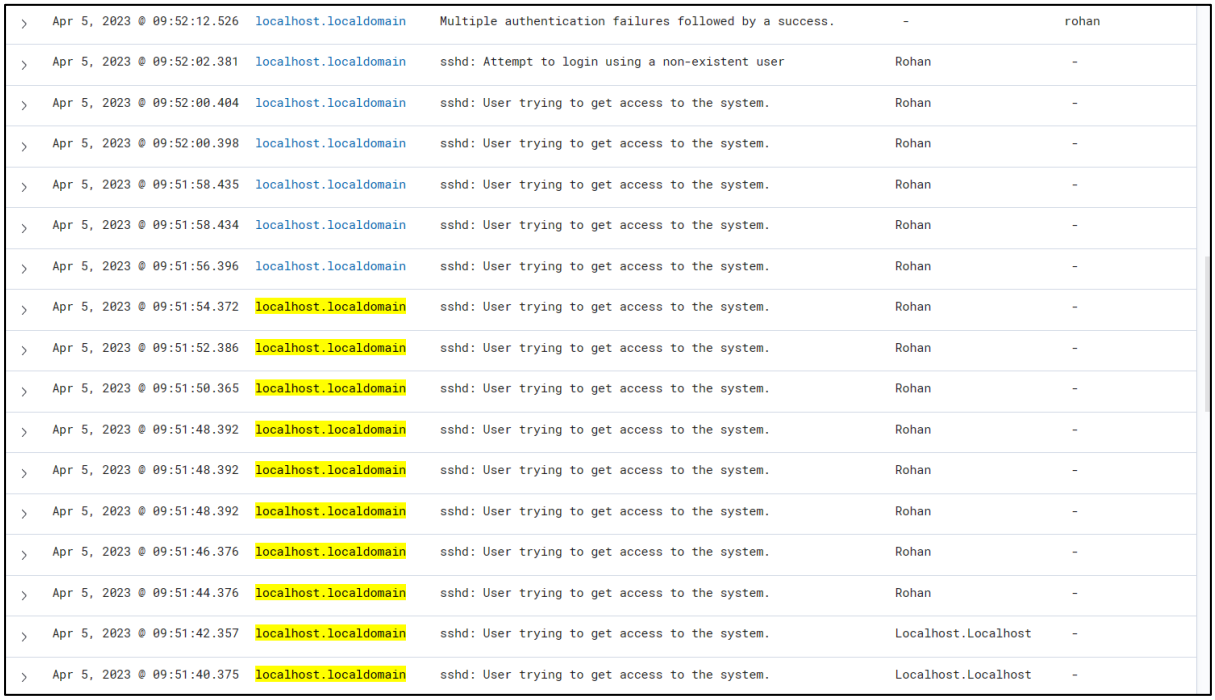
(Figure 45 Exploiting 22 Port)

Now after exploiting this, the attacker was able to get a successful Username and password of one of our servers.

(Figure 46 Successful attack)

Now let's see what our SIEM tool detects here

Now I have created a modified predefined rule in my SIEM tool which will detect if **Multiple authentication failure followed by a success.**



›	Apr 5, 2023 @ 09:52:12.526	localhost.localdomain	Multiple authentication failures followed by a success.	-	rohan
›	Apr 5, 2023 @ 09:52:02.381	localhost.localdomain	sshd: Attempt to login using a non-existent user	Rohan	-
›	Apr 5, 2023 @ 09:52:00.404	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:52:00.398	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:58.435	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:58.434	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:56.396	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:54.372	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:52.386	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:50.365	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:48.392	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:48.392	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:48.392	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:46.376	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:44.376	localhost.localdomain	sshd: User trying to get access to the system.	Rohan	-
›	Apr 5, 2023 @ 09:51:42.357	localhost.localdomain	sshd: User trying to get access to the system.	Localhost.Localhost	-
›	Apr 5, 2023 @ 09:51:40.375	localhost.localdomain	sshd: User trying to get access to the system.	Localhost.Localhost	-

(Figure 47 First hypothesis proven)

So here my hypothesis seems to be true, now let’s look for another hypothesis which I have assumed.

Here I have created a rule for the 2nd hypothesis: if an attacker will try to perform a Brute-Force attack, my wazuh will detect it in our SIEM Tool.

H2 Query:

```
<rule id="200009" level="10" frequency="3" timeframe="120" ignore="60">
    <if_matched_sid>5760</if_matched_sid>
    <description>Brute force attempt to login</description>
    <mitre>
      <id>T1110</id>
    </mitre>
    <group>authentication_failures,gdpr_IV_35.7.d,gdpr_IV_32.2,hipaa_164.312.b,nist_800_53_SI.4,nist_800_53_AU.14,nist_800_53
_AC.7,pci_dss_11.4,pci_dss_10.2.4,pci_dss_10.2.5,tsc_CC6.1,tsc_CC6.8,tsc_CC7.2,tsc_CC7.3,</group>
  </rule>
```

(Figure 48 Created Rule)

Now let's test our second hypothesis and see if it is true or false.

timestamp per 30 minutes

Time	agent.name	rule.description	data.srcuser	data.dstuser
Apr 5, 2023 @ 10:02:21.183	localhost.localdomain	Multiple authentication failures followed by a success.	-	rohan
Apr 5, 2023 @ 10:02:15.042	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:02:13.024	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:02:11.040	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:02:09.046	localhost.localdomain	sshd: Attempt to login using a non-existent user	Rohan	-
Apr 5, 2023 @ 10:02:09.046	localhost.localdomain	sshd: Attempt to login using a non-existent user	Rohan	-
Apr 5, 2023 @ 10:02:07.054	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:02:07.050	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:02:03.023	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:02:03.018	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:02:01.046	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:01:58.997	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:01:57.017	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:01:57.012	localhost.localdomain	Brute force attempt to login	Rohan	-
Apr 5, 2023 @ 10:01:53.010	localhost.localdomain	Brute force attempt to login	Rohan	-

(Figure 49 Testing Second Hypothesis)

So now to block this type of attack we can use Active-Response which is one of the features of Wazuh.

After adding some rules in our Wazuh we can see that it detects and blocks the SSH Brute-Force attack.

›	Apr 5, 2023 @ 10:20:17.635	localhost.localdomain	Host Unblocked by firewall-drop Active Response	-	-	IEUser
›	Apr 5, 2023 @ 10:19:15.656	localhost.localdomain	Host Blocked by firewall-drop Active Response	-	-	IEUser
›	Apr 5, 2023 @ 10:19:15.582	localhost.localdomain	Host Blocked by firewall-drop Active Response	-	-	IEUser
›	Apr 5, 2023 @ 10:19:15.581	localhost.localdomain	Brute force attempt to login	IEUser	-	-
›	Apr 5, 2023 @ 10:19:15.547	localhost.localdomain	Brute force attempt to login	IEUser	-	-
›	Apr 5, 2023 @ 10:19:13.579	localhost.localdomain	Host Blocked by firewall-drop Active Response	-	-	IEUser
›	Apr 5, 2023 @ 10:19:13.560	localhost.localdomain	Brute force attempt to login	IEUser	-	-
›	Apr 5, 2023 @ 10:19:11.549	localhost.localdomain	sshd: Attempt to login using a non-existent user	IEUser	-	-
›	Apr 5, 2023 @ 10:09:27.001	localhost.localdomain	WhiteListed IP Login	-	rohan	-
›	Apr 5, 2023 @ 10:02:21.183	localhost.localdomain	Multiple authentication failures followed by a success.	-	rohan	-
›	Apr 5, 2023 @ 10:02:15.042	localhost.localdomain	Brute force attempt to login	Rohan	-	-
›	Apr 5, 2023 @ 10:02:13.024	localhost.localdomain	Brute force attempt to login	Rohan	-	-
›	Apr 5, 2023 @ 10:02:11.040	localhost.localdomain	Brute force attempt to login	Rohan	-	-
›	Apr 5, 2023 @ 10:02:09.046	localhost.localdomain	sshd: Attempt to login using a non-existent user	Rohan	-	-

(Figure 50 Hypothesis Proven)

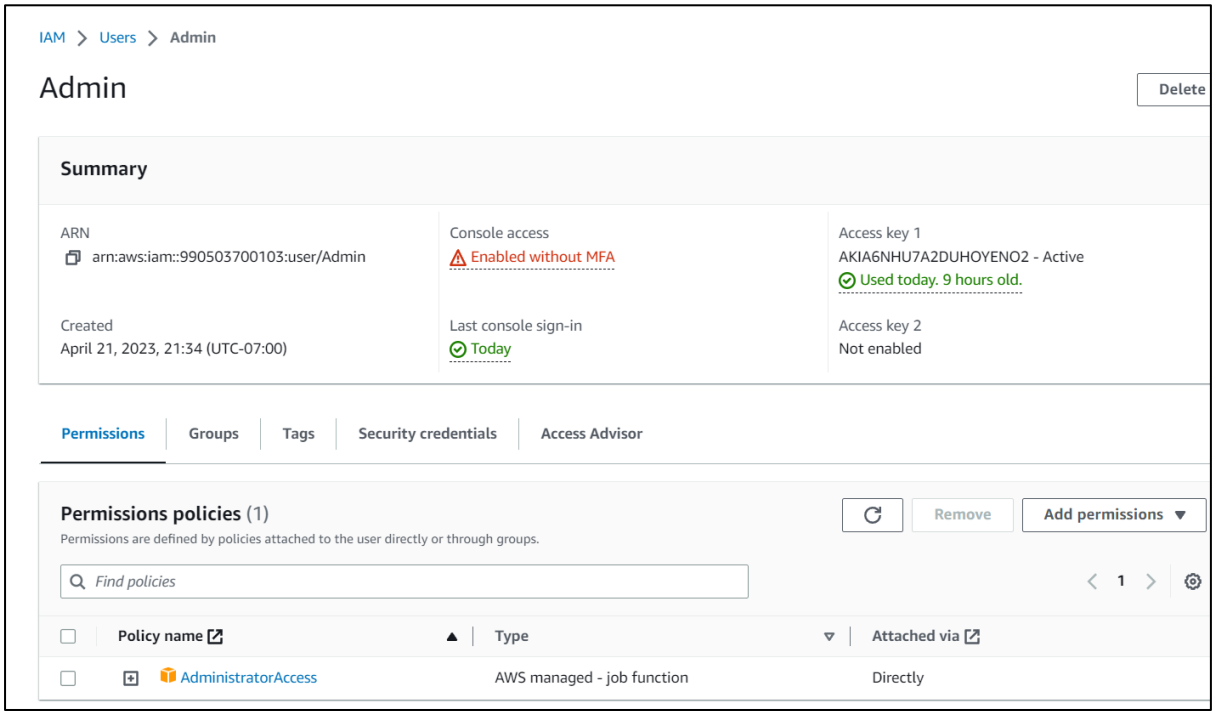
```
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.0.102:22 - Starting bruteforce
[-] 192.168.0.102:22 - Failed: 'IEUser:123'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.0.102:22 - Failed: 'IEUser:321'
[-] Could not connect: The connection with (192.168.0.102:22) timed out.
[-] Could not connect: The connection with (192.168.0.102:22) timed out.
[-] Could not connect: The connection with (192.168.0.102:22) timed out.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

(Figure 51 Blocking out attack)

So yeah, we are Pretty much done with our hypothesis and we seem to have proven our hypothesis right.

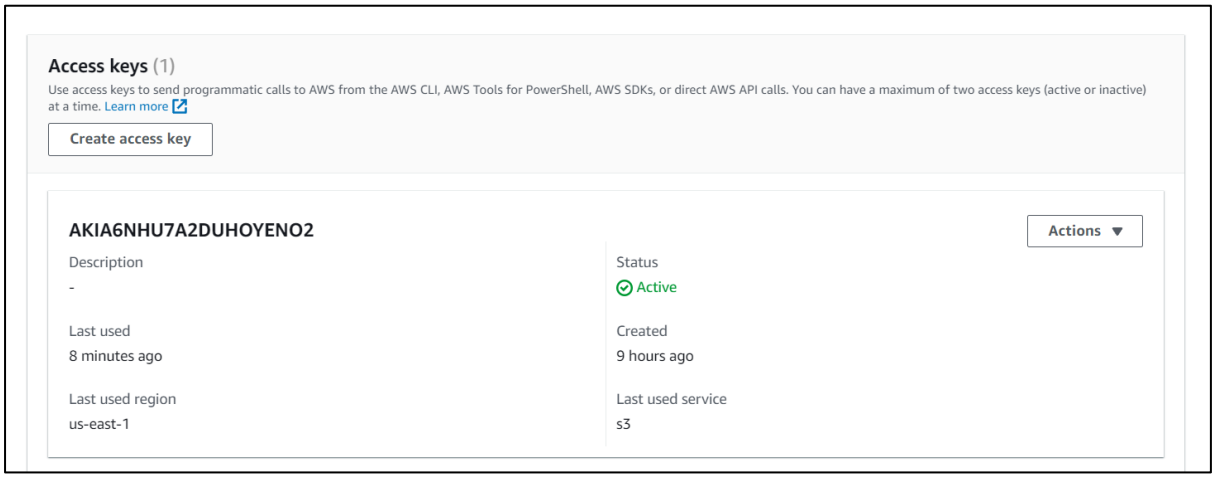
6.8 Integration AWS with Wazuh:

Step 1: Create an AWS Account which should have an admin full access,



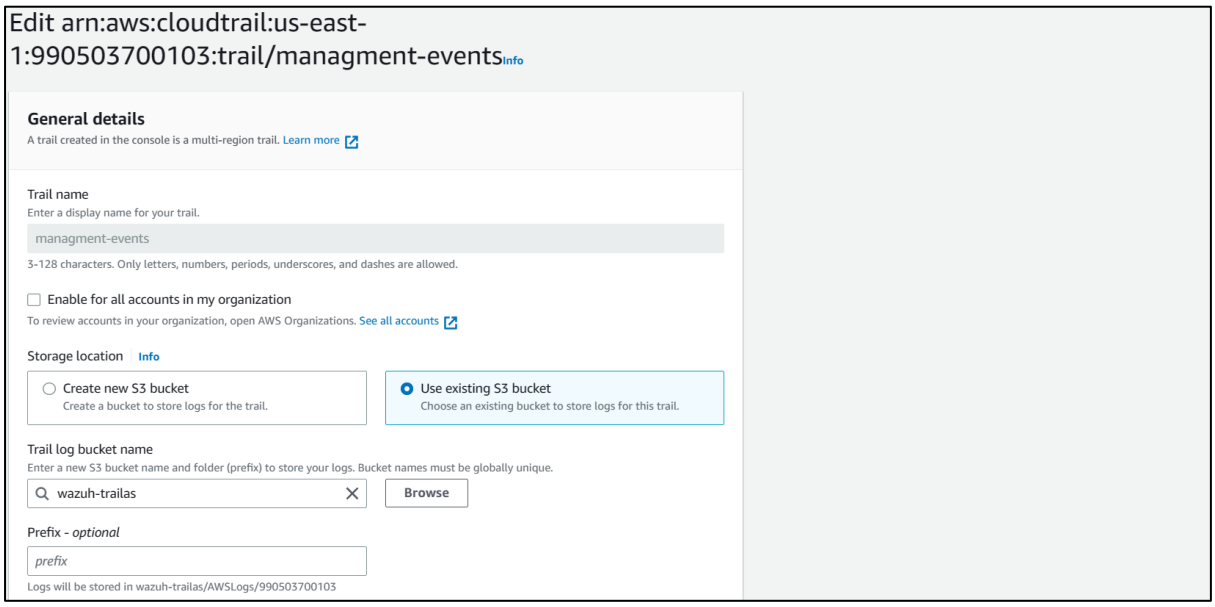
(Figure 52 Creating Admin Account with Admin Access)

Step 2: Make sure you have access key and secret access key enabled.



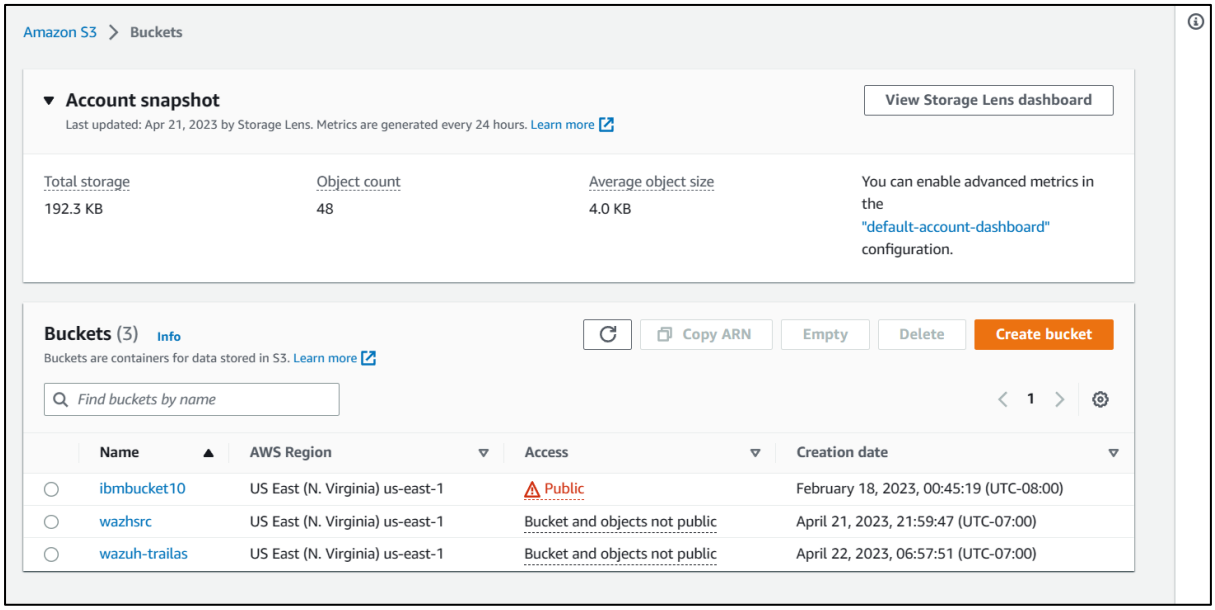
(Figure 53 Make Sure Access key and Secret Access key is enabled or else you have to create)

Step 3: While creating a trail make sure you specify bucket where we can store AWS services logs.



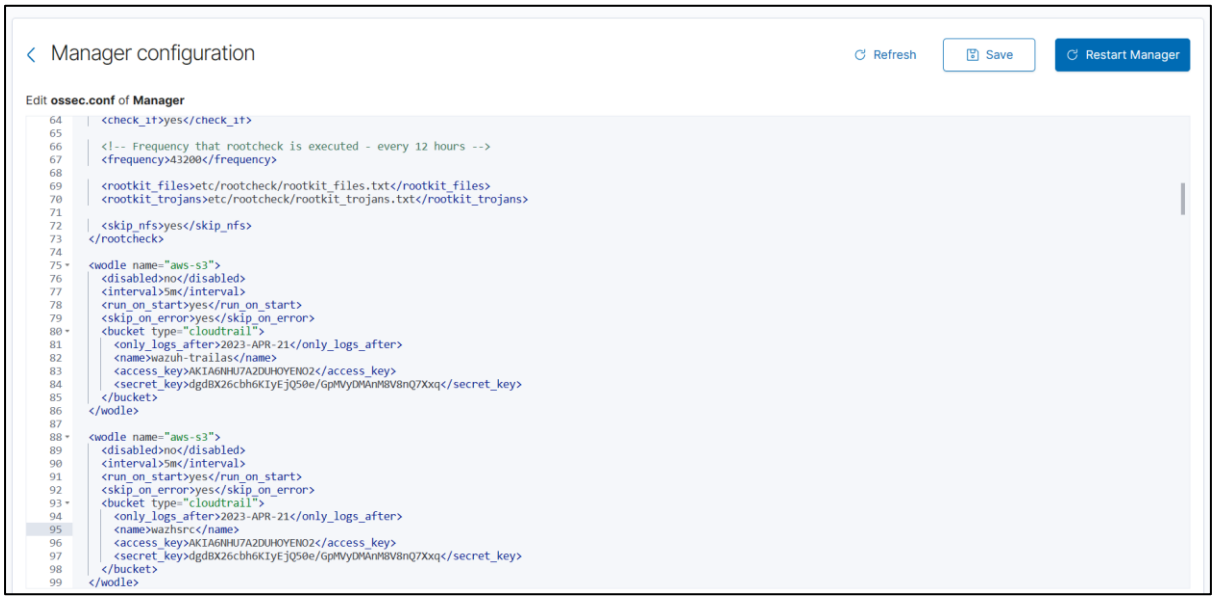
(Figure 54 Creating cloud trail to store logs in S3)

Step 4: As you can see that our logs will be stored in wazhsrc and wazuh-trailas.



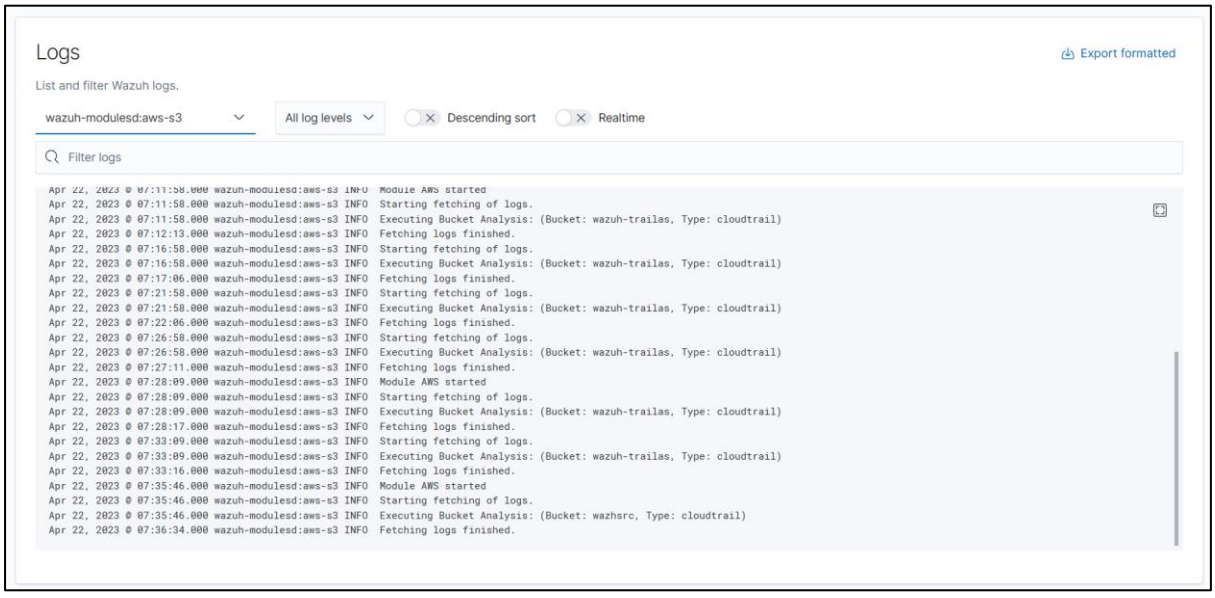
(Figure 55 Logs will be stored in 2 Buckets)

Step 5: we will add wodle in our configuration files so that we will able to see AWS services Logs in our Wazuh.



(Figure 56 Adding Bucket name and access key to conf file)

Step 6: As you can see that we were able to fetch the logs from our bucket.



(Figure 57 Fetching logs finishes)

Step 7: Now you can monitor AWS Events from our Wazuh only which is a good thing.

	Time	data.aws.source	rule.description	rule.level	rule.id
›	Apr 22, 2023 @ 07:36:33.785	cloudtrail	AWS Cloudtrail: cloudtrail.amazonaws.com - StopLogging.	3	80202
›	Apr 22, 2023 @ 07:36:00.700	cloudtrail	AWS Root Account Activity Detection: signin.amazonaws.com - ConsoleLogin	12	100002
›	Apr 22, 2023 @ 07:35:52.565	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - StopInstances.	3	80202
›	Apr 22, 2023 @ 07:35:52.562	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - CreateKeyPair.	3	80202
›	Apr 22, 2023 @ 07:35:52.562	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - RunInstances.	3	80202
›	Apr 22, 2023 @ 07:35:52.562	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - CreateKeyPair. Error: Client.MissingParameter.	4	80203
›	Apr 22, 2023 @ 07:33:15.352	cloudtrail	AWS Cloudtrail: iam.amazonaws.com - ListUsers.	3	80202
›	Apr 22, 2023 @ 07:33:15.351	cloudtrail	AWS Cloudtrail: iam.amazonaws.com - ListGroups.	3	80202
›	Apr 22, 2023 @ 07:33:15.348	cloudtrail	AWS Root Account Activity Detection: signin.amazonaws.com - ConsoleLogin	12	100002
›	Apr 22, 2023 @ 07:22:05.649	cloudtrail	AWS Cloudtrail: signin.amazonaws.com - ConsoleLogin - User login success.	3	80253
›	Apr 22, 2023 @ 07:12:10.554	cloudtrail	AWS Cloudtrail: s3.amazonaws.com - PutBucketPolicy.	3	80202
›	Apr 22, 2023 @ 07:12:10.554	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - StopInstances.	3	80202
›	Apr 22, 2023 @ 07:12:10.311	cloudtrail	AWS Cloudtrail: cloudtrail.amazonaws.com - StopLogging.	3	80202
›	Apr 22, 2023 @ 07:12:09.426	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - StartInstances.	3	80202
›	Apr 22, 2023 @ 07:12:09.415	cloudtrail	AWS Cloudtrail: s3.amazonaws.com - PutBucketPolicy.	3	80202
›	Apr 22, 2023 @ 07:12:09.410	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - StopInstances.	3	80202
›	Apr 22, 2023 @ 07:12:09.398	cloudtrail	AWS Cloudtrail: ec2.amazonaws.com - RunInstances.	3	80202

(Figure 58 Generating AWS Events)

Chapter 7
7.0 CONCLUSION & FUTURE WORK

Wazuh:

Wazuh requires proper configuration and maintenance to be effective. Additionally, while Wazuh provides comprehensive security capabilities, it should be used in conjunction with other security tools and practices to provide a holistic approach to security.

TheHive:

TheHive is a powerful and flexible tool that can help security teams manage incidents more effectively. However, as with any tool, it's important to ensure that it's properly configured and used in conjunction with other security tools and best practices to maximize its effectiveness.

Cortex:

We can integrate cortex with hive to provide better work and analysis of the alert which we have got from our SIEM.

So, in conclusion we can use these different tools to provide better security and detection to our client's security posture and before that we first need to understand the architecture also we will be Deploying MISP and Integrating MISP with Cortex by using it's API key.

Chapter 8:
8.0 Reference

https://documentation.wazuh.com/current/getting-started/architecture.html

https://docs.thehive-project.org/thehive/

https://docs.thehive-project.org/cortex/

https://documentation.wazuh.com/current/user-manual/capabilities/active-response