National College of Ireland

Configuration Manual

MSc Research Project
Cybersecurity

Vanessa Pereira
Student ID: x21179875

School of Computing
National College of Ireland

Supervisor: Evgeniia Jayasekera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vanessa Pereira
Student ID:	X21179875
Programme:	MSc Cyber Security **Year:** 2022-2023
Module:	Research Project
Supervisor:	Evgeniia Jayasekera
Submission Due Date:	18th September 2023
Project Title:	Cyber Attack Detection and Response using open source tools
Word Count:	2506 **Page Count** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vanessa Pereira
Date:	15th September 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vanessa Pereira
Student ID: x21179875

1. Section 1: System Configuration:

The setup is installed on Virtualbox VMs and following is the configuration details.

	Memory (RAM)	Storage (HDD)	CPU	Operating System
Wazuh	8 GB	50 GB	2	Ubuntu 22.04
MISP	2 GB	25 GB	1	Ubuntu 22.04
TheHive / Cortex	4 GB	50 GB	2	Ubuntu 22.04
Shuffle/ Apache Webserver	6 GB	50 GB	2	Ubuntu 22.04

2. Section 2: Setting up the Webserver:

An apache webserver is setup on the same machine as Shuffle. To evaluate the web attacks detection capabilities, we integrate teler a lightweight HTTP IDS.

A DVWA (Damn Vulnerable Web Application) is set up on the Ubuntu machine [1].

The Wazuh agent is installed on the machine.

Configuring Teler [2]:

Download and extract the teler binary:

```
wget https://github.com/kitabisa/teler/releases/download/v2.0.0-rc.3/teler_2.0.0-rc.3_linux_amd64.tar.gz
tar -xvzf teler_2.0.0-rc.3_linux_amd64.tar.gz
```

Download the sample configuration file teler.example.yaml and rename it to teler.yaml.
Modify the log_format parameter in teler.yaml for Apache Configuration:

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

Update the logs parameter in teler.yaml for logging:

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

Add a configuration block to the Wazuh agent configuration file (/var/ossec/etc/ossec.conf) to monitor the teler output.log file:

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

Restart the Wazuh agent:

```
systemctl restart wazuh-agent
```

Run the following command to start the teler application:

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

Configure Wazuh Rules:

Add custom rules to the Wazuh local_rules.xml file (/var/ossec/etc/rules/local_rules.xml). These rules are based on teler alerts and work with predefined teler.yaml rules:

```
<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>
```

Restart the Wazuh manager:

```
systemctl restart wazuh-manager
```

To Emulate the attack run the following command

```
nikto -h http://<UBUNTU_IP>/dvwa/
```

This apache webserver is used to simulate a web application attack in our lab setup to generate logs in wazuh which will be forwarded to the shuffle to trigger the workflow of creating a case in TheHive and carrying out analysis through cortex integration.
The workflow configuration is specified at the end of this document.

3.Section 3: Installing the Tools

a.Installation of Wazuh:

Wazuh Indexer
Wazuh Dashboard
Graylog
Wazuh Manager

Wazuh Indexer Installation: [3] [4] [5]

Install the new Wazuh-Indexer 4.4 release

Install Prerequisites:

```
apt-get install debconf adduser procps
apt-get install gnupg apt-transport-https
```

Install GPG Key and Add repo

```
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-
ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644 /usr/share/keyrings/wazuh.gpg

echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
/etc/apt/sources.list.d/wazuh.list
```

Install

```
apt-get update
apt-get -y install wazuh-indexer
```

Deploying certificates for encryption and security.

The wazuh-certs-tool.sh script used for this is a custom script by SOCFortress

```
wget https://raw.githubusercontent.com/socfortress/Wazuh-Rules/main/wazuh-certs-tool.sh -q -O /tmp/wazuh-certs-tool.sh
wget https://raw.githubusercontent.com/socfortress/Wazuh-Rules/main/config.yml -q -O /tmp/config.yml
```

Update the /tmp/config.yml file with your hostname and IP.

Run the /tmp/wazuh-certs-tool.sh - A script to generate the certificates.

Obtain the the value of the CN of the hostname.pem certificate.

```
openssl x509 -in wazuh-indexer01.<hostname>.demo -text -noout
NODE_NAME=wazuh-indexer01.<hostname>.demo
```

Copy certs into /etc/wazuh-indexer/certs

```
mkdir /etc/wazuh-indexer/certs
cd /tmp/wazuh-certificates
cp ./$NODE_NAME.pem ./$NODE_NAME-key.pem ./admin.pem ./admin-key.pem ./root-ca.pem /etc/wazuh-indexer/certs/
mv -n /etc/wazuh-indexer/certs/$NODE_NAME.pem /etc/wazuh-indexer/certs/indexer.pem
mv -n /etc/wazuh-indexer/certs/$NODE_NAME-key.pem /etc/wazuh-indexer/certs/indexer-key.pem
```

Set ownership and permissions

```
chmod 500 /etc/wazuh-indexer/certs
chmod 400 /etc/wazuh-indexer/certs/*
chown -R wazuh-indexer:wazuh-indexer /etc/wazuh-indexer/certs
```

Wazuh Indexer Configuration

network.host: Specifies the address for this node, serving as both HTTP and transport traffic bindings. This address will be used for node binding and as the publication address. Utilize the identical node address specified in the config.yml to generate SSL certificates.

node.name: Identifies the name of the Wazuh indexer node in accordance with the configuration file (config.yml). As an example, "node-1" could be used.

cluster.initial_master_nodes: Consists of a list of names belonging to master-eligible nodes. These names are outlined within the config.yml file.

plugins.security.nodes_dn: Enumerates the Distinguished Names of certificates linked to all nodes within the Wazuh indexer cluster. Modify the common names (CN) and corresponding values to align with the configuration settings and definitions provided in the config.yml file.

Memory Locking

```
1.      Uncomment or add this line to the /etc/wazuh-indexer/opensearch.yml file:
bootstrap.memory_lock: true
2.      Edit the limit of system resources:
nano /usr/lib/systemd/system/wazuh-indexer.service
3.      Place under [Service] block LimitMEMLOCK=infinity
4.      Set JVM Options to 50% of total memory available
nano /etc/wazuh-indexer/jvm.options
```

Start the Service

```
systemctl daemon-reload
systemctl enable wazuh-indexer
systemctl start wazuh-indexer
```

Cluster Initialization

Run the Wazuh indexer indexer-security-init.sh script on any Wazuh indexer node to load the new certificates information and start the single-node cluster.

```
/usr/share/wazuh-indexer/bin/indexer-security-init.sh
```

Install Wazuh-Dashboard: [3] [4] [5]

Only needs to be installed on Wazuh-Indexer Node 01

1. Install Prerequisites:

```
apt-get install debhelper tar curl libcap2-bin -y
apt-get update
```

2. Install:

```
apt-get -y install wazuh-dashboard
```

3. Configure Wazuh-Dashboard: Set up certificates to enable the connection of the Wazuh-Dashboard service with the Wazuh-Indexer cluster.

```
mkdir /etc/wazuh-dashboard/certs
cp /etc/wazuh-indexer/certs/indexer.pem /etc/wazuh-dashboard/certs/
cp /etc/wazuh-indexer/certs/indexer-key.pem /etc/wazuh-dashboard/certs/
cp /etc/wazuh-indexer/certs/root-ca.pem /etc/wazuh-dashboard/certs/
chmod 500 /etc/wazuh-dashboard/certs
chmod 400 /etc/wazuh-dashboard/certs/*
chown -R wazuh-dashboard:wazuh-dashboard /etc/wazuh-dashboard/certs
```

4. Edit the /etc/wazuh-dashboard/opensearch_dashboards.yml to set configuration:

```
server.host: 0.0.0.0
server.port: 443
opensearch.hosts: ["https://wazuh-indexer01.<hostname>.demo:9200"]
opensearch.ssl.verificationMode: certificate
#opensearch.username:
#opensearch.password:
opensearch.requestHeadersWhitelist: ["securitytenant","Authorization"]
opensearch_security.multitenancy.enabled: false
opensearch_security.readonly_mode.roles: ["kibana_read_only"]
server.ssl.enabled: true
server.ssl.key: "/etc/wazuh-dashboard/certs/indexer-key.pem"
server.ssl.certificate: "/etc/wazuh-dashboard/certs/indexer.pem"
opensearch.ssl.certificateAuthorities: ["/etc/wazuh-dashboard/certs/root-ca.pem"]
uiSettings.overrides.defaultRoute: /app/wazuh
```

5. Start Wazuh-Dashboard

```
systemctl daemon-reload
systemctl enable wazuh-dashboard
systemctl start wazuh-dashboard
```

6. Securing The Cluster

Modify the initial passwords for both the admin and wazuh-dashboard users. Utilize the Wazuh passwords tool on Wazuh Indexer Node 01 to update the passwords of the Wazuh indexer users. Run Script:

```
/usr/share/wazuh-indexer/plugins/opensearch-security/tools/wazuh-passwords-tool.sh --change-all
```

7. Change Wazuh Dashboard Password:

Execute the provided command on your Wazuh dashboard node to modify the kibanaserver password within the Wazuh dashboard keystore. Replace "<kibanaserver-password>" with the randomly generated password from the initial step.

```
echo <kibanaserver-password> | /usr/share/wazuh-dashboard/bin/opensearch-dashboards-keystore --allow-root add -f --stdin
opensearch.password
```

8. Restart the service

```
systemctl restart wazuh-dashboard
```

9. Create User

Create the graylog user and give it a backend role of admin

10. Configure Connection to Wazuh-Manager

To engage with the Wazuh API using the Wazuh Dashboards, it's essential to set up the configuration of the Wazuh Dashboard service, enabling it to establish a connection with the API of the Wazuh Master node.

```
nano /usr/share/wazuh-dashboard/data/wazuh/config/wazuh.yml
```

BLOCK TO CHANGE

hosts:
- default:
  url: https://*MASTER_NODE*
  port: 55000
  username: wazuh-wui
  password: wazuh-wui
  run_as: false

Install Graylog: [3] [4] [5]

Install Graylog Version 5.x and connect it to our Wazuh-Indexer.

1. Install MongoDB

```
apt-get install gnupg
wget -qO - https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -
echo "deb http://repo.mongodb.org/apt/debian bullseye/mongodb-org/6.0 main" | sudo tee /etc/apt/sources.list.d/mongodb-
org-6.0.list
apt-get update
apt-get install -y mongodb-org
systemctl daemon-reload
systemctl enable mongod.service
systemctl restart mongod.service
systemctl --type=service --state=active | grep mongod
```

2. Install Graylog

```
wget https://packages.graylog2.org/repo/packages/graylog-5.0-repository_latest.deb
sudo dpkg -i graylog-5.0-repository_latest.deb
apt-get update && sudo apt-get install graylog-server
```

3. Configure Graylog

Review the guidelines provided in the configuration file found at /etc/graylog/server/server.conf and make necessary adjustments as required. Furthermore, ensure to include password_secret and root_password_sha2, as these elements are obligatory; Graylog won't initiate without them.

```
To create your password_secret run the following command:
pwgen -N 1 -s 96

To create your root_password_sha2 run the following command:
echo -n "Enter Password: " && head -1 </dev/stdin | tr -d '\n' | sha256sum | cut -d" " -f1

Configure the Connection to your Wazuh-Indexer:
elasticsearch_hosts = https://user:pass@wazuh-indexerhostname:9200
```

Graylog now includes JAVA 17 within its package, eliminating the need for a separate installation. Proceed to copy the cacerts file and import the information from rootCA.crt into the Java Keystore.

```
cp -a /usr/share/graylog-server/jvm/lib/security/cacerts /etc/graylog/server/certs/cacerts

/usr/share/graylog-server/jvm/bin/keytool -importcert -keystore /etc/graylog/server/certs/cacerts -storepass changeit -alias
root_ca -file /etc/graylog/server/certs/root-ca.pem

nano /etc/default/graylog-server
```

```
# Path to a custom java executable. By default the java executable of the
# bundled JVM is used.
#JAVA=/usr/bin/java

# Default Java options for heap and garbage collection.
GRAYLOG_SERVER_JAVA_OPTS="-Xms1g -Xmx1g -server -XX:+UseG1GC -XX:-OmitStackTraceInFastThrow"

# Avoid endless loop with some TLSv1.3 implementations.
GRAYLOG_SERVER_JAVA_OPTS="$GRAYLOG_SERVER_JAVA_OPTS -Djdk.tls.acknowledgeCloseNotify=true"

# Fix for log4j CVE-2021-44228
#GRAYLOG_SERVER_JAVA_OPTS="$GRAYLOG_SERVER_JAVA_OPTS -Dlog4j2.formatMsgNoLookups=true"
GRAYLOG_SERVER_JAVA_OPTS="$GRAYLOG_SERVER_JAVA_OPTS -Dlog4j2.formatMsgNoLookups=true -
Djavax.net.ssl.trustStore=/etc/graylog/server/certs/cacerts -Djavax.net.ssl.trustStorePassword=changeit"

# Pass some extra args to graylog-server. (i.e. "-d" to enable debug mode)
GRAYLOG_SERVER_ARGS=""

# Program that will be used to wrap the graylog-server command. Useful to
# support programs like authbind.
GRAYLOG_COMMAND_WRAPPER=""
```

```
systemctl start graylog-server
```

Install Wazuh-Manager: [3] [4] [5]

Installing version 4.4.x of the Wazuh-Manager

Install Prerequisites:

```
apt-get install gnupg apt-transport-https
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-
ring:/usr/share/keyrings/wazuh.gpg --import && chmod 644 /usr/share/keyrings/wazuh.gpg
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt/ stable main" | tee -a
/etc/apt/sources.list.d/wazuh.list
apt-get update
```

Install Wazuh-Manager:

```
apt-get -y install wazuh-manager
systemctl daemon-reload
systemctl enable wazuh-manager
systemctl start wazuh-manager
```

Forward Logs to Graylog:

Set up Graylog Input

1. Access the Graylog WebUI and go to System->Inputs.
2. Initiate a fresh Raw/Plaintext TCP input.
3. Keep the default settings unchanged and choose to save. Graylog is now ready to receive TCP messages on port 5555.

Set Up Fluent-Bit on Wazuh Manager

1. The Wazuh Agent gathers logs from endpoints and transmits them to the Manager.
2. The Manager checks the incoming logs against its set of rules. If a match is found, the log is recorded in the /var/ossec/logs/alerts/alerts.json file.
3. Fluent Bit reads the contents of the alerts.json file and forwards its entries to the designated Graylog input.

```
curl https://raw.githubusercontent.com/fluent/fluent-bit/master/install.sh | sh
```

Edit the /etc/fluent-bit/fluent-bit.conf to collect the alerts.json file and send it to Graylog:

```
[SERVICE]
    flush        5
    daemon       Off
    log_level    info
    parsers_file parsers.conf
    plugins_file plugins.conf
    http_server  Off
    http_listen  0.0.0.0
    http_port    2020
    storage.metrics on
    storage.path /var/log/flb-storage/
    storage.sync normal
    storage.checksum off
    storage.backlog.mem_limit 5M
    Log_File /var/log/td-agent-bit.log
[INPUT]
    name  tail
    path  /var/ossec/logs/alerts/alerts.json
    tag wazuh
    parser  json
    Buffer_Max_Size 5MB
    Buffer_Chunk_Size 400k
    storage.type      filesystem
    Mem_Buf_Limit     512MB
[OUTPUT]
    Name  tcp
    Host  *your graylog host*
    Port  *your graylog port*
    net.keepalive off
    Match wazuh
    Format  json_lines
    json_date_key true
```

```
systemctl enable fluent-bit
systemctl start fluent-bit
```

b. Install TheHive

Java Virtual Machine:

```
apt-get install -y openjdk-8-jre-headless
echo JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64" >> /etc/environment
export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
```

Cassandra Database:

Apache Cassandra is a database that offers scalability and high availability. TheHive is compatible with the most recent stable release, version 3.11.x, of Cassandra.

```
1.      Add Apache repository references
curl -fsSL https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
echo "deb http://www.apache.org/dist/cassandra/debian 311x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

2.      Install the package
sudo apt update
sudo apt install cassandra
```

Data is typically saved in the /var/lib/cassandra directory as the default location.

Cassandra is the backend database which TheHive will write to.

Make some configuration tweeks to create a Cassandra cluster and enable TheHive to write to it.

Configuration:

```
Change the cluster_name with thp. Run the command cqlsh:
cqlsh localhost 9042
cqlsh> UPDATE system.local SET cluster_name = 'thp' where key='local';
Exit and then run:
nodetool flush
```

Edit the /etc/cassandra/cassandra.yaml file.

```
# content from /etc/cassandra/cassandra.yaml

cluster_name: 'thp'
listen_address: 'xx.xx.xx.xx' # address for nodes
rpc_address: 'xx.xx.xx.xx' # address for clients
seed_provider:
    - class_name: org.apache.cassandra.locator.SimpleSeedProvider
      parameters:
          # Ex: "<ip1>,<ip2>,<ip3>"
          - seeds: 'xx.xx.xx.xx' # 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'
```

Restart the Cassandra service

```
service cassandra restart
```

Storing Files

To save files on the local file system, begin by selecting a specific directory: Create the directory by running:

```
mkdir -p /opt/thp/thehive/files
```

This chosen path will be applied in TheHive 's configuration. Once TheHive is installed, it's important to grant ownership of the designated file storage path to the user "TheHive ": Make sure to execute:

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

Install TheHive

```
curl https://raw.githubusercontent.com/TheHive-Project/TheHive/master/PGP-PUBLIC-KEY | sudo apt-key add -
echo 'deb https://deb.thehive-project.org release main' | sudo tee -a /etc/apt/sources.list.d/thehive-project.list
sudo apt-get update
sudo apt-get install thehive4
```

Indexing Engine

Devote a specific directory to house TheHive 's indexes: Generate a dedicated directory for TheHive 's indexes:

```
mkdir /opt/thp/thehive/index
```

Ensure proper ownership of the designated index directory: Apply the correct ownership using the command:

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

Changes to TheHive configuration file:

Database
In order to utilize the Cassandra database, it is necessary to modify and enhance the content of TheHive 's configuration file (/etc/TheHive /application.conf) with the subsequent lines:

```
db {
  provider: janusgraph
  janusgraph {
    storage {
      backend: cql
      hostname: ["<Cassandra server 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
      }
    }
  }
}
```

Filesystem

```
If you chose to store files on the local filesystem:

Ensure permission of the folder
chown -R thehive:thehive /opt/thp/thehive/files

add following lines to TheHive configuration file (/etc/thehive/application.conf)

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

Execute the following steps:

1. Save the modified configuration file.

2. Initiate the service by running:

```
service thehive start
```

Kindly be aware that the service might require a certain duration to initiate. After it has successfully started, you can open your web browser and access http://YOUR_SERVER_ADDRESS:9000/.

The initial admin user is admin@TheHive .local, and the password is set to 'secret'. It is advisable to modify this default password for security reasons [6].

c. Install Cortex

Prerequisite for installing cortex is to install elasticsearch

Install the Elasticsearch package provided by Elastic:

```
# PGP key installation
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-key D88E42B4

# Alternative PGP key installation
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

# Debian repository configuration
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list

# Install https support for apt
sudo apt install apt-transport-https

# Elasticsearch installation
sudo apt update && sudo apt install elasticsearch
```

Edit /etc/elasticsearch/elasticsearch.yml and add the following lines:

```
http.host: <server-ip>
cluster.name: hive
thread_pool.search.queue_size: 100000
```

Start the Service

```
sudo systemctl enable elasticsearch.service
sudo systemctl start elasticsearch.service
sudo systemctl status elasticsearch.service
```

Install Cortex:
The release repository includes packages for Cortex version 3.1.0 and above. Configure the apt settings to use the release repository:

- Retrieve the PGP public key
- Add the repository to the sources list
- Update the package information

```
curl https://raw.githubusercontent.com/TheHive-Project/TheHive/master/PGP-PUBLIC-KEY | sudo apt-key add -
echo 'deb https://deb.thehive-project.org release main' | sudo tee -a /etc/apt/sources.list.d/thehive-project.list
sudo apt-get update
```

After completing these steps, you can utilize the apt command to install Cortex version 3.1.0+:

```
apt install cortex
```

The primary essential parameter for initiating Cortex is the server key (play.http.secret.key). This key is employed to authenticate cookies containing data. If Cortex operates in cluster mode, all instances must share the same key. You can generate the basic configuration using the following commands (assuming you have created a dedicated user named cortex for Cortex):

- Create a directory for the Cortex configuration.
- Insert the following lines to generate the secret key and add it to the application.conf file.

```
sudo mkdir /etc/cortex
(cat << _EOF_
# Secret key
# ~~~~~~
# The secret key is used to secure cryptographics functions.
# If you deploy your application to several instances be sure to use the same key!
play.http.secret.key="$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 64 | head -n 1)"
_EOF_
) | sudo tee -a /etc/cortex/application.conf
```

After adding the secret key, initiate the Cortex service and enable it to start on boot: Add the Cortex service to system startup:

```
sudo systemctl enable cortex
```

Start the Cortex service:

```
sudo service cortex start
```

Installing Analyzers and Responders

Presently, all analyzers and responders supported by TheHive Project are scripted in either Python 2 or 3. They do not require a build phase, but their dependencies need to be installed. Before proceeding, it is necessary to install system package dependencies that some of them rely on: Execute the following command to install the required system package dependencies:

```
sudo apt-get install -y --no-install-recommends python-pip python2.7-dev python3-pip python3-dev ssdeep libfuzzy-dev libfuzzy2
libimage-exiftool-perl libmagic1 build-essential git libssl-dev
```

Ensure that pip and setuptools are up to date by running these commands:

```
sudo pip install -U pip setuptools && sudo pip3 install -U pip setuptools
```

Clone the Cortex-analyzers repository using the following command:

```
git clone https://github.com/TheHive-Project/Cortex-Analyzers
```

Install the requirements for all analyzers using these commands:

```
for I in $(find Cortex-Analyzers -name 'requirements.txt'); do sudo -H pip2 install -r $I; done &&
for I in $(find Cortex-Analyzers -name 'requirements.txt'); do sudo -H pip3 install -r $I || true; done
```

In the Cortex configuration file (application.conf), provide the paths to the directories containing the analyzers and responders as shown below:

```
analyzer {
  # Directory that holds analyzers
  urls = [
    "https://download.thehive-project.org/analyzers.json",
    "/path/to/default/analyzers",
    "/path/to/my/own/analyzers"
  ]

  fork-join-executor {
    # Min number of threads available for analyze
    parallelism-min = 2
    # Parallelism (threads) ... ceil(available processors * factor)
    parallelism-factor = 2.0
    # Max number of threads available for analyze
    parallelism-max = 4
  }
}

responder {
  # Directory that holds responders
  urls = [
    "https://download.thehive-project.org/responders.json",
    "/path/to/default/responder",
    "/path/to/my/own/responder"
  ]

  fork-join-executor {
    # Min number of threads available for analyze
    parallelism-min = 2
    # Parallelism (threads) ... ceil(available processors * factor)
    parallelism-factor = 2.0
    # Max number of threads available for analyze
    parallelism-max = 4
  }
}
```

To Configure analyzers like AbuseIPDB and Virtutotal.

- Login to these websites and copy the API Key.
- Then in the analyzers option look for the analyzer (AbuseIPDB/Virustotal) and click edit.
- Add a name. Set cert_check to false.
- Enter the API key you copied and save [7].

d. Install MISP

Run the following script [8].

```
# Please check the installer options first to make the best choice for your install
wget --no-cache -O /tmp/INSTALL.sh https://raw.githubusercontent.com/MISP/MISP/2.4/INSTALL/INSTALL.sh
bash /tmp/INSTALL.sh

# This will install MISP Core
wget --no-cache -O /tmp/INSTALL.sh https://raw.githubusercontent.com/MISP/MISP/2.4/INSTALL/INSTALL.sh
bash /tmp/INSTALL.sh -c
```

e. Install Shuffle

To install Shuffle, follow these steps [9]:

1. Update your system's package information:

```
sudo apt get update
```

2. Upgrade installed packages to their latest versions:

```
sudo apt upgrade
```

3. Install Docker and Docker Compose:

```
sudo apt install docker.io
sudo apt install docker-compose
```

4. Download the Shuffle repository:

```
git clone https://github.com/frikky/Shuffle
cd Shuffle
```

5. Run Docker Compose:

```
sudo docker-compose up -d
```

Please wait until the process completes; this will result in the creation of the shuffle-database folder.

6. Adjust prerequisites for the Opensearch database (Elasticsearch):

```
sudo chown 1000:1000 -R shuffle-database
```

7. Restart Docker Compose:

```
sudo docker-compose restart
```

4. Section 4: Integrating the tools

Integrating TheHive and Cortex.

- Login to Cortex, navigate to organization create a new user and provide org-admin role to this user. Now login with that user.
- In the new login, navigate to Organization and create a new API user and create an API key and copy this key.
- Now open TheHive application.conf file, edit the cortex config. Point it to the url cortex is listening on and enter the key copied from cortex.
- Restart TheHive

Integrating Cortex and MISP.

- Login to Cortex and navigate to Analyzers
- Look for MISP Analyzer
- Add a name
- URL point to the IP of MISP example: https://<server-ip>/>
- Add the API Key copied from MISP. Set cert_check to false and save.

To obtain key API Key for MISP:

- Login to MISP console, navigate to Administrator and then click on List Auth Key.
- Generate a new key by clicking on add authentication key.
- Enter the Allowed IPs as 0.0.0.0/0 and click submit.
- Copy the Auth Key Created.

5.Section 5: Creating the Shuffle Workflow

- Create a new workflow and give it a name and description.
- To receive alert from wazuh into shuffle we will create a trigger using the Webhook.

Our Wazuh manager will receive a log, it will then parse through that and evaluate it against its ruleset. If it matches a rule, then we will send it over to our webhook in shuffle to trigger our rest of the workflow.

- Click on the triggers tab and select Webhook, name it receive wazuh alert.
- The next node (change me) must repeat the contents that are received via the webhook that triggers this workflow. So in the cell erase everything and click on the plus icon and select Execution Argument. Save this.
- Configure Wazuh manager to send the alerts to the Webhook. To do this navigate to Shuffle's github https://github.com/Shuffle/Shuffle/tree/main/functions/extensions/wazuh and copy custom-shuffle and custom-shuffle.py scripts to our Wazuh manager.
- Edit the custom-shuffle.py and add the tag verify=false as the webhook is using https, we are using self-signed cert and wazuh manager will complain that it does not trust the cert. Save this file and change the permission and ownership to ossec.
- Edit the ossec.conf file to add a block for Shuffle with the webhook url. Save this file and restart the wazuh manager.

Navigate to the executions tab and see that our alerts come into shuffle.

Then we add our apps to the workflow. The first app is TheHive .

- Click add authentication.
- Add the API key from TheHive .
- Add the url for TheHive .
- Add the organization for which you added the API Key.

Process to Create Alert:

- Now that we have added the authentication we name the Find Action as Create Alert. Tag is incident.
- In Source reference call change me and add the timestamp.
- Add Title as SourceIP call change me and add the source IP and rule description.
- Now save this.

To notify in a Microsoft teams channel:

- Login to your teams and create a channel for your team.
- Add team members and name this channel.
- Now in the settings for this channel you will find the webhook, copy this value.
- Add an app called Microsoft teams, in the authentication tab add a name and in the URL add the webhook you copied.
- Add the Find Actions as Send simple text.
- In Message select the auto complete and click change me and specify the rule name and rule description.

To add Artifact add another app of TheHive and name it Get Artifact.

- Actions Select Add Alert artifact
- Specify the alert id by clicking create alert node and select alert id.
- Data type set to IP.
- Data click change me and add the source IP. This will add the IP as an observable to our alert in TheHive.

To analyse this alert, we will add another app of http and name it creates case.

- Find Action is set to curl
- Add this Curl command in the statement.

```
curl -XPOST -H 'Authorization: Bearer **YOUR API KEY**' http://192.169.200.39:9000/api/alert/$create_alert.id/createCase
```

To run analysis against cortex we add another app of TheHive and name it gets artefact id.

- Select the Find action as Get case artifacts.
- Specify the case id by clicking create case node and select alert id.
- Select DataType as IP

To automate the running of our cortex job we add another app of TheHive and name it.

- Select the Find action as Run analyzer.
- Specify the Cortex id which is the value you see when you connect cortex to TheHive , paste the value.
- Add the Analyzer id from TheHive under observables click on run analyzer and copy the value of the analyser you see there. Add the value in the analyser id field.
- For Artifact id go to the autocomplete select get artefact id and select the id and save this.

This workflow will pull the alerts from Wazuh and create a case in TheHive.

It will also post a message in Microsoft Teams channel.
Then it will get the artefact and create a case for us in TheHive.
The cortex which is already integrated with TheHive will run analysis on the artifact based on the analyser configured. And add the reputation report in TheHive case.
This will help our analysts to detect and respond to alerts quickly.

References

[1] Digininja, “Damn Vulnerable Web Application,” Github, [Online]. Available: https://github.com/digininja/DVWA#linux-packages.

[2] V. Obhan, “Detecting web attacks using Wazuh and teler,” Wazuh Blog, 28 October 2023. [Online]. Available: https://wazuh.com/blog/detecting-web-attacks-using-wazuh-and-teler/.

[3] Wazuh, “Installing the Wazuh server step by step,” Wazuh Inc., 2023. [Online]. Available: https://documentation.wazuh.com/current/installation-guide/wazuh-server/step-by-step.html.

[4] T. Walton, “SOCFortress Knowledge Base Integration,” SOCFortress, [Online]. Available: https://github.com/socfortress/ASK-SOCFortress-Module.

[5] T. Walton, “Installing the New Wazuh version 4.4 — The SOCFortress Way,” SocFortress, May 2022. [Online]. Available: https://socfortress.medium.com/installing-the-new-wazuh-version-4-4-the-socfortress-way-ea3a8030d94b.

[6] TheHive-Project, “TheHive Project Documentation,” TheHive-Project, 02 June 2021. [Online]. Available: https://docs.thehive-project.org/thehive/installation-and-configuration/installation/step-by-step-guide/.

[7] J. Leonard, “CortexDocs,” TheHive-Project, 06 September 2022. [Online]. Available: https://github.com/TheHive-Project/CortexDocs/tree/master.

[8] MISP, “MISP,” MISP Project, [Online]. Available: https://misp.github.io/MISP/xINSTALL.ubuntu2204.

[9] Frikky, “Getting started with Shuffle — an Open Source SOAR platform part 2,” Shuffle Automation, 26 May 2020. [Online]. Available: https://medium.com/shuffle-automation/getting-started-with-shuffle-an-open-source-soar-platform-part-2-1d7c67a64244.