

SOC Automation Lab on Microsoft Azure

Automated Detection & Incident Response Using Wazuh, TheHive, and Shuffle

1. Project Overview

This project implements a cloud-based SOC automation lab on Microsoft Azure, designed to demonstrate a complete detection → enrichment → case management → response workflow.

The lab integrates:

- **Wazuh** as the SIEM platform for log collection and detection
- **TheHive** as the incident response and case management system
- **Shuffle** as the SOAR platform for automation and orchestration
- **VirusTotal** as the enrichment technology

To reflect realistic SOC architecture, the environment is deployed using two separate Azure Virtual Machines, isolating SIEM operations from incident response tooling. Telemetry is generated from a Windows 11 endpoint running on VirtualBox, with attack simulation performed using Mimikatz.

Key skills demonstrated in this lab:

- **SOC Architecture Design**
Designed a two-VM SOC architecture on Azure, separating SIEM and incident response components to reflect real-world SOC deployment practices.
- **SIEM & SOAR Integration**
Integrated Wazuh with Shuffle to automatically forward alerts, orchestrate workflows, and trigger downstream actions.
- **Detection Engineering**
Configured Sysmon log ingestion and custom Wazuh rules to detect malicious activity such as Mimikatz execution and process renaming.
- **Incident Response Automation**
Automated alert enrichment, case creation in TheHive, analyst notification, and controlled response actions using Shuffle.

- **Azure Cloud Security & Networking**
Secured cloud infrastructure using Azure Network Security Groups, IP-restricted access, and localhost-only bindings for backend services.
- **Linux Administration**
Installed, configured, and troubleshooted Wazuh, Elasticsearch, Cassandra, and TheHive services on Ubuntu servers.
- **Regex-Based Data Parsing**
Used regex within Shuffle workflows to extract indicators such as SHA-256 hashes from SIEM alerts for automated processing.
- **Threat Intelligence Integration**
Integrated VirusTotal to enrich alerts with file reputation data before case creation and analyst review.

2. Architecture Overview

This diagram illustrates the end-to-end SOC automation workflow, focusing on how security events flow between detection, enrichment, case management, and analyst response. Each component represents a specific functional role within the SOC automation architecture.

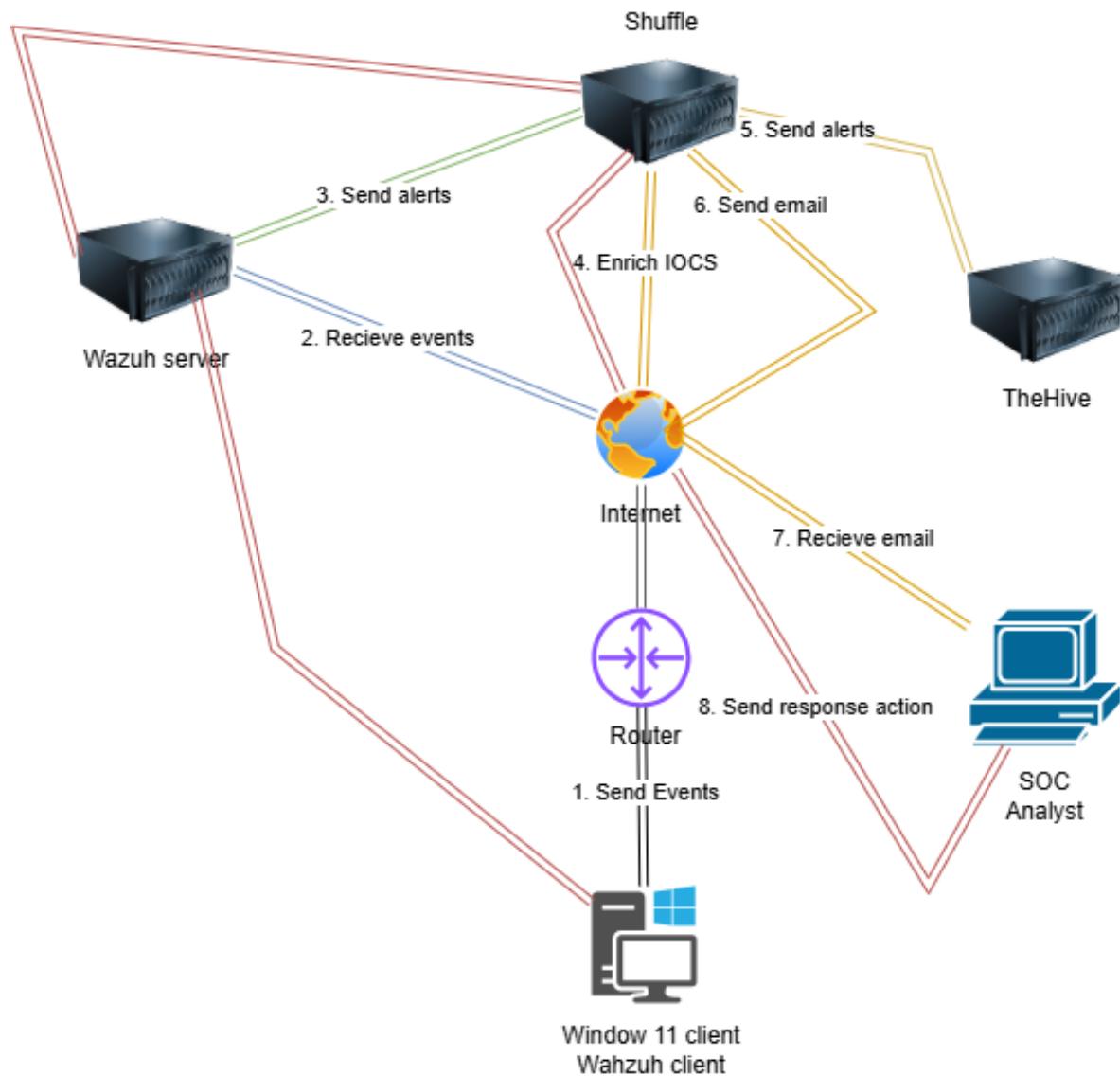


Figure 1

List of components:

- Windows 11 Client (Wazuh Agent):
 - Acts as the monitored endpoint. It generates security telemetry and attack-related events (e.g., Mimikatz execution) which are collected by the Wazuh agent.
- Router:
 - Represents the network path that enables communication between the endpoint, cloud-hosted SOC components, and the internet.
- Internet (Cloud):
 - Serves as the communication medium connecting on-premise or local endpoints with cloud-based SOC infrastructure and external threat intelligence services.
- Wazuh Server (SIEM):
 - Functions as the central log collection and detection engine. It receives events from the Windows 11 client, applies detection rules, and generates security alerts based on suspicious activity.
- Shuffle (SOAR Platform):
 - Orchestrates automated workflows. It receives alerts from Wazuh, enriches indicators of compromise (IOCs) using external threat intelligence sources, sends alerts to TheHive, and triggers analyst notifications.
- TheHive (Incident Response Platform):
 - Provides centralised case and alert management. Enriched alerts from Shuffle are converted into structured cases for investigation and tracking.
- SOC Analyst:
 - Represents the human analyst in the loop. The analyst receives email notifications for high-priority incidents, reviews cases in TheHive, and decides on appropriate response actions.

Workflow Representation in the Diagram

The numbered paths in the diagram represent the operational flow:

1. The Windows 11 client sends security events to the Wazuh server.
2. Wazuh receives and processes endpoint events.
3. Wazuh forwards alerts to Shuffle.
4. Shuffle enriches alerts with threat intelligence from external sources.
5. Shuffle sends enriched alerts to TheHive for case creation.

6. Shuffle sends email notifications to the SOC analyst.
7. The SOC analyst receives alerts and reviews incident details.
8. The analyst's response action is sent back through Shuffle and executed via Wazuh.

Design Rationale

- **Separation of concerns:** SIEM and IR components are isolated
- **Security:** Databases never exposed publicly
- **Scalability:** SIEM and IR layers can scale independently
- **Cloud-native:** Architecture aligns with Azure networking and security models

3. Azure Infrastructure Setup

3.1 Create Azure Virtual Machines

Create **two Ubuntu Linux VMs**:

Azure VM 1 – Wazuh SIEM Server

- OS: Ubuntu LTS
- Purpose: Wazuh Manager
- Public IP: Static
- Authentication: SSH key

Azure VM 2 – Incident Response Server

- OS: Ubuntu LTS
- Purpose: TheHive, Elasticsearch, Cassandra
- Public IP: Static
- Authentication: SSH key

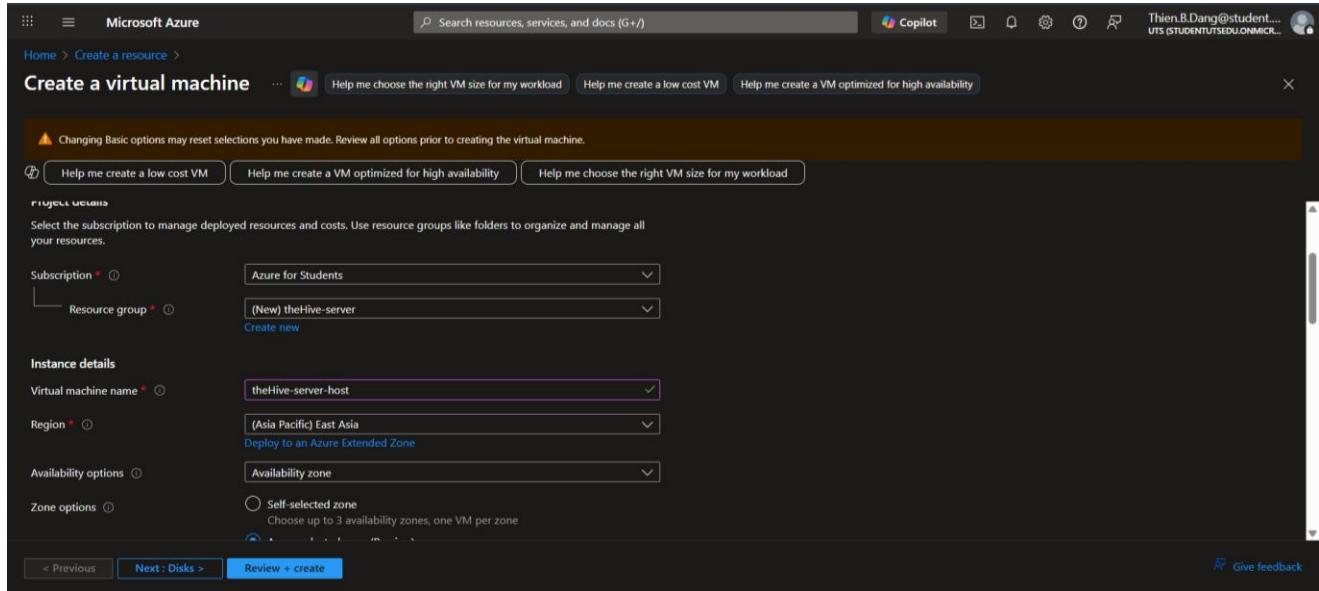


Figure 2

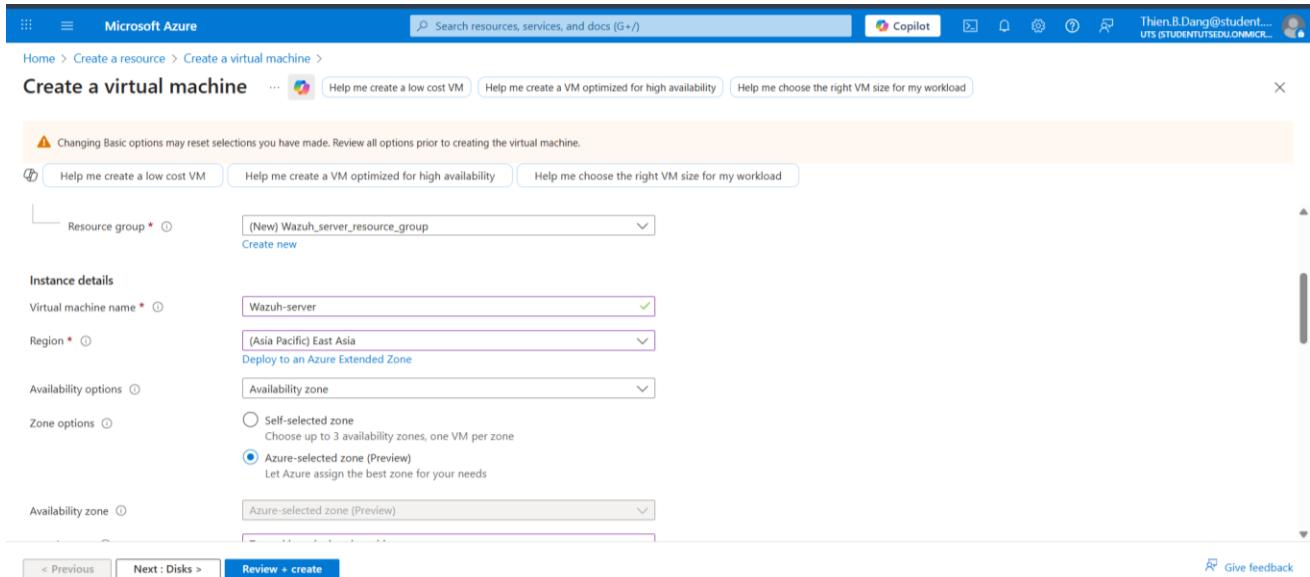


Figure 3

3.2 Configure Azure Network Security Groups (NSGs)

Each VM has **its own NSG**, attached at the **NIC level**.

NSG – Wazuh SIEM VM

Port	Purpose	Source
22	SSH	My IP Address
1514	Agent telemetry	Endpoint IP
1515	Agent registration	Endpoint IP

NSG – TheHive IR VM

Port	Purpose	Source
22	SSH	My IP Address
900	TheHive Web UI	My IP Address

=> Allowed TCP and UDP traffic only from my IP address for secure and limited access, as well as other needed network connections from components included in this lab scenario.

4. Endpoint Setup (Windows 11)

4.1 Virtual Machine Setup

- Platform: VirtualBox
- OS: Windows 11

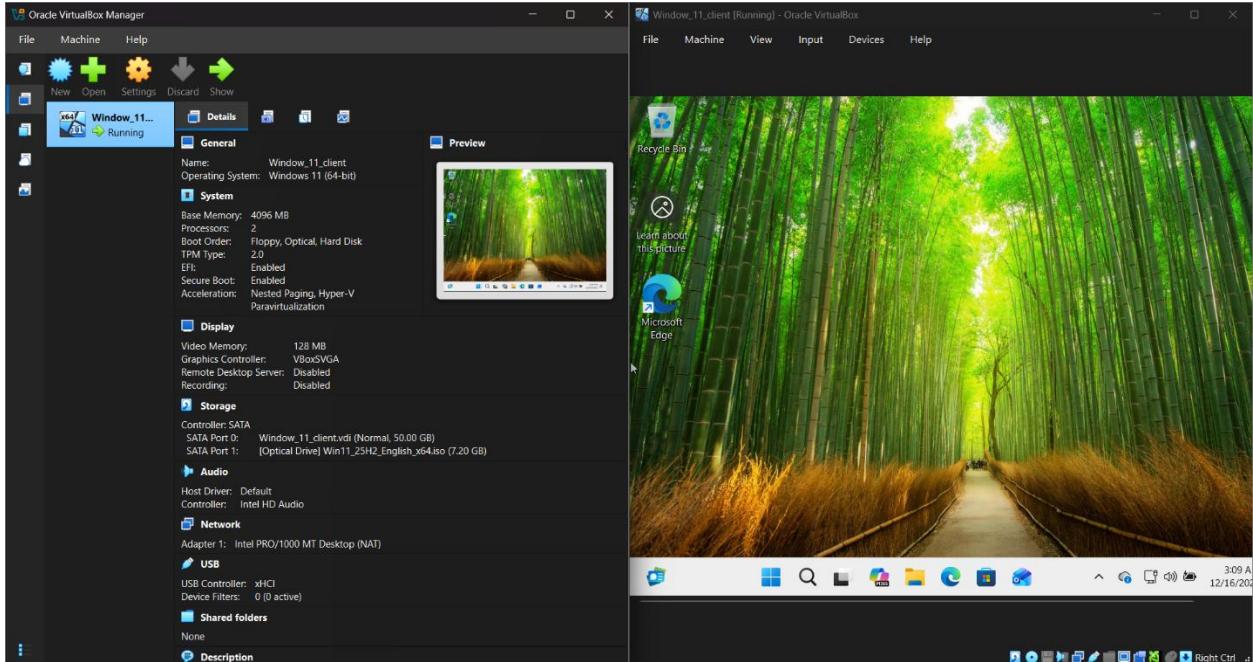
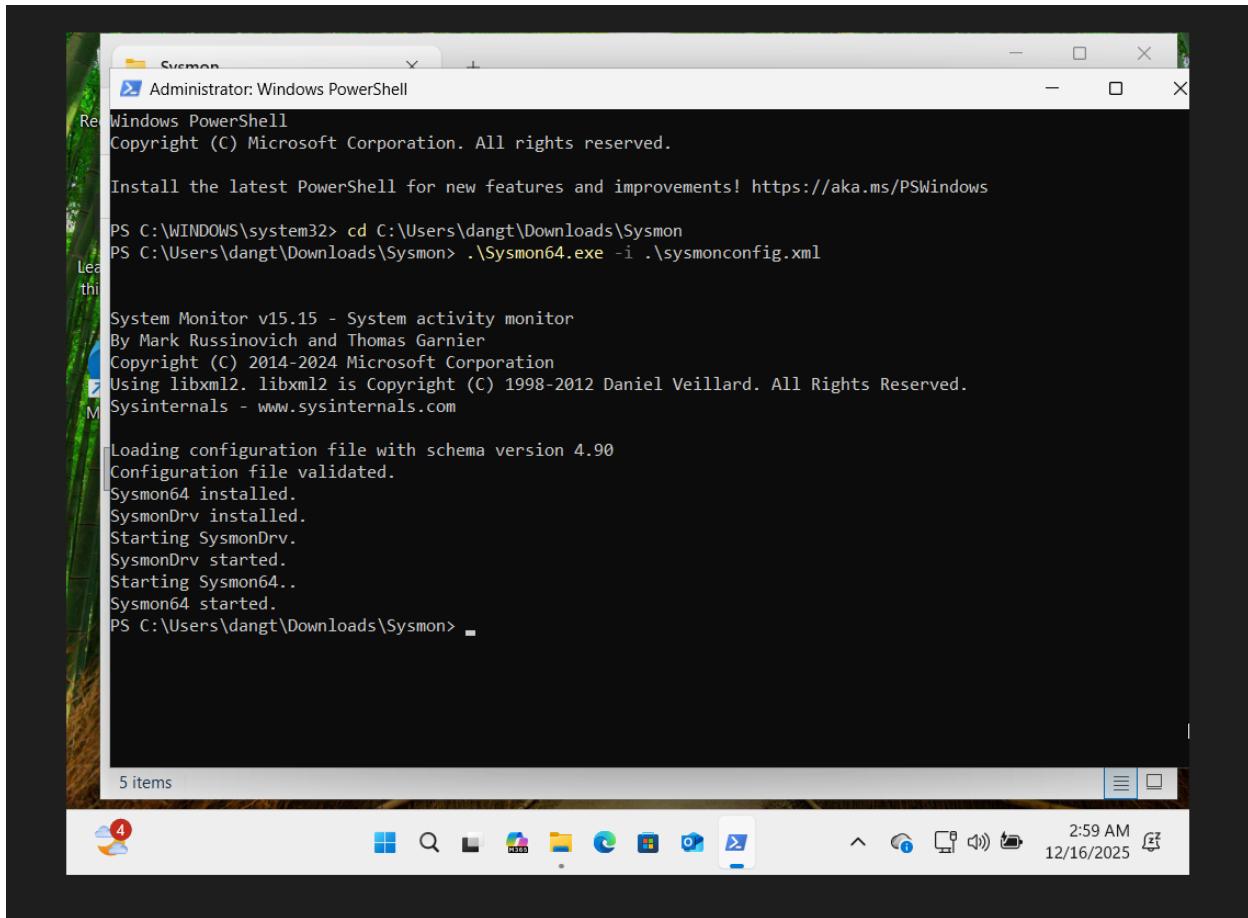


Figure 4

4.2 Install Sysmon

- Download Sysmon from Microsoft Sysinternals
- Install with a standard configuration
- Verify Sysmon service is running



The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell" running in a "Sysmon" folder. The command ".\Sysmon64.exe -i .\sysmonconfig.xml" is being run to install Sysmon. The output indicates the successful installation of Sysmon64, SysmonDrv, and Sysmon64.. The taskbar at the bottom shows several pinned icons and the system tray indicating the date and time.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> cd C:\Users\dangt\Downloads\Sysmon
PS C:\Users\dangt\Downloads\Sysmon> .\Sysmon64.exe -i .\sysmonconfig.xml

System Monitor v15.15 - System activity monitor
By Mark Russinovich and Thomas Garnier
Copyright (C) 2014-2024 Microsoft Corporation
Using libxml2. libxml2 is Copyright (C) 1998-2012 Daniel Veillard. All Rights Reserved.
Sysinternals - www.sysinternals.com

Loading configuration file with schema version 4.90
Configuration file validated.
Sysmon64 installed.
SysmonDrv installed.
Starting SysmonDrv.
SysmonDrv started.
Starting Sysmon64..
Sysmon64 started.
PS C:\Users\dangt\Downloads\Sysmon>
```

Figure 5

4.3 Install Wazuh Agent

- Download Wazuh Agent
- Register agent with Wazuh Manager (Azure VM 1)
- Confirm agent appears in Wazuh dashboard

5. Wazuh SIEM Deployment (Azure VM 1)

5.1 Install Wazuh

```
sudo apt update && sudo apt upgrade -y  
curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh  
sudo bash ./wazuh-install.sh -a
```

Save the generated credentials securely.

5.2 Agent Communication Model

- **Port 1515:** Agent registration and key exchange
- **Port 1514:** Encrypted telemetry, alerts, and active response

This separation improves security and reliability.

5.3 Configure Log Collection

On the **Windows agent**:

1. Open C:\Program Files (x86)\ossec-agent\ossec.conf
2. Enable Sysmon log collection
3. Reduce noise from low-value logs
4. Restart the Wazuh agent service

On the **Wazuh Manager**:

- Enable full logging:
- <log_all>yes</log_all>
- <log_all_json>yes</log_all_json>
- Restart Wazuh Manager

5.4 Detection Engineering (Mimikatz)

1. Download Mimikatz on the Windows endpoint
2. Exclude the directory from Windows Defender (lab only)
3. Execute mimikatz.exe in an elevated PowerShell session
4. Verify alerts appear in Wazuh

5. Rename the binary and re-execute to test rule robustness

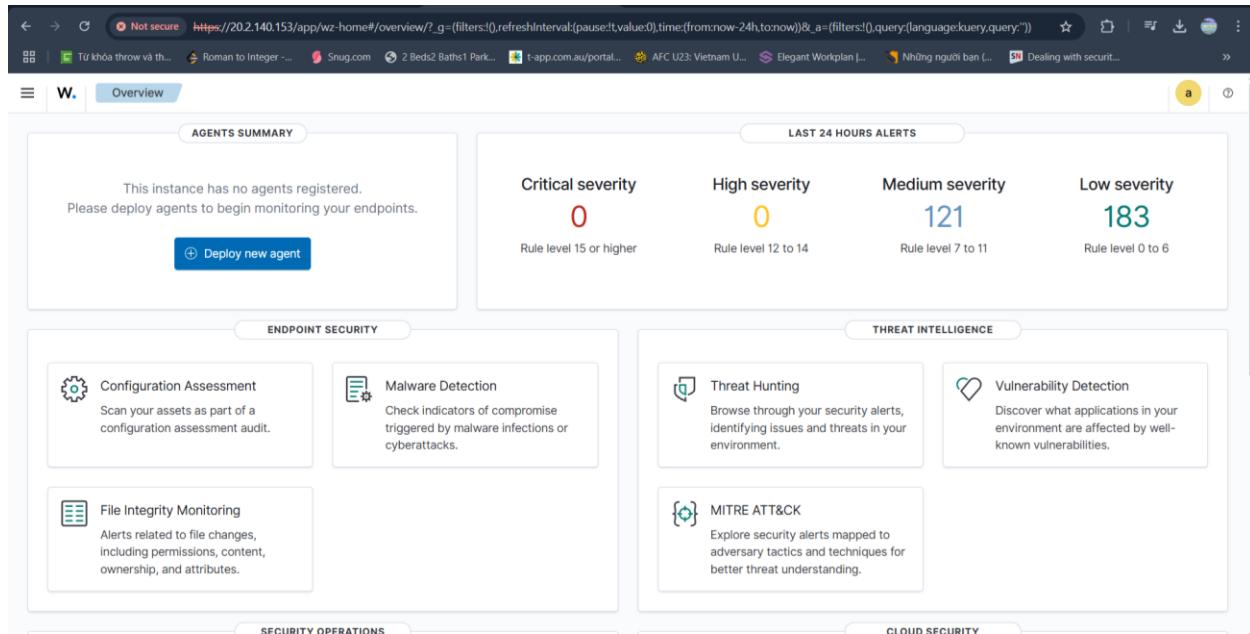
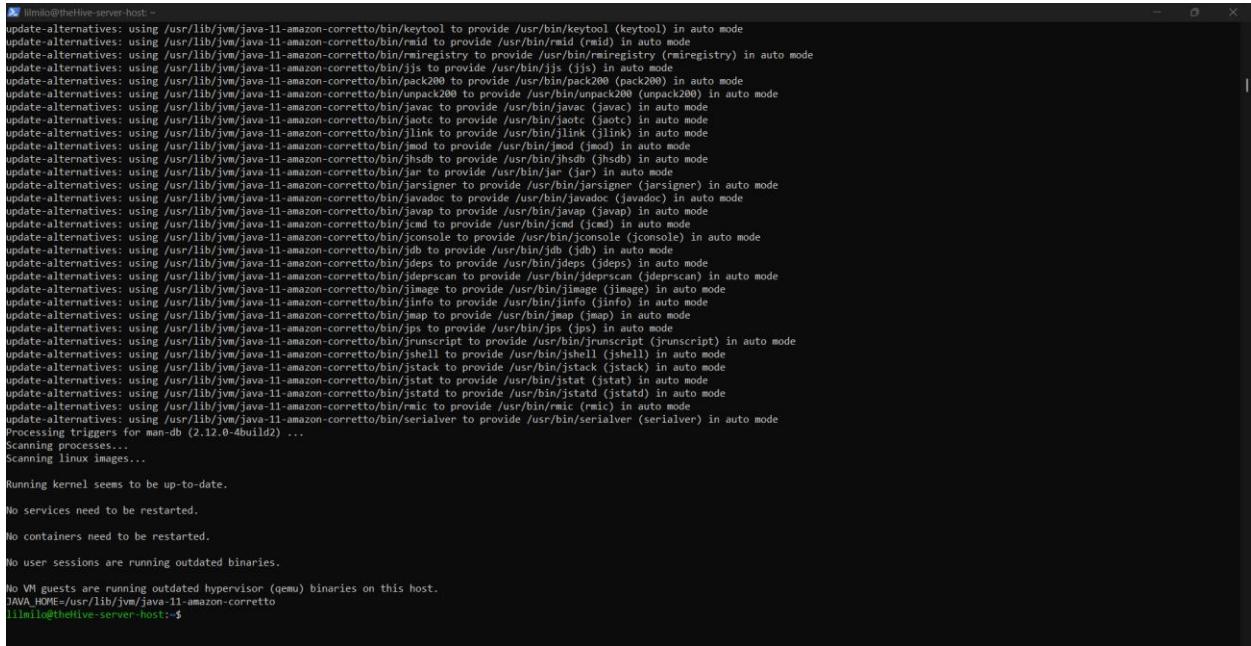


Figure 6

6. Incident Response Platform (Azure VM 2)

6.1 Install Dependencies

Installed essential dependencies, Java (Corretto 11), Cassandra, and Elasticsearch, following best practices for security and performance optimization.



```
lilimio@thehive-server-host: ~
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/keytool to provide /usr/bin/keytool (keytool) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/rmid to provide /usr/bin/rmid (rmid) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jjs to provide /usr/bin/jjs (jjs) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/pack200 to provide /usr/bin/pack200 (pack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/unpack200 to provide /usr/bin/unpack200 (unpack200) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/javac to provide /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jaotc to provide /usr/bin/jaotc (jaotc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jlink to provide /usr/bin/jlink (jlink) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jmod to provide /usr/bin/jmod (jmod) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jhsdb to provide /usr/bin/jhsdb (jhsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jar to provide /usr/bin/jar (jar) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jcmd to provide /usr/bin/jcmd (jcmd) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/javadoc to provide /usr/bin/javadoc (javadoc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jdeps to provide /usr/bin/jdeps (jdeps) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jdb to provide /usr/bin/jdb (jdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jdepscan to provide /usr/bin/jdepscan (jdepscan) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jimage to provide /usr/bin/jimage (jimage) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jinfo to provide /usr/bin/jinfo (jinfo) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jmap to provide /usr/bin/jmap (jmap) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jps to provide /usr/bin/jps (jps) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jscript to provide /usr/bin/jscript (jscript) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/jstard to provide /usr/bin/jstard (jstard) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/rmic to provide /usr/bin/rmic (rmic) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-amazon-corretto/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
JAVA_HOME=/usr/lib/jvm/java-11-amazon-corretto
lilimio@thehive-server-host:$
```

Figure 7

6.2 Configure Elasticsearch

cluster.name: thehive

node.name: node-1

network.host: 127.0.0.1

discovery.type: single-node

```

root@theLive-server: ~
GNU nano 7.2
/etc/elasticsearch/elasticsearch.yml *

# Make sure that the heap size is set to about half the memory available
# on the system and that the owner of the process is allowed to use this
# limit.

# Elasticsearch performs poorly when the system is swapping the memory.

# -----
# Network
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
network.host: 52.184.83.147
discovery.type: single-node
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
http.port: 9200
#
# For more information, consult the network module documentation.

# -----
# Discovery
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "::1"]
#
discovery.seed_hosts: ["host1"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
cluster.initial_master_nodes: ["node-1"]
discovery.type: single-node
#
# For more information, consult the discovery and cluster formation module documentation.

# -----
# Various
# Allow wildcard deletion of indices:
#
action.destructive_requires_name: false

# -----
# BEGIN SECURITY AUTO CONFIGURATION
# The following settings, TLS certificates, and keys have been automatically
# generated to configure Elasticsearch security features on 20-01-2026 13:53:56

[Read 1877 lines]

```

Figure 8

6.3 Configure Cassandra

Bind Cassandra to localhost only:

listen_address: 127.0.0.1

rpc_address: 127.0.0.1

```

root@theLive-server: ~
GNU nano 7.2
/etc/cassandra/cassandra.yaml

# Cassandra storage config YAML

# NOTE:
# See https://cassandra.apache.org/doc/latest/configuration/ for
# full explanations of configuration directives
# /NOTE

# The name of the cluster. This is mainly used to prevent machines in
# one logical cluster from joining another.
cluster_name: 'SOCAutomatic'

# This defines the number of tokens randomly assigned to this node on the ring
# The more tokens, relative to other nodes, the larger the proportion of data
# that this node will store. You probably want all nodes to have the same number
# of tokens assuming they have equal hardware capability.
#
# If you leave this unspecified, Cassandra will use the default of 1 token for legacy compatibility,
# and will use the initial_token as described below.

# Specifying initial_token will override this setting on the node's initial start,
# on subsequent starts, this setting will apply even if initial token is set.

# See https://cassandra.apache.org/doc/latest/getting_started/production.html#tokens for
# best practice information about num_tokens.
#
num_tokens: 16

# Triggers automatic allocation of num_tokens tokens for this node. The allocation
# algorithm attempts to choose tokens in a way that optimizes replicated load over
# the nodes in the datacenter for the replica factor.
#
# The load assigned to each node will be close to proportional to its number of
# vnodes.
#
# Only supported with the Murmur3Partitioner.

# Replica Factor is determined via the replication strategy used by the specified
# keyspace.
# allocate_tokens_for_keyspace: KEYSPACE

# Replica factor is explicitly set, regardless of keyspace or datacenter.
# This is the replica factor within the datacenter, like NTS.
# allocate_tokens_for_local_replication_factor: 3

[Read 1877 lines]

```

Figure 9

6.4 Install and Configure TheHive

1. Install TheHive 5
2. Generate Play secret:
3. openssl rand -base64 64
4. Configure /etc/thehive/application.conf:

```
play.http.secret.key = "<generated-secret>"
```

```
http {  
    enabled = true  
    address = "0.0.0.0"  
    port = 9000  
}
```

```
search {  
    uri = "http://127.0.0.1:9200"  
}
```

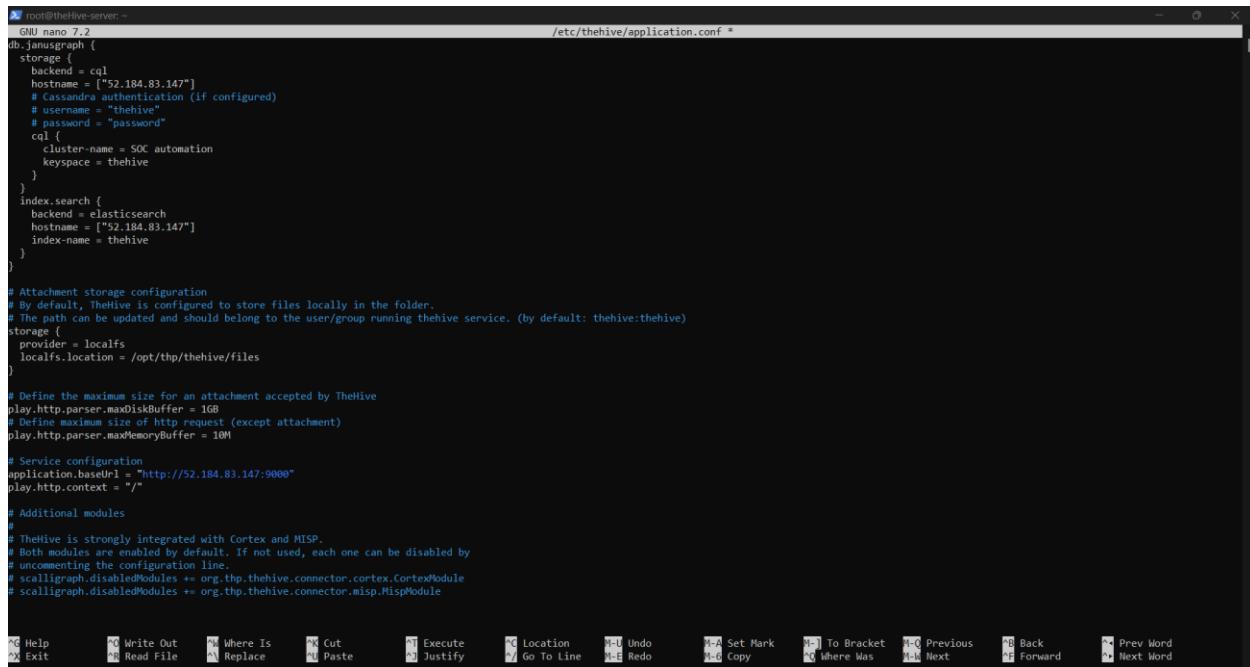
```
cassandra {  
    contact-points = ["127.0.0.1"]  
}
```

4. Start services in order:

```
systemctl restart cassandra  
sleep 20  
systemctl restart elasticsearch  
sleep 20  
systemctl restart thehive
```

5. Access:

http://<AZURE_PUBLIC_IP>:9000



```
root@thehive-server:~  
GNU nano 7.2  
db.janusgraph {  
    storage {  
        backend = cql  
        hostname = ["52.184.83.147"]  
        # Cassandra authentication (if configured)  
        # username = "thehive"  
        # password = "password"  
        cql {  
            cluster-name = SOC_automation  
            keyspace = thehive  
        }  
    }  
    index.search {  
        backend = elasticsearch  
        hostname = ["52.184.83.147"]  
        index-name = thehive  
    }  
}  
  
# Attachment storage configuration  
# By default, TheHive is configured to store files locally in the folder.  
# The path can be updated and should belong to the user/group running thehive service. (by default: thehive:thehive)  
storage {  
    provider = localfs  
    localfs.location = /opt/thp/thehive/files  
}  
  
# Define the maximum size for an attachment accepted by TheHive  
play.http.parser.maxDiskBuffer = 1GB  
# Define maximum size of http request (except attachment)  
play.http.parser.maxMemoryBuffer = 10M  
  
# Service configuration  
application.baseUrl = "http://52.184.83.147:9000"  
play.http.context = "/"  
  
# Additional modules  
#  
# TheHive is strongly integrated with Cortex and MISP.  
# Both modules are enabled by default. If not used, each one can be disabled by  
# uncommenting the configuration line.  
# scaligraph.disabledModules += org.thp.thehive.connector.cortex.CortexModule  
# scaligraph.disabledModules += org.thp.thehive.connector.misp.MispModule
```

Figure 10

7. SOAR Automation with Shuffle

7.1 Create Shuffle Workflow

- Create a new workflow named **SOC Automation Project**
- Add webhook trigger for Wazuh alerts

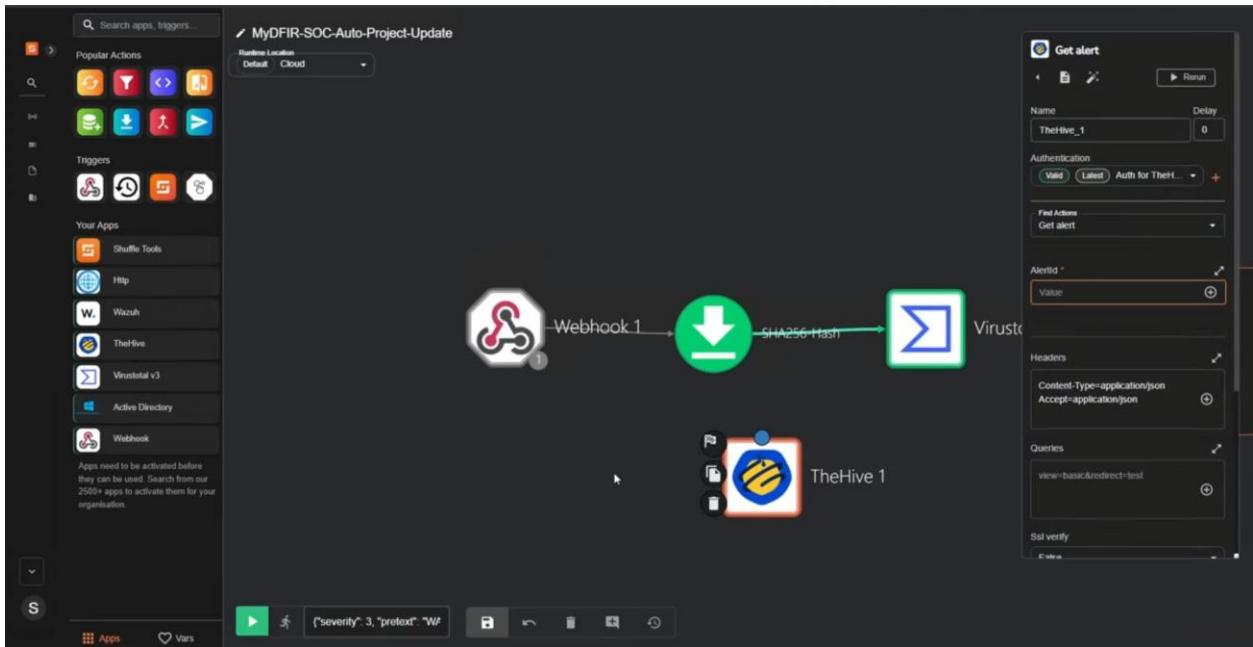


Figure 11

7.2 Integrate Wazuh with Shuffle

- Configure Wazuh to forward alerts to Shuffle webhook
- Test by generating sample alerts

7.3 Parsing & Enrichment

- Use regex to extract SHA-256 hashes
- Integrate VirusTotal for reputation checks
- Attach enrichment data to alerts

7.4 Integrate with TheHive

- Generate TheHive API key
- Configure Shuffle to create alerts in TheHive
- Map severity, summary, and observables

7.5 Analyst Interaction & Response

- Configure email notifications for high-severity alerts

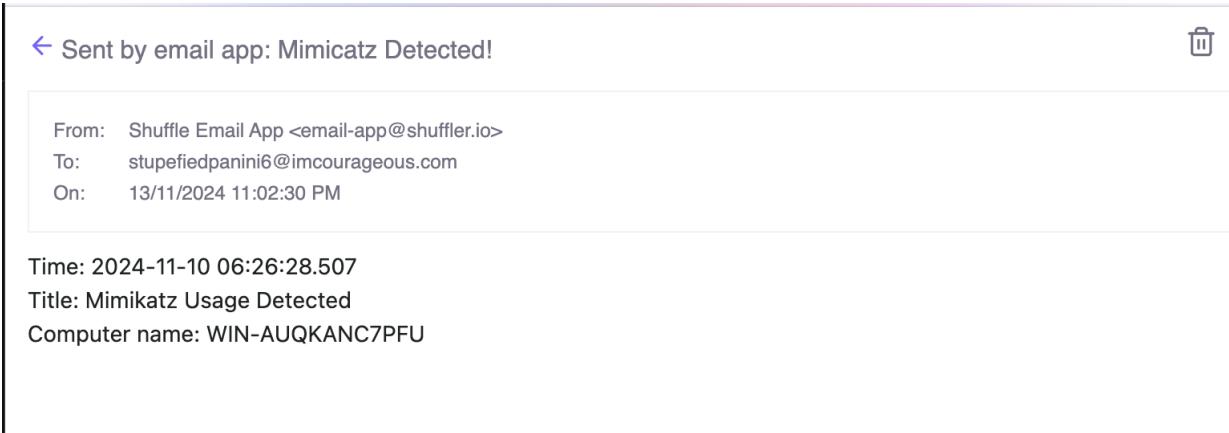


Figure 12

- Add analyst approval step for IP blocking
- Trigger Wazuh active response upon approval

```
rescue      reset-failed restart
[root@wazuh:~# systemctl restart wazuh-manager.service
[root@wazuh:~# cd /var/ossec/bin
[root@wazuh:/var/ossec/bin# ls
agent_control    manage_agents      wazuh-apid      wazuh-db          wazuh-logtest      wazuh-regex
agent_groups     rbac_control       wazuh-authd     wazuh-dbd         wazuh-logtest-legacy  wazuh-remoted
agent_upgrade    verify-agent-conf  wazuh-clusterd   wazuh-execd       wazuh-maild       wazuh-reportd
clear_stats       wazuh-agentlessd   wazuh-control    wazuh-integratord  wazuh-modulesd    wazuh-syscheckd
cluster_control  wazuh-analysisd   wazuh-csyslogd   wazuh-logcollector  wazuh-monitord
[root@wazuh:/var/ossec/bin# ./agent_control

Wazuh agent_control: Control remote agents.
Available options:
  -h           This help message.
  -l           List available (active or not) agents.
  -lc          List active agents only.
  -ln          List disconnected agents only.
  -i <id>      Extracts information from an agent.
  -R -a        Restart all agents.
  -R -u <id>    Restart the specified agent.
  -r -a        Runs the integrity/rootkit checking on all agents now.
  -r -u <id>    Runs the integrity/rootkit checking on one agent now.

  -s           Changes the output to CSV (comma delimited).
  -j           Changes the output to JSON .
Available options for active response:
  -b <ip>      Blocks the specified ip address.
  -f <ar> -a     Used with -b, specifies which response to run. Apply AR on all agents.
  -f <ar> -u <id> Used with -b, specifies which response to run. Apply AR on specified agent.
  -L           List available active responses.
[root@wazuh:/var/ossec/bin# -L
-L: command not found
[root@wazuh:/var/ossec/bin# ./agent_control -L

Wazuh agent_control. Available active responses:

  Response name: firewall-drop0, command: firewall-drop
[root@wazuh:/var/ossec/bin# ./agent_control -b 8.8.8.8. -f firewall-drop0 -u 001
```

Figure 13

8. Testing & Validation

End-to-end validation steps:

1. Execute Mimikatz on Windows endpoint
2. Detect alert in Wazuh
3. Forward alert to Shuffle
4. Enrich via VirusTotal
5. Create case in TheHive
6. Notify analyst
7. Execute response action

9. Conclusion

This SOC Automation Lab demonstrates a realistic, cloud-native approach to modern security operations. By separating SIEM and incident response workloads across two Azure Virtual Machines, the project reflects enterprise SOC design principles while showcasing automation, detection, and response capabilities applicable to real-world cybersecurity roles.