Cybersecurity SOC Automation Lab

Objectives

The primary goal of this project was to develop a resilient, automated Security Operations Center (SOC) capable of bridging the gap between raw telemetry (log) and actionable intelligence. The specific objectives were:

- Infrastructure Modernization: Rebuild an end-to-end SOC stack using the latest stable versions of open-source tools to resolve technical obsolescence issues identified in legacy community guides.
- Endpoint Visibility: Establish high-fidelity monitoring on a Windows 11 endpoint by instrumenting Sysmon to capture deep process-level artifacts (Event ID 1).
- Detection Engineering: Develop and validate custom SIEM rules in **Wazuh** to identify advanced adversary techniques, specifically credential dumping (T1003) via the **Mimikatz** toolset.
- SOAR Orchestration: Architect a **Shuffle** workflow to automate the enrichment of indicators of compromise (IOCs) using the **VirusTotal API**.
- Incident Lifecycle Management: Integrate **TheHive** to automate case creation, ensuring that high-severity alerts are immediately presented to analysts with full context and external threat intelligence.
- Response Optimization: Reduce the **Mean Time to Response (MTTR)** by implementing an automated notification loop via SMTP, ensuring 24/7 alert awareness without constant dashboard monitoring.

This report documents the engineering of a security automation pipeline designed to identify high-fidelity endpoint threats—specifically credential dumping—and execute a fully orchestrated response. By integrating **Wazuh** (SIEM), **Shuffle** (SOAR), and **TheHive** (Case Management), the project successfully demonstrates a reduction in response latency. The system validates the entire incident lifecycle: from raw telemetry (log) ingestion to threat intelligence enrichment and automated analyst notification.

Infrastructure & System Design

The laboratory environment was designed to mirror a production enterprise environment, utilizing a distributed model to balance high-resource database operations with lightweight endpoint monitoring.

Component	Technology	Role
Endpoint	Windows 11 / Sysmon 15.15	Telemetry Source & Attack Surface
SIEM	Wazuh Manager 4.x	Log Aggregation & Detection Engine

SOAR	Shuffle	Workflow Orchestration & API Middleware
Case Management	TheHive 5.x	Incident Tracking & Analyst Interface
Databases	Cassandra & Elasticsearch	Distributed Storage & Search
Intelligence	VirusTotal API	Threat Indicator Enrichment

Implementation Phase

Phase I: Infrastructure Hardening and Backend Integration

Establishing a scalable security backbone for the SOC is a common challenge, as many tools fail to communicate or consume excessive memory. By moving from a local setup to a public-facing IP cluster, a professional environment was created where the case manager has instant, reliable access to its search engine and database. This eliminates data silos and ensures that when an analyst opens a case, the data is indexed and searchable in milliseconds, providing the speed needed to catch a fast-moving attacker.

Phase II: High-Fidelity Log and Data Sanitization

Gaining high-fidelity visibility into the Windows environment without being blinded by data is the next priority. Standard Windows logs are often too vague to catch advanced threats, but by using a precision-tuned monitoring configuration, the system captures the critical footprints left behind—such as process creation and network connections—while ignoring harmless background noise. This prevents alert fatigue, ensuring analysts only see high-value events. It is the difference between a security camera that records every leaf blowing in the wind versus one that only alerts when a person enters the building.

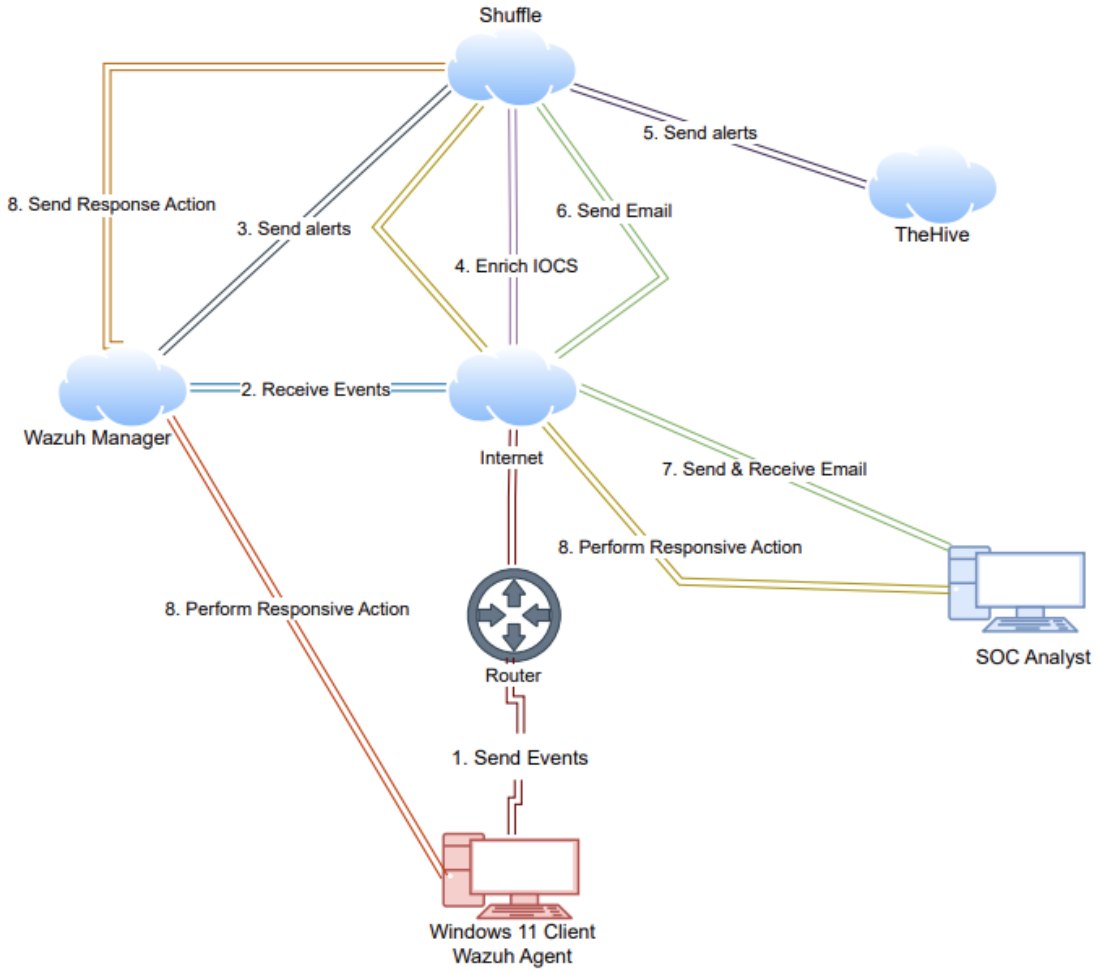
Phase III: Behavioral Detection Engineering

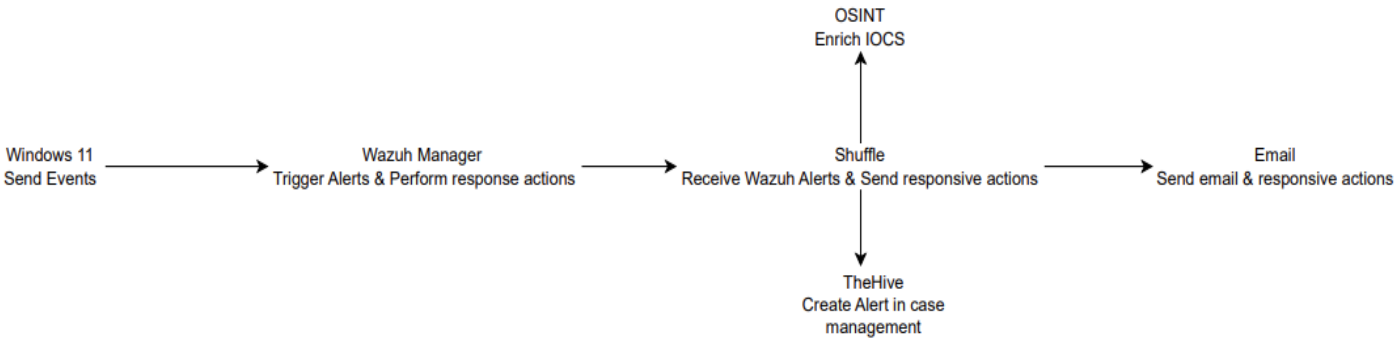
Turning raw data into actionable intelligence is where proactive defense begins. Attackers often hide by renaming their tools to blend in with legitimate processes. By analyzing the raw DNA of an attack, a custom detection rule was crafted to identify the internal identity of a tool rather than just its name. This moves the SOC from a reactive state to a proactive one, where the system hunts for specific adversary behaviors, ensuring even stealthy attacks are flagged immediately.

Phase IV: Orchestrated Incident Response and Automation

Finally, implementing an automation force multiplier replaces 30 minutes of manual work with seconds of automated response. In a traditional SOC, an analyst must manually copy file hashes, check threat intelligence, create tickets, and send notifications. An automated workflow now performs these repetitive tasks instantly, even accounting for technical glitches through refined logic. This dramatically lowers the Mean Time to Response (MTTR), freeing analysts to focus on strategic defense rather than manual triage.

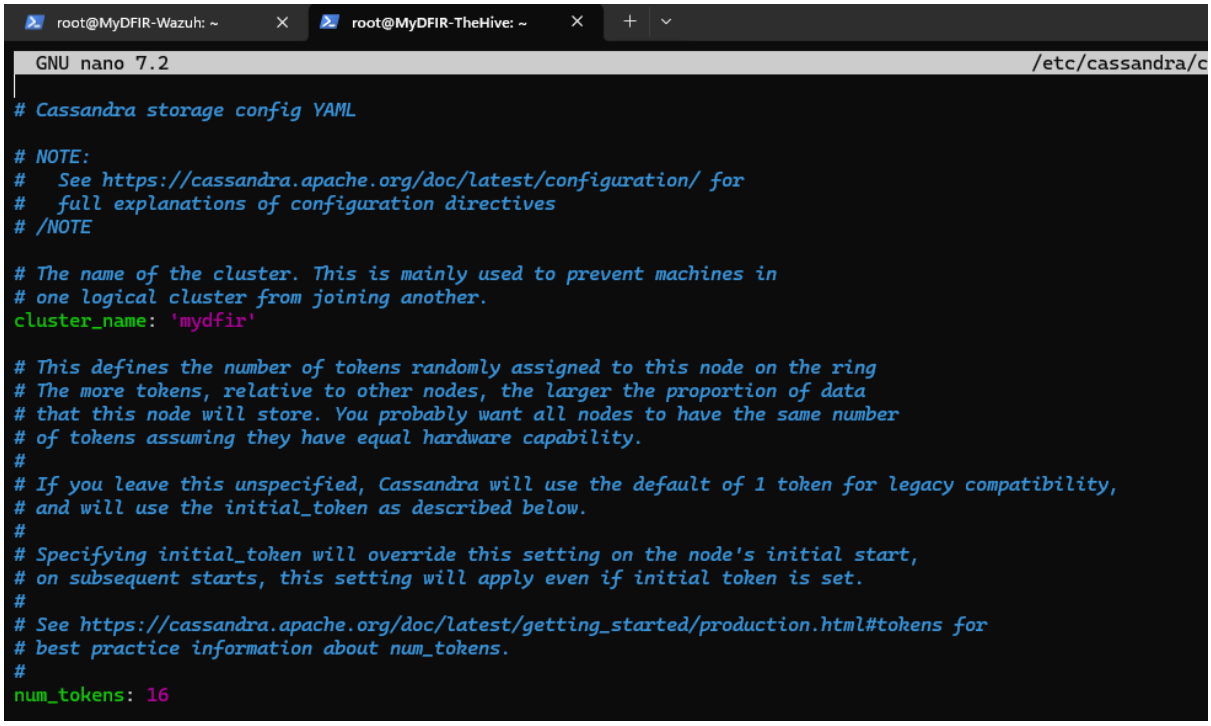
System Architecture





```
root@MyDFIR-TheHive:~# nano /etc/cassandra/cassandra.yaml
root@MyDFIR-TheHive:~# systemctl stop cassandra.service
root@MyDFIR-TheHive:~# rm -rf /var/lib/cassandra/*
root@MyDFIR-TheHive:~# systemctl status cassandra.service
○ cassandra.service - LSB: distributed storage system for structured data
     Loaded: loaded (/etc/init.d/cassandra; generated)
     Active: inactive (dead) since Fri 2026-01-16 22:07:34 UTC; 52s ago
   Duration: 23h 56min 46.930s
       Docs: man:systemd-sysv-generator(8)
    Process: 935 ExecStart=/etc/init.d/cassandra start (code=exited, status=0/SUCCESS)
    Process: 10503 ExecStop=/etc/init.d/cassandra stop (code=exited, status=0/SUCCESS)
        CPU: 22min 15.401s
```

The initial interface of The Hive server confirms successful installation and web service accessibility, representing the foundational layer upon which the entire case management system will be built.



Editing the cassandra.yaml file to change the cluster_name from the default to "mydfir" serves as the first step in customizing the database identity for our specific SOC environment.

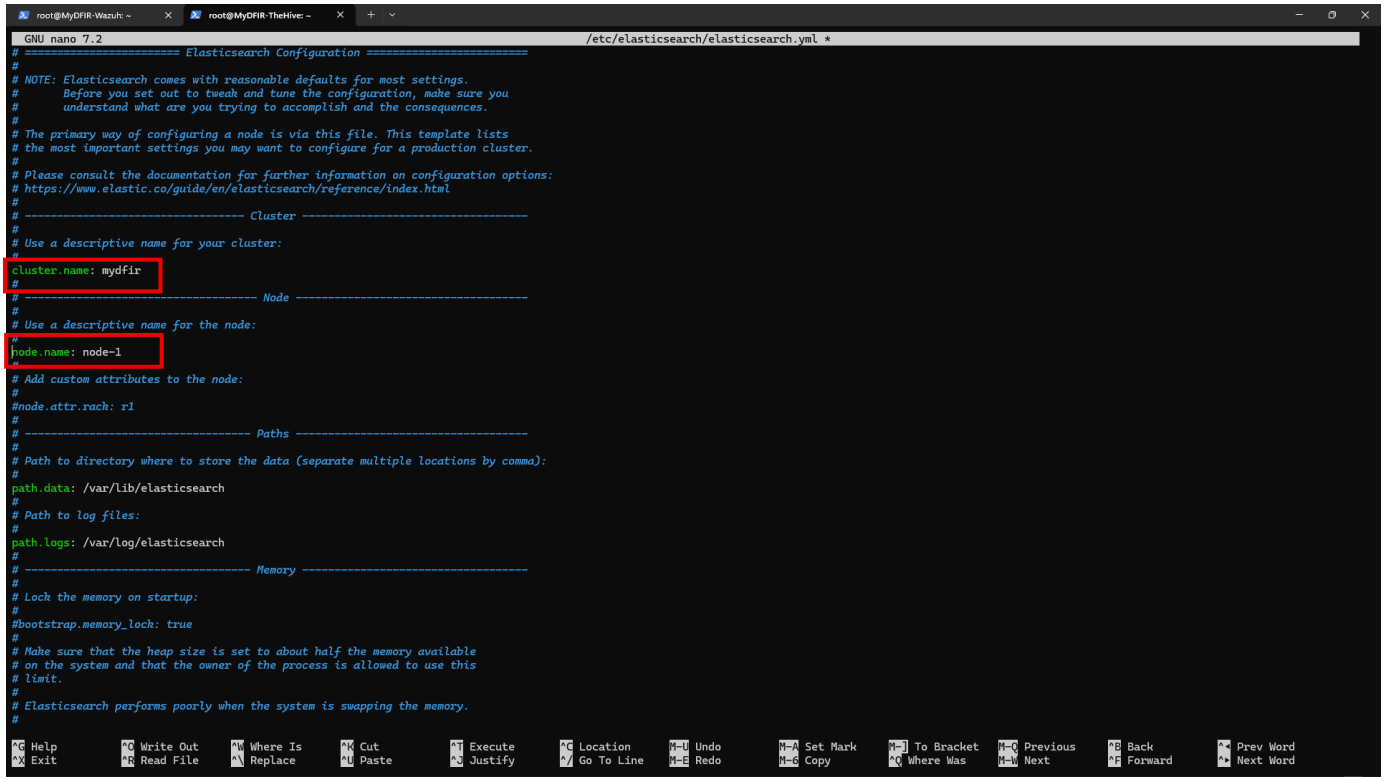
```
# Address or interface to bind to and tell other Cassandra nodes to connect to.
# You _must_ change this if you want multiple nodes to be able to communicate!
#
# Set listen_address OR listen_interface, not both.
#
# Leaving it blank leaves it up to InetAddress.getLocalHost(). This
# will always do the Right Thing _if_ the node is properly configured
# (hostname, name resolution, etc), and the Right Thing is to use the
# address associated with the hostname (it might not be). If unresolvable
# it will fall back to InetAddress.getLoopbackAddress(), which is wrong for production systems.
#
# Setting listen_address to 0.0.0.0 is always wrong.
#
listen_address: [redacted]

# Set listen_address OR listen_interface, not both. Interfaces must correspond
# to a single address, IP aliasing is not supported.
# listen_interface: eth0
```

```
# The address or interface to bind the native transport server to.
#
# Set rpc_address OR rpc_interface, not both.
#
# Leaving rpc_address blank has the same effect as on listen_address
# (i.e. it will be based on the configured hostname of the node).
#
# Note that unlike listen_address, you can specify 0.0.0.0, but you must also
# set broadcast_rpc_address to a value other than 0.0.0.0.
#
# For security reasons, you should not expose this port to the internet.  Firewall it if needed.
rpc_address: [redacted]

# Set rpc_address OR rpc_interface, not both. Interfaces must correspond
# to a single address, IP aliasing is not supported.
# rpc_interface: eth1
```

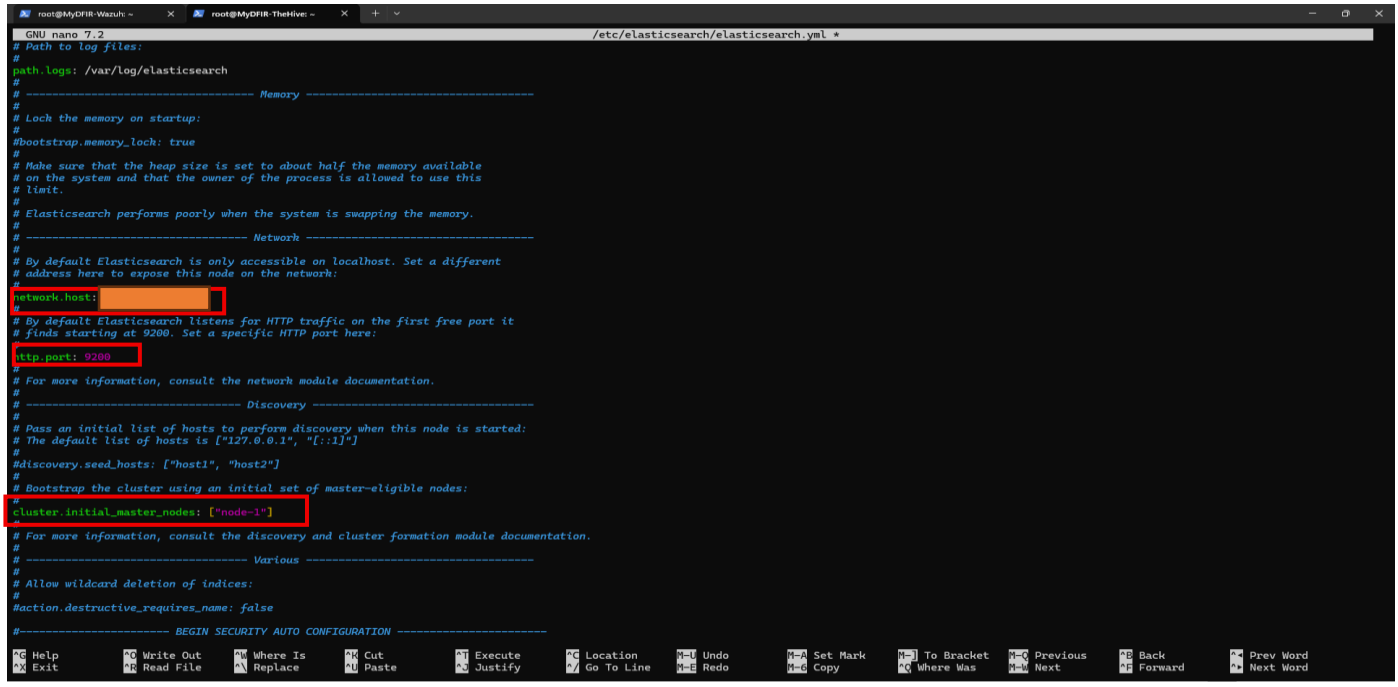
Modifying Cassandra's listen_address and rpc_address to the server's public IP is a critical network configuration that enables external service communication, allowing The Hive application to connect to its database.

```
# any class that implements the SeedProvider interface and has a
# constructor that takes a Map<String, String> of parameters will do.
seed_provider:
  # Addresses of hosts that are deemed contact points.
  # Cassandra nodes use this list of hosts to find each other and learn
  # the topology of the ring.  You must change this if you are running
  # multiple nodes!
  - class_name: org.apache.cassandra.locator.SimpleSeedProvider
    parameters:
      # seeds is actually a comma-delimited list of addresses.
      # Ex: "<ip1>,<ip2>,<ip3>"
      - seeds:
```

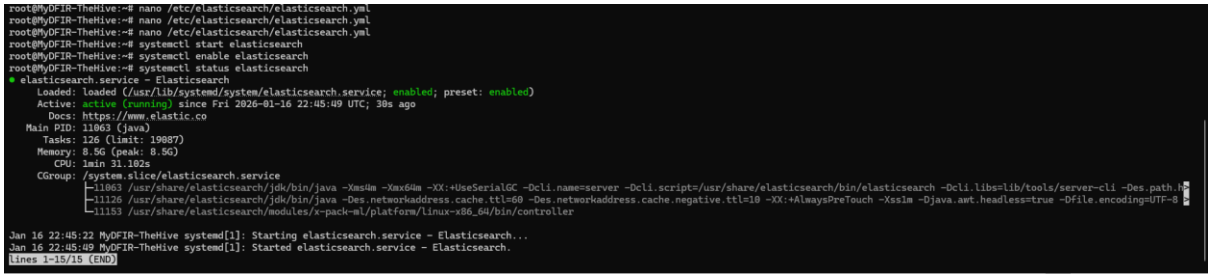
Updating the seed_provider and restarting Cassandra. While the seed provider is most critical for multi-node clusters, modifying it here demonstrates thorough configuration hygiene and prepares the environment for potential scaling. Restarting the service applies all changes, and the status check verifies the database engine is now running with our new identity and network posture.



ElasticSearch Configuration: changed default cluster name to mydfir cluster name and uncommented node name.



Configuring Elasticsearch's cluster.name, network.host, and http.port. Similar to Cassandra, Elasticsearch must be network-accessible. Binding to the server's IP and opening port 9200 allows Wazuh's Filebeat to ship logs into it. Setting the initial master node is essential for cluster stability, preventing "split-brain" scenarios even in a single-node setup.



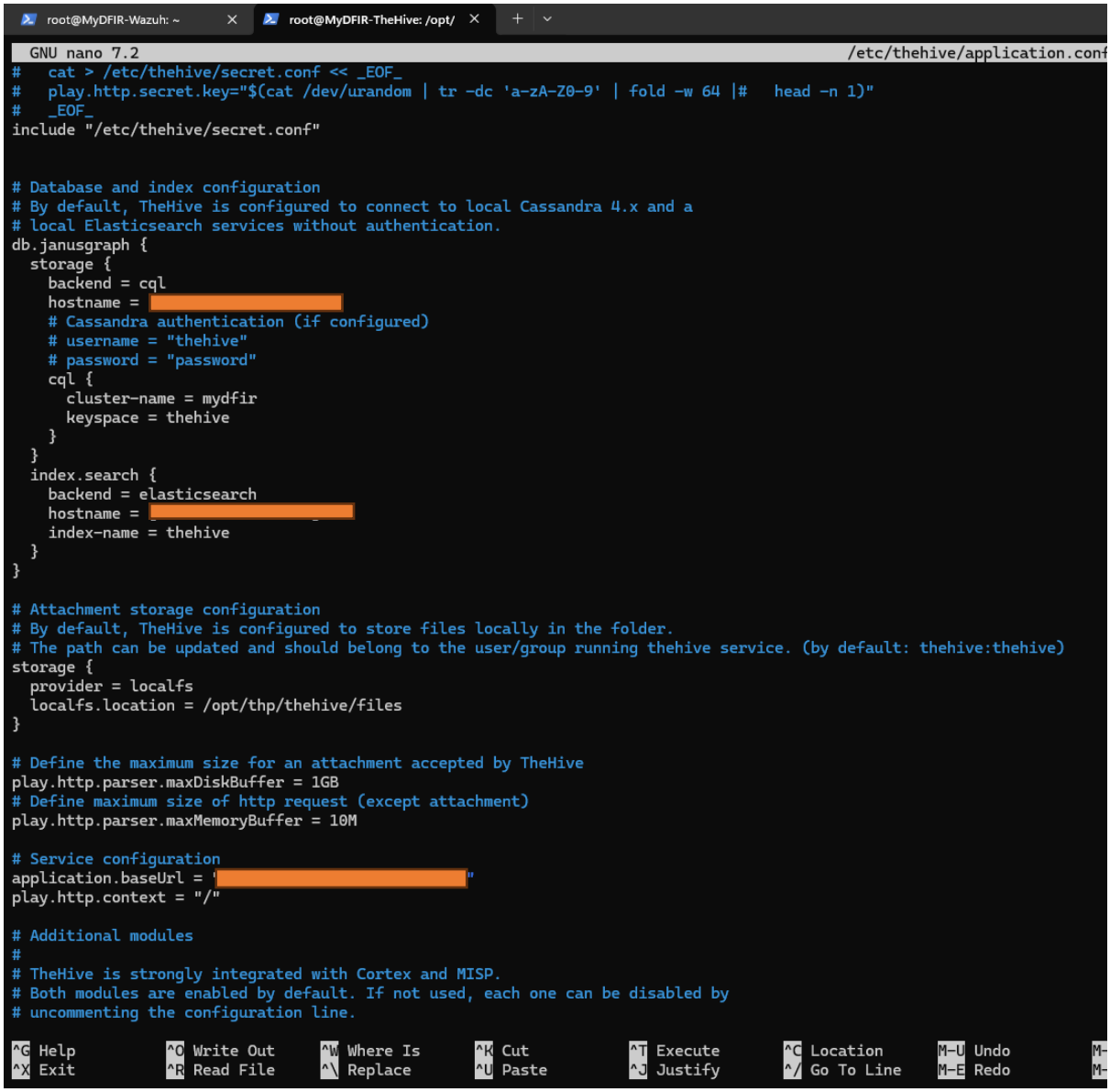
The 'active (running)' status for Elasticsearch confirms all configuration edits were syntactically correct and the service is operational, a prerequisite for data ingestion.

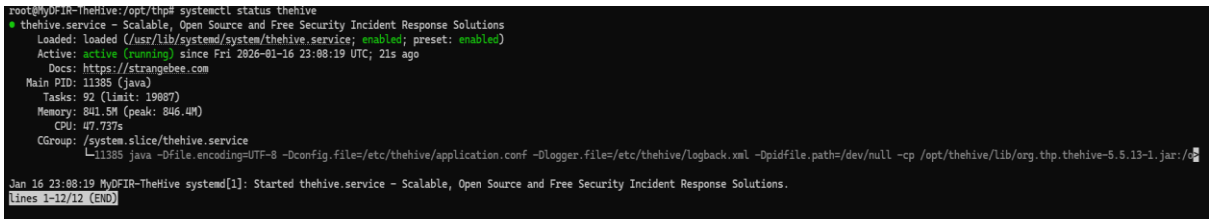
```
root@MyDFIR-TheHive:~# cd /opt/thp
root@MyDFIR-TheHive:/opt/thp# ll
total 12
drwxr-xr-x 3 root root 4096 Jan 15 20:21 ./
drwxr-xr-x 5 root root 4096 Jan 15 20:21 ../
drwxr-xr-x 5 root root 4096 Jan 15 20:21 thehive/
root@MyDFIR-TheHive:/opt/thp# chown -R thehive:thehive /opt/thp
root@MyDFIR-TheHive:/opt/thp# ll
total 12
drwxr-xr-x 3 thehive thehive 4096 Jan 15 20:21 ./
drwxr-xr-x 5 root    root    4096 Jan 15 20:21 ../
drwxr-xr-x 5 thehive thehive 4096 Jan 15 20:21 thehive/
root@MyDFIR-TheHive:/opt/thp#
```

Changing the ownership of the /opt/thp directory to the thehive user addresses a common permission issue, ensuring the application can write to its required working directories.

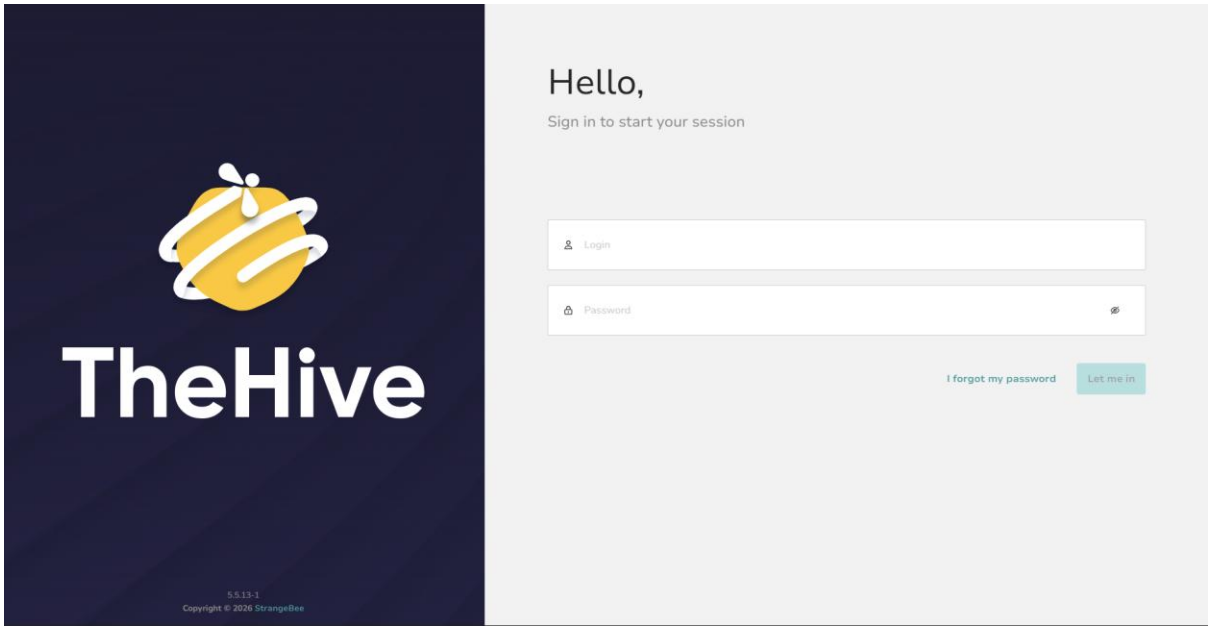
The Hive Configuration:

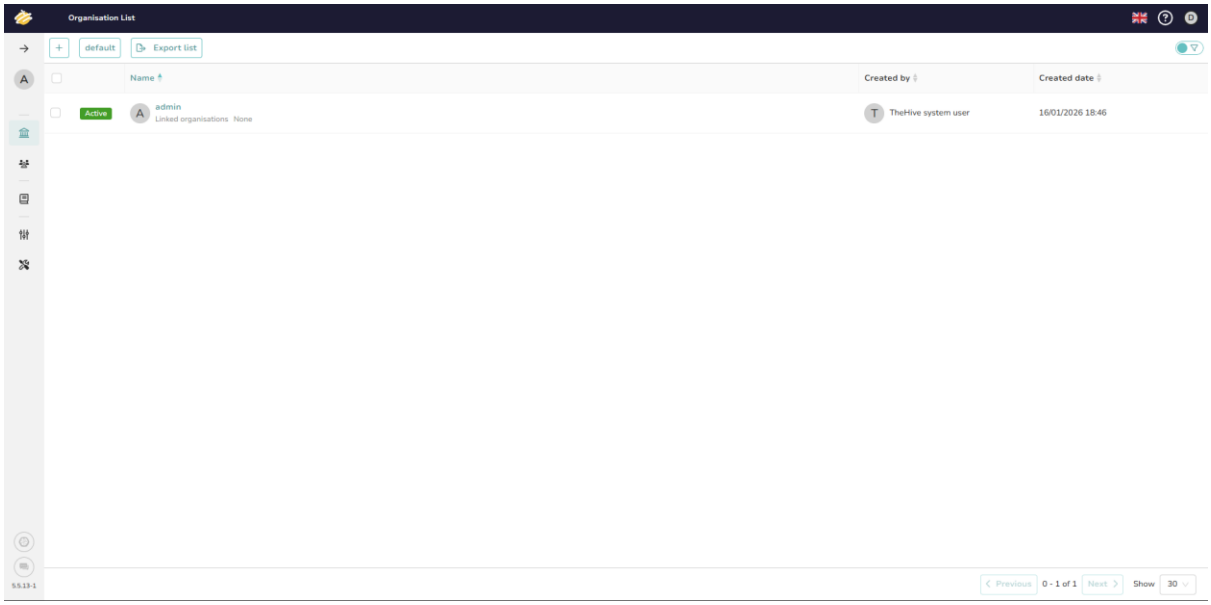
```
root@MyDFIR-TheHive:/opt/thp# nano /etc/thehive/application.conf
root@MyDFIR-TheHive:/opt/thp#
```



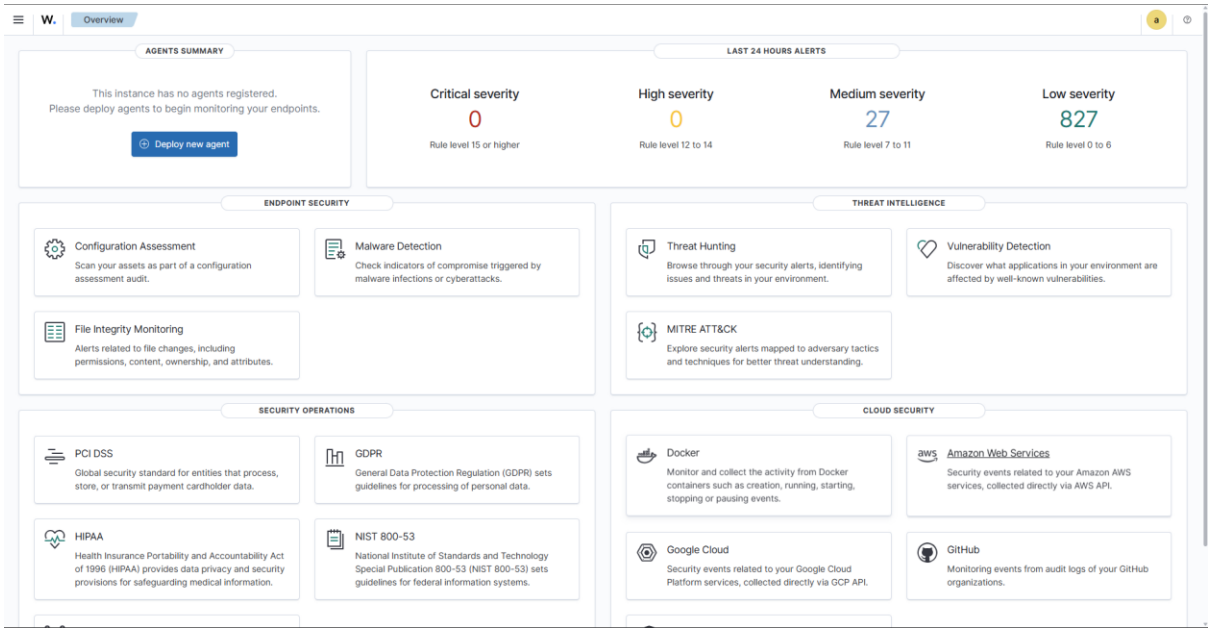


Starting The Hive service launches the application, enabling it ensures persistence after reboot, and checking its status confirms the successful integration of the entire stack—The Hive, Cassandra, and Elasticsearch.

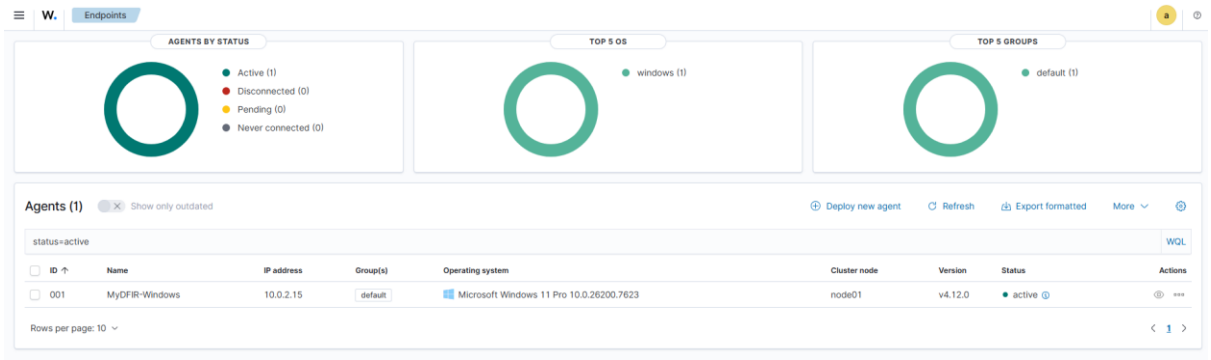




Hive is enabled and we can go to next step.



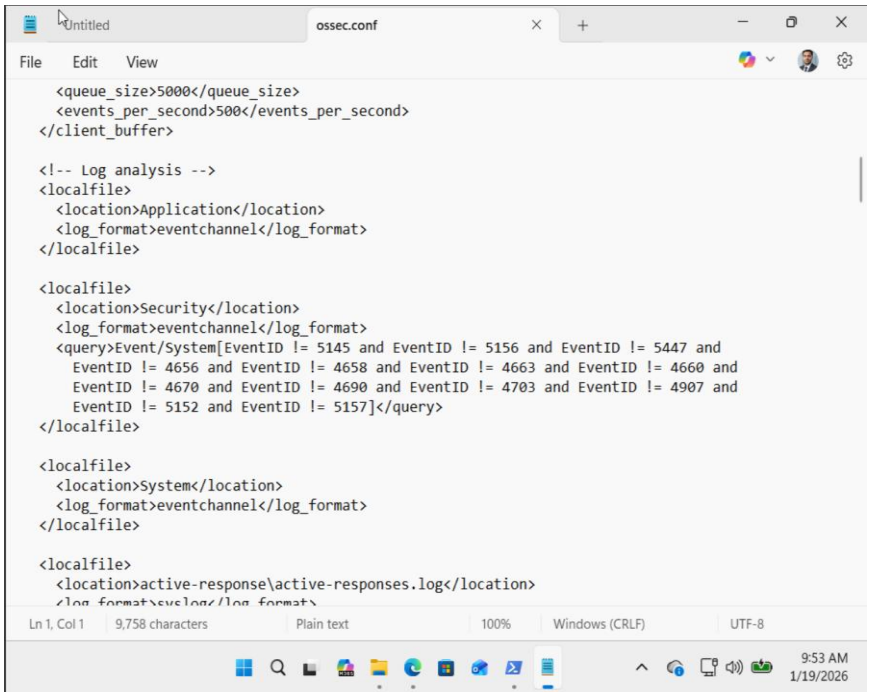
Wazuh manager dashboard overview. This validates the successful installation of the SIEM manager. The dashboard is the central nervous system for monitoring, showing agent health and alert summaries. Seeing this means the core detection and log aggregation platform is ready.



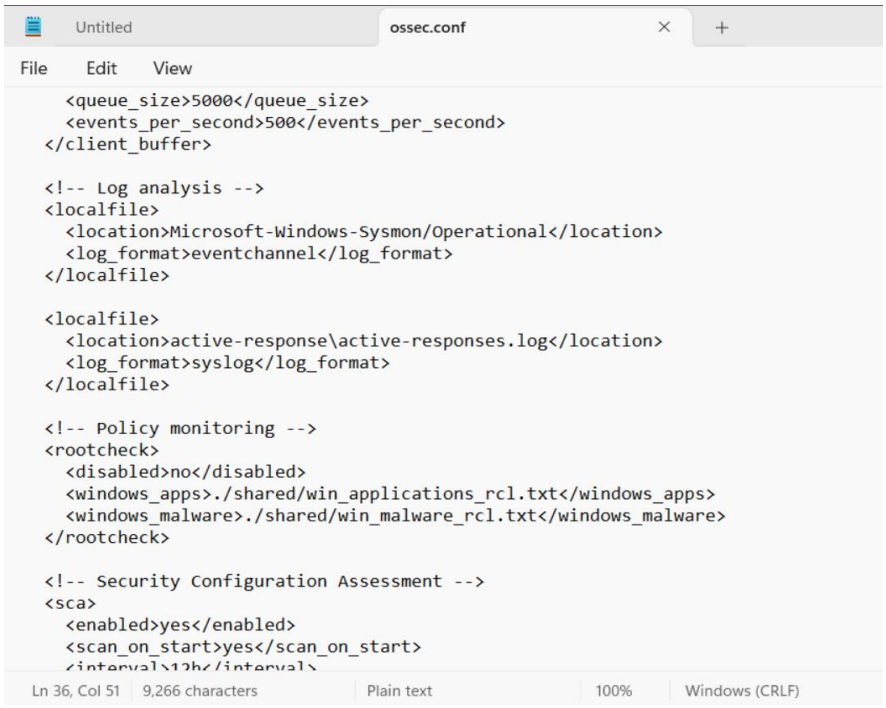
The agent deployment interface generates a pre-configured installation command, simplifying the process of adding the Windows endpoint to the monitoring framework with the correct manager IP and agent name.

The Windows agent appearing as 'Active' in the dashboard is a key connectivity milestone, proving the endpoint can successfully communicate with the SIEM manager.

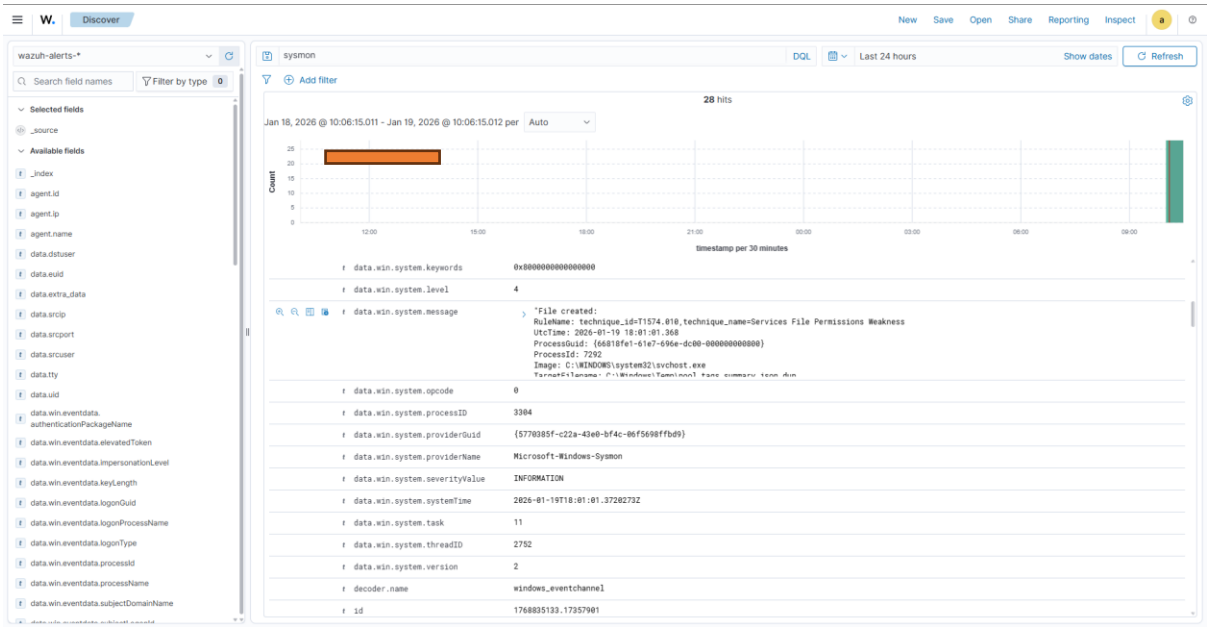
Ingestion of Sysmon Telemetry to Wazuh (SIEM):



Editing the Wazuh agent's configuration file to target only the Sysmon event channel represents a strategic data minimization effort, focusing collection on high-value security telemetry.

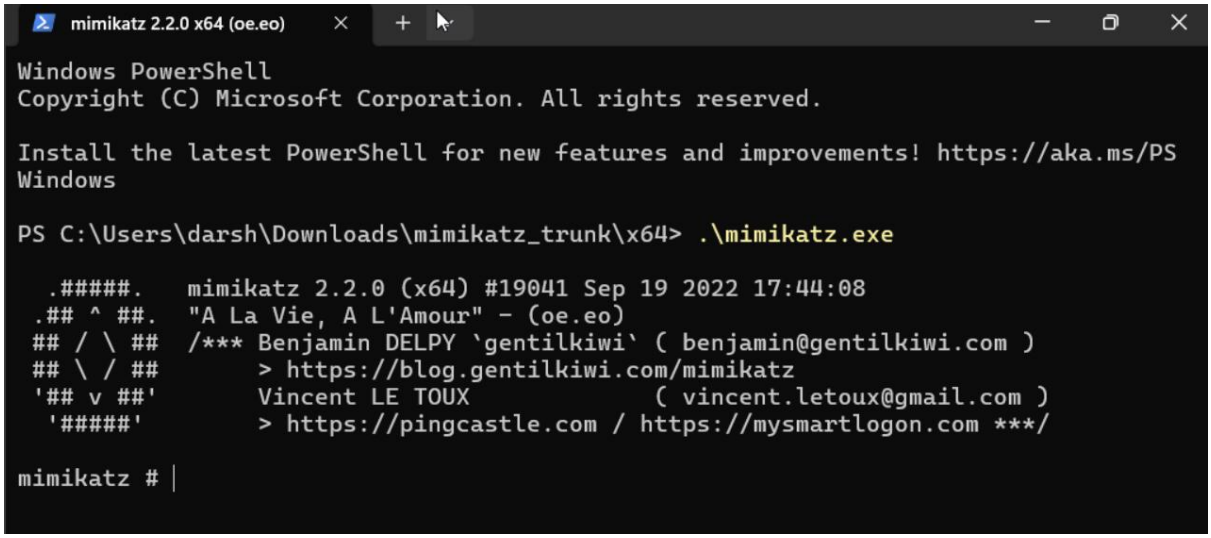


Restarting the Wazuh agent service is a necessary step to load and apply the new focused logging configuration on the windows endpoint.

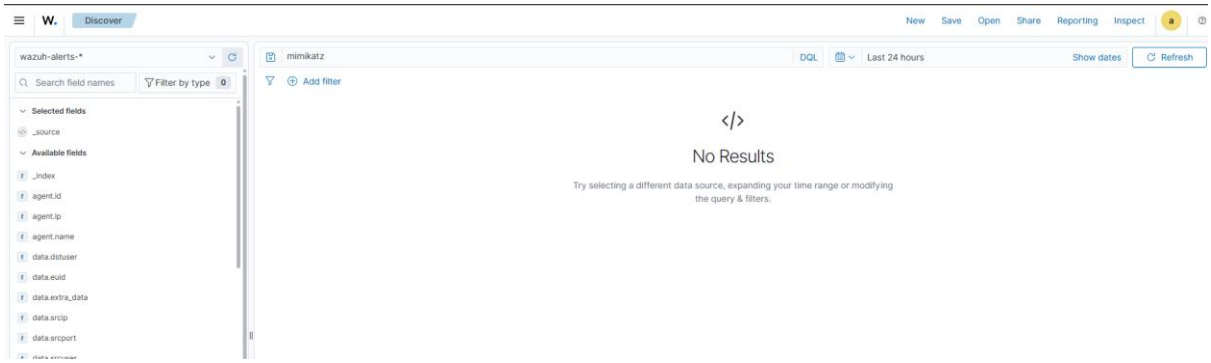


The presence of Sysmon logs within the Wazuh Discover tab validates the agent configuration, confirming that detailed endpoint telemetry is successfully being parsed and indexed.

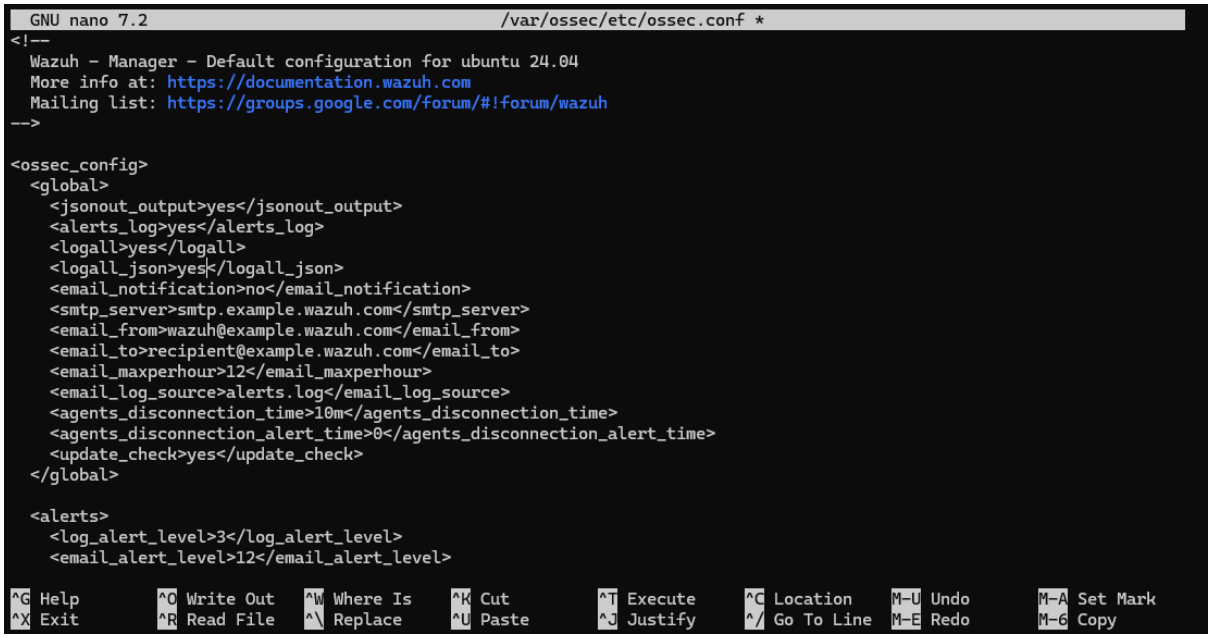
Temporarily disabling Windows Defender's real-time protection is a controlled action to allow the Mimikatz test tool to execute without interference for detection testing.



The initial execution of Mimikatz without generating an alert establishes a baseline, demonstrating the need for custom detection logic within the SIEM.



To generate an alert on Wazuh manager, we have to create a custom rule for mimikatz. And before that we have make some configuration changes to wazuh itself



Enabling the logall option in the Wazuh manager's configuration is essential for rule development, as it archives all normalized events for inspection and analysis.

```
root@MyDFIR-Wazuh:~# cp /var/ossec/etc/ossec.conf ~/ossec_backup.conf
root@MyDFIR-Wazuh:~# nano /var/ossec/etc/ossec.conf
root@MyDFIR-Wazuh:~# systemctl restart wazuh-manager.service
root@MyDFIR-Wazuh:~# cd /var/ossec/logs/archives
root@MyDFIR-Wazuh:/var/ossec/logs/archives# ls -ls
total 100
 4 drwxr-x--- 3 wazuh wazuh  4096 Jan 15 19:53 2026
72 -rw-r----- 2 wazuh wazuh 67562 Jan 19 15:46 archives.json
24 -rw-r----- 2 wazuh wazuh 24457 Jan 19 15:46 archives.log
root@MyDFIR-Wazuh:/var/ossec/logs/archives# cat archives.log
2026 Jan 19 15:43:02 MyDFIR-Wazuh->rootcheck Starting rootcheck scan.
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': tmpfs            812580     1292    811288       1% /run
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': efivarfs            256       28       224      12% /sys/firmware/efi/efivars
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': tmpfs           4062892       80   4062812       1% /dev/shm
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': Filesystem     1024-blocks      Used Available Capacity Mounted on
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': tmpfs              5120        0      5120       0% /run/lock
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': /dev/vda1        523244     6288    516956       2% /boot/efi
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': tmpfs            812576       12    812564       1% /run/user/0
2026 Jan 19 15:43:06 MyDFIR-Wazuh->df -P ossec: output: 'df -P': /dev/vda2     156735724 26561520 123403412      18% /
2026 Jan 19 15:43:06 MyDFIR-Wazuh->last -n 20 ossec: output: 'last -n 20':
root     pts/0        [redacted] Mon Jan 19 14:28   still logged in
root     pts/0        [redacted] Fri Jan 16 21:59 - 00:14  (02:14)
reboot   system boot  [redacted] Fri Jan 16 21:59   still running
root     pts/0        [redacted] Thu Jan 15 22:14 - down   (03:40)
reboot   system boot  [redacted] Thu Jan 15 22:10 - 01:55  (03:44)
root     pts/0        [redacted] Thu Jan 15 19:46 - down   (00:38)
reboot   system boot  [redacted] Thu Jan 15 19:42 - 20:24  (00:41)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
linuxuse pts/0        [redacted] Tue Jan  6 15:59 - 15:59  (00:00)
wtmp begins Tue Jan [redacted]
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:06 MyDFIR-Wazuh env[112613]: Started wazuh-monitord...
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:06 MyDFIR-Wazuh env[112979]: 2026/01/19 15:43:06 wazuh-modulesd:router: INFO: Loaded router module.
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:06 MyDFIR-Wazuh env[112979]: 2026/01/19 15:43:06 wazuh-modulesd:content_manager: INFO: Loaded content_manager module.
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh systemd[1]: Starting snapd.service - Snap Daemon...
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh snapd[113148]: overlord.go:294: Acquiring state lock file
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh snapd[113148]: overlord.go:299: Acquired state lock file
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:06 MyDFIR-Wazuh sshd[112948]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=64.225.70.76  user=root
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh snapd[113148]: daemon.go:276: started snapd/2.73+ubuntu24.04 (series 16; classic) ubuntu/24.04 (amd64) linux/6.8.0-90-generic.
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh kernel: loop0: detected capacity change from 0 to 8
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh snapd[113148]: daemon.go:370: adjusting startup timeout by 30s (pessimistic estimate of 30s plus 5s per snap)
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh snapd[113148]: backends.go:70: AppArmor status: apparmor is enabled and all features are available
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh systemd[1]: tmp-syscheck\x2dmountpoint\x2d1061707713.mount: Deactivated successfully.
2026 Jan 19 15:43:08 MyDFIR-Wazuh->journald Jan 19 15:43:07 MyDFIR-Wazuh snapd[113148]: backend.go:145: reloading profiles for snap-confine
```

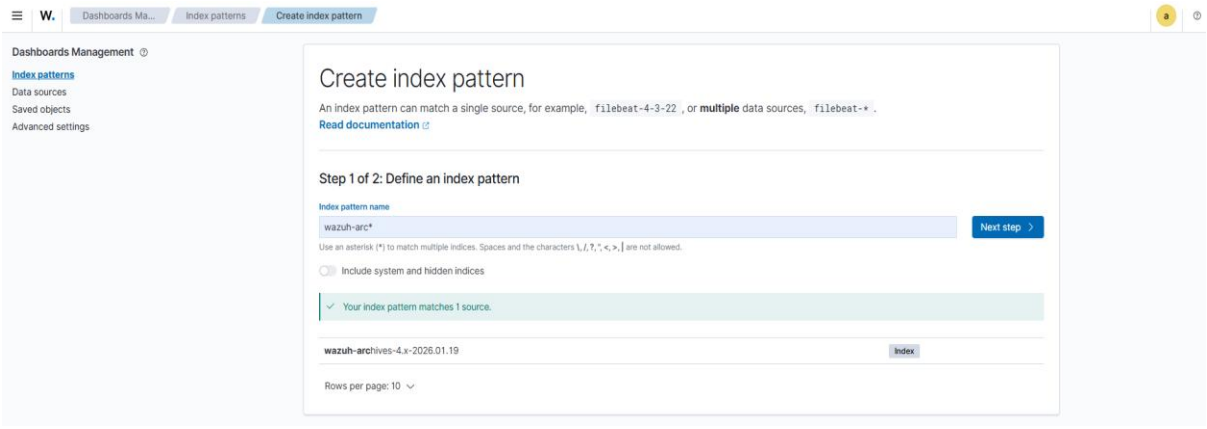
Viewing the live archives.log output provides the raw JSON data of the Mimikatz event, which serves as the blueprint for crafting a precise detection rule.

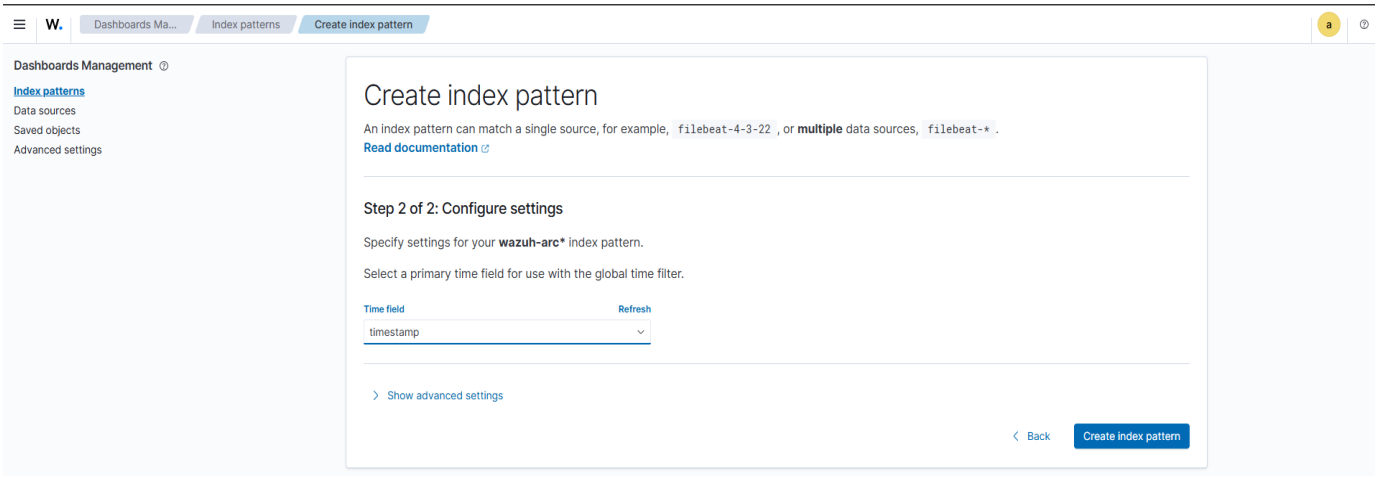
```
root@MyDFIR-Wazuh:/var/ossec/logs/archives# nano /etc/filebeat/filebeat.yml
root@MyDFIR-Wazuh:/var/ossec/logs/archives# systemctl restart filebeat.service
root@MyDFIR-Wazuh:/var/ossec/logs/archives#
```

Configuring Filebeat to enable the Wazuh archives module completes the data pipeline, shipping the verbose logs from the manager to Elasticsearch for advanced querying.

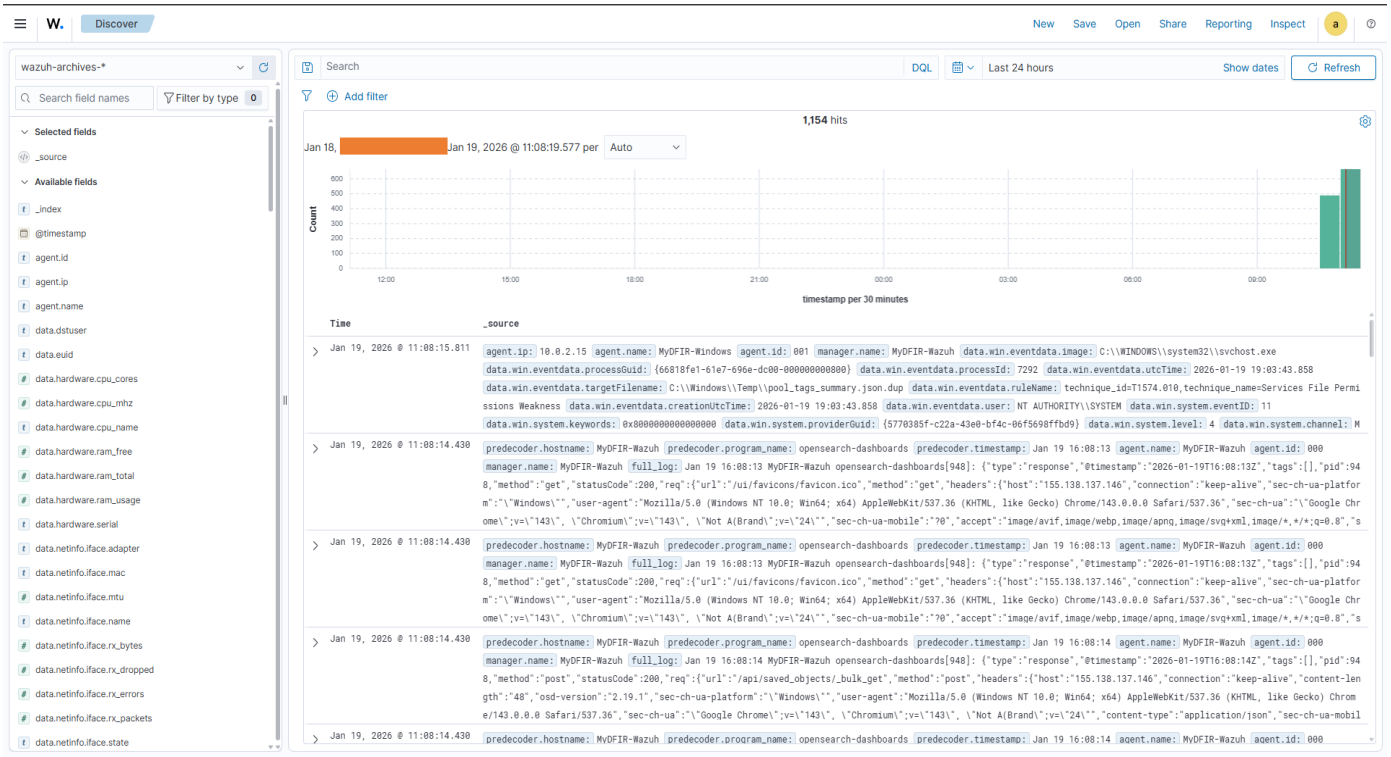
```
  GNU nano 7.2
# Wazuh - Filebeat configuration file
output.elasticsearch.hosts:
        - 127.0.0.1:9200
#         - <elasticsearch_ip_node_2>:9200
#         - <elasticsearch_ip_node_3>:9200

output.elasticsearch:
  protocol: https
  username: ${username}
  password: ${password}
  ssl.certificate_authorities:
    - /etc/filebeat/certs/root-ca.pem
  ssl.certificate: "/etc/filebeat/certs/wazuh-server.pem"
  ssl.key: "/etc/filebeat/certs/wazuh-server-key.pem"
setup.template.json.enabled: true
setup.template.json.path: '/etc/filebeat/wazuh-template.json'
setup.template.json.name: 'wazuh'
setup.ilm.overwrite: true
setup.ilm.enabled: false

filebeat.modules:
  - module: wazuh
    alerts:
      enabled: true
    archives:
      enabled: true
```

Archives is enabled to true, which allows wazuh to log everything under an archive.

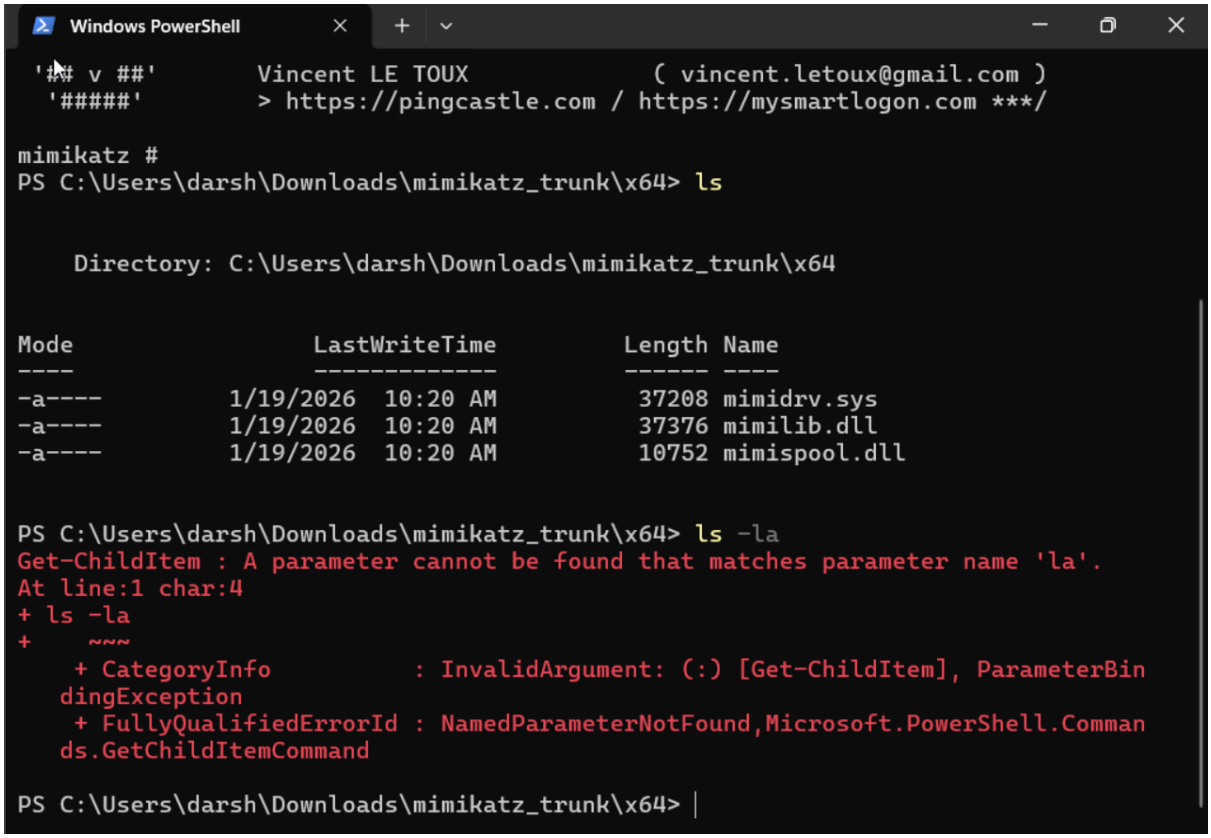




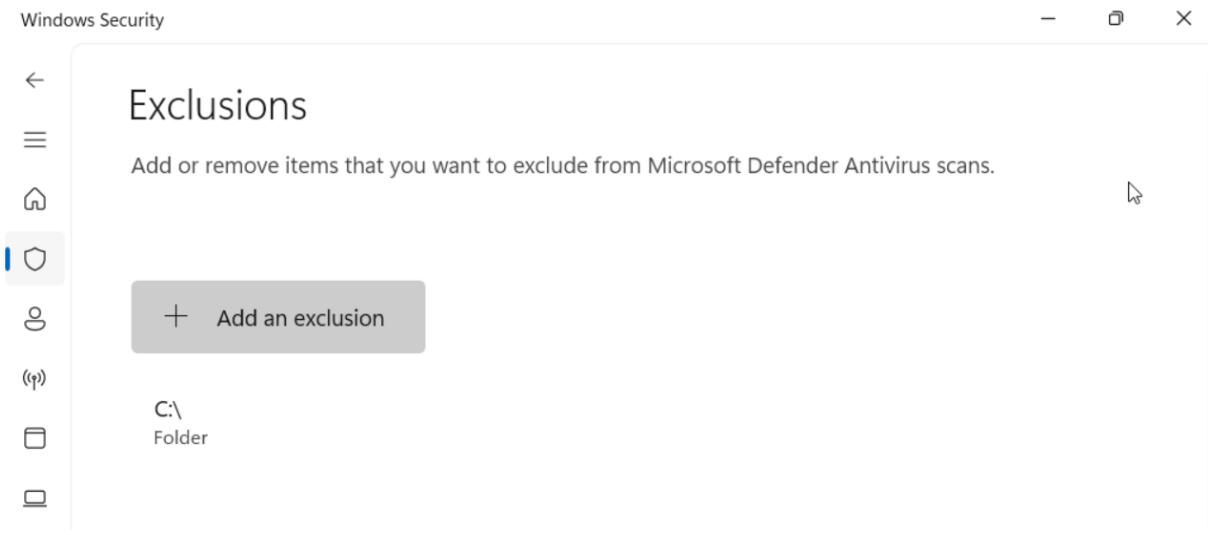
Creating a new index to find wazuh-archives-* pattern. An index pattern acts as a query filter to define which subset of your logged data you want to interact with in the dashboard, enabling efficient searching and analysis of large datasets spread across many individual indices.



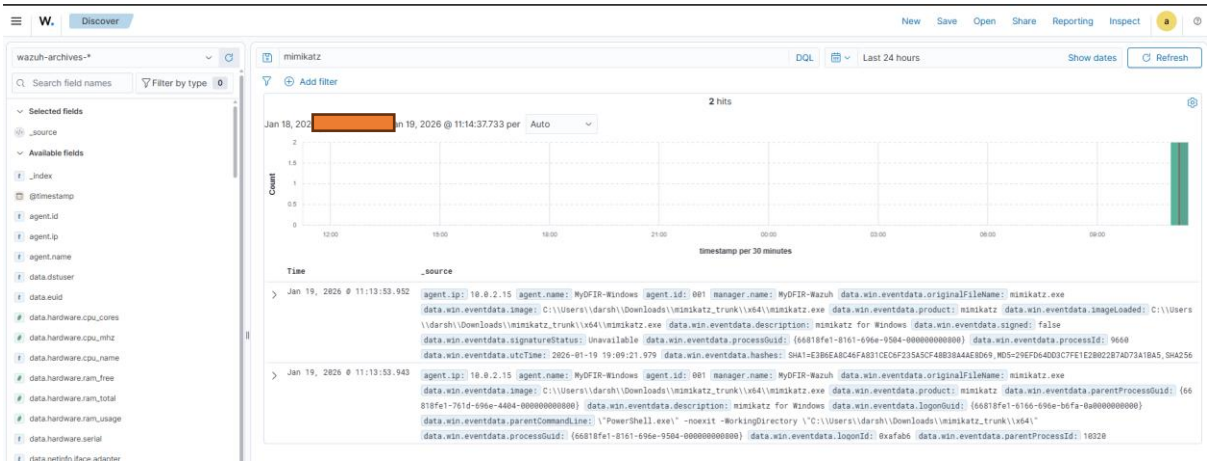
Successfully finding the Mimikatz event within the new archive index confirms the data flow and allows for identification of the key field (OriginalFileName) to use in a detection rule.



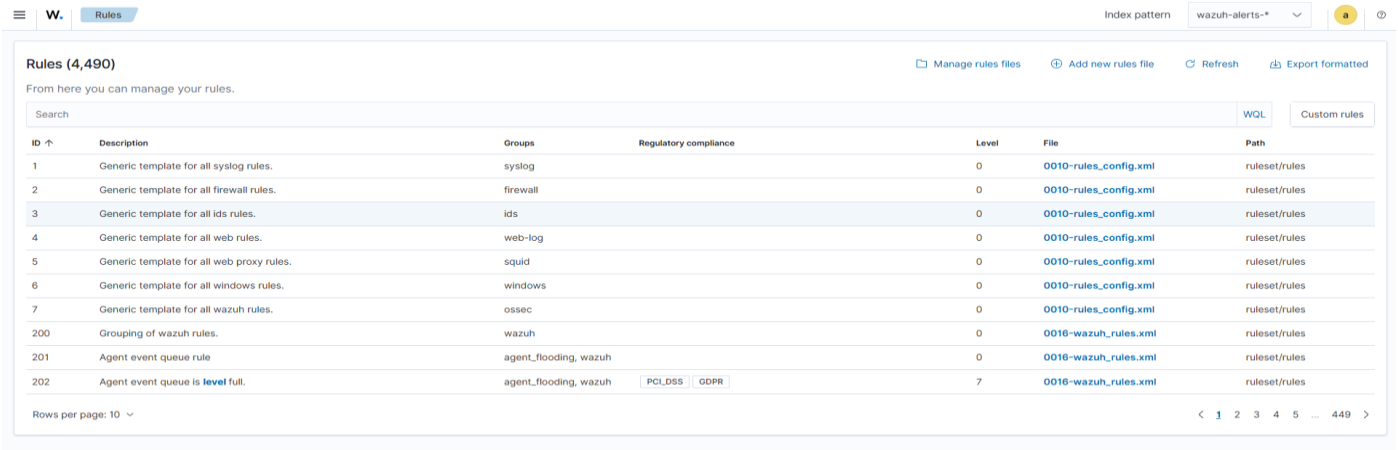
Defender deleted my mimikatz.exe so we have disable firewall for C drive.



Adding a Windows Defender exclusion for the C: drive is a troubleshooting step to prevent the continuous deletion of the test binary, ensuring consistent tool availability.



We can see the mimikatz logs into wazuh after disabling firewall and applying filter.



Creating a custom rule that create an alert when mimikatz is executed.

Rules

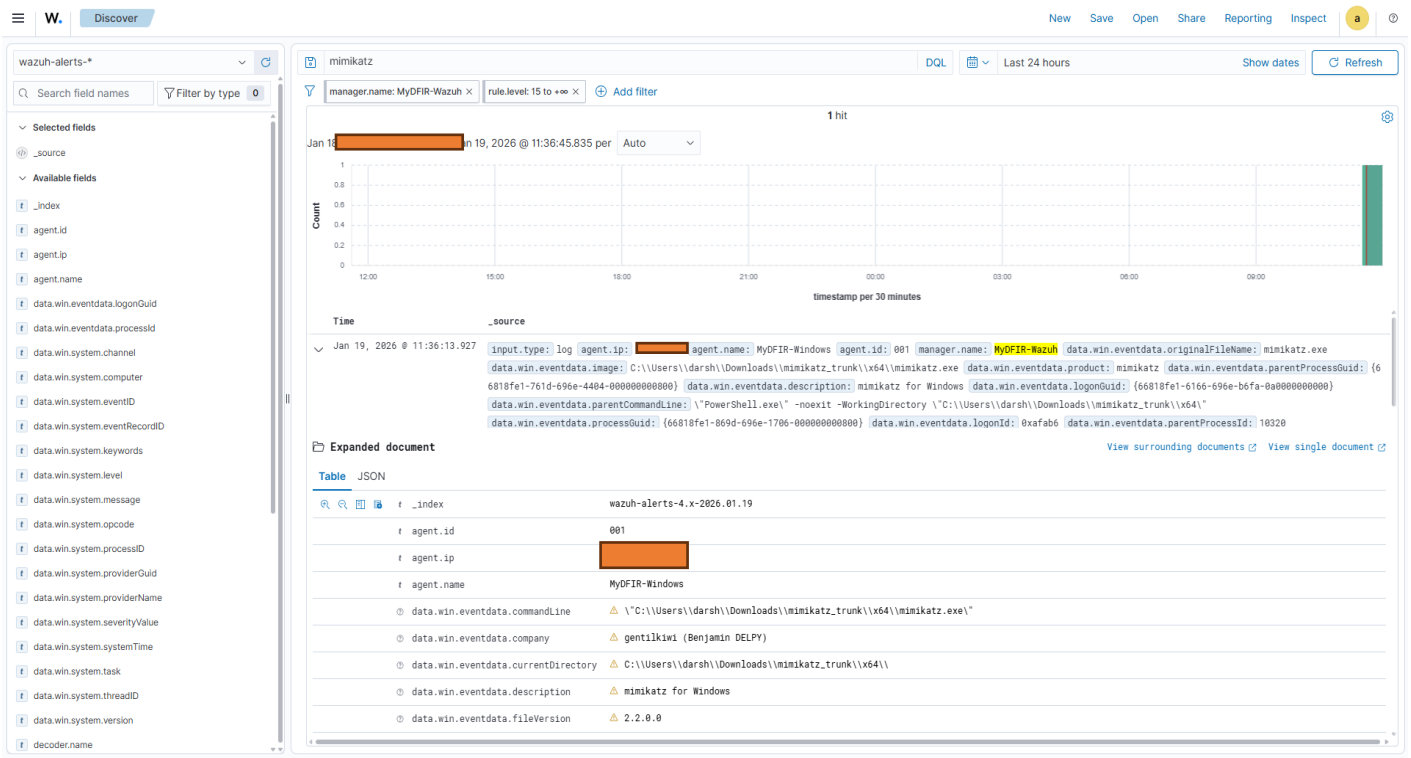
local_rules.xml

```
<!-- Local rules -->

<!-- Modify it at your will. -->
<!-- Copyright (C) 2015, Wazuh Inc. -->

<!-- Example -->
<group name="local,syslog,sshd,">

  <!--
  Dec 10 01:02:02 host sshd[1234]: Failed none for root from 1.1.1.1 port 1066 ssh2
  -->
  <rule id="100001" level="5">
    <if_sid>5716</if_sid>
    <srcip>1.1.1.1</srcip>
    <description>sshd: authentication failed from IP 1.1.1.1.</description>
    <group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
  </rule>

  <rule id="100002" level="15">
    <if_group>sysmon_event1</if_group>
    <field name="win.eventdata.originalFileName" type="pcre2">(?i)mimikatz\.exe</field>
    <description>Mimikatz Usage Detected</description>
    <mitre>
      <id>T1003</id>
    </mitre>
  </rule>

</group>
```

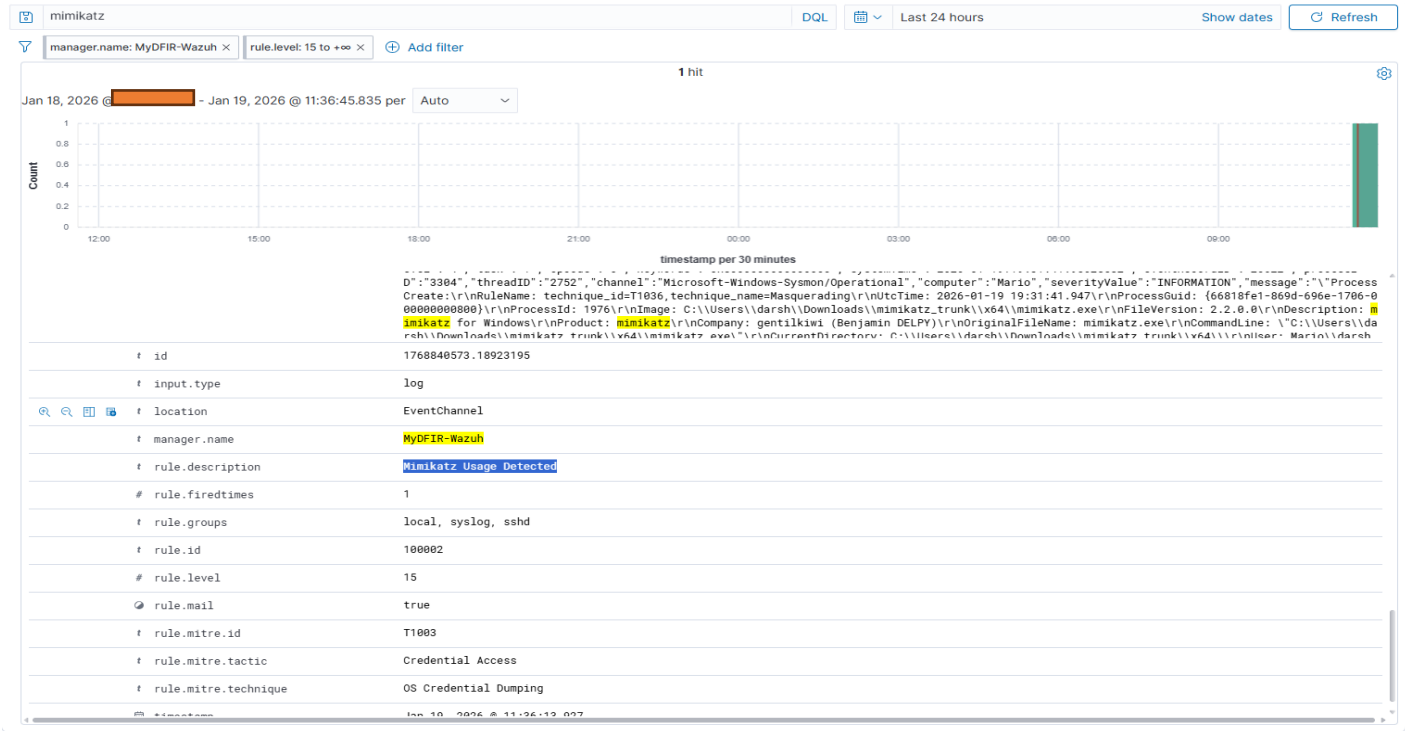
Accessing and editing the local_rules.xml file is where detection logic is codified, translating observed adversary behavior into an XML rule that triggers an alert. Saving the new rule and restarting the Wazuh manager service activates the detection logic, loading it into the running SIEM processes.

```
-a----         1/19/2026  11:08 AM          37376 mimilib.dll
-a----         1/19/2026  11:08 AM          10752 mimispool.dll


PS C:\Users\darsh\Downloads\mimikatz_trunk\x64> ls


    Directory: C:\Users\darsh\Downloads\mimikatz_trunk\x64


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         1/19/2026  11:31 AM          37208 mimidrv.sys
-a----         1/19/2026  11:31 AM        1355264 mimikatz.exe
-a----         1/19/2026  11:31 AM          37376 mimilib.dll
-a----         1/19/2026  11:31 AM          10752 mimispool.dll


PS C:\Users\darsh\Downloads\mimikatz_trunk\x64> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

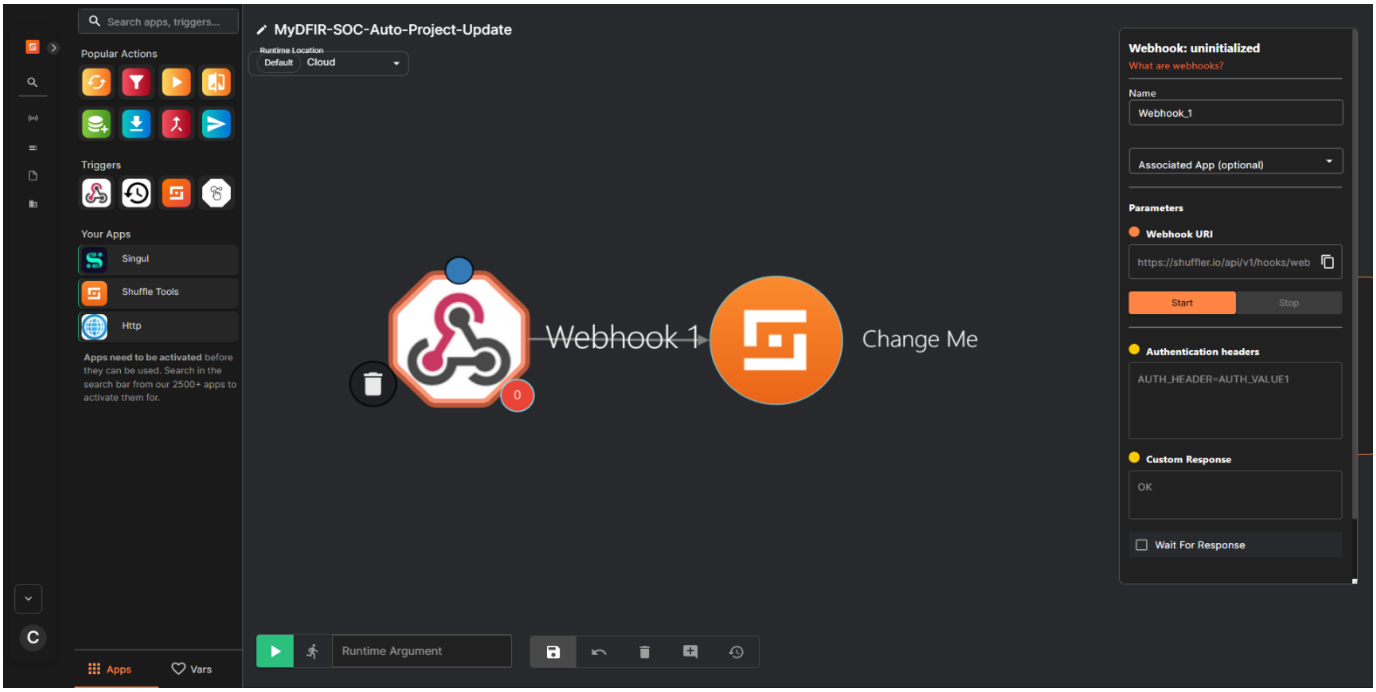
Executing Mimikatz again after the rule deployment serves as a controlled test to validate the functionality of the new detection.



Alert is generated using custom rule under the **wazuh-alerts-*** that I applied to detect mimikatz.



The appearance of a "Mimikatz Usage Detected" alert in **the wazuh-alerts-*** index confirms the custom rule is syntactically correct and operating as designed.



Generating and copying the unique webhook URL in Shuffle creates the endpoint that will connect the SIEM's alerting to the automation orchestration platform.

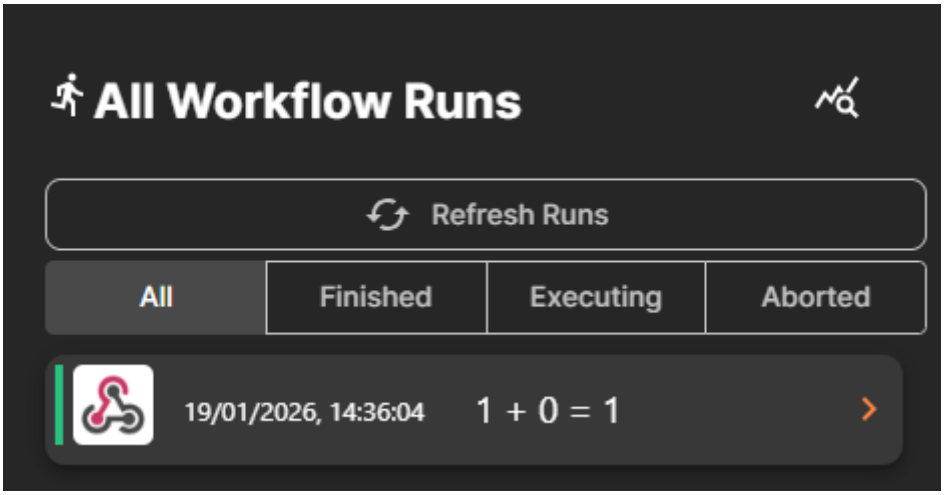
```
root@MyDFIR-Wazuh:/var/ossec/logs/archives# nano /var/ossec/etc/ossec.conf
root@MyDFIR-Wazuh:/var/ossec/logs/archives# systemctl restart wazuh-manager.service
root@MyDFIR-Wazuh:/var/ossec/logs/archives#
```

```
 GNU nano 7.2                                  /var/ossec/etc/ossec.conf *
<!--
  Wazuh - Manager - Default configuration for ubuntu 24.04
  More info at: https://documentation.wazuh.com
  Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    <alerts_log>yes</alerts_log>
    <logall>yes</logall>
    <logall_json>yes</logall_json>
    <email_notification>no</email_notification>
    <smtp_server>smtp.example.wazuh.com</smtp_server>
    <email_from>wazuh@example.wazuh.com</email_from>
    <email_to>recipient@example.wazuh.com</email_to>
    <email_maxperhour>12</email_maxperhour>
    <email_log_source>alerts.log</email_log_source>
    <agents_disconnection_time>10m</agents_disconnection_time>
    <agents_disconnection_alert_time>0</agents_disconnection_alert_time>
    <update_check>yes</update_check>
  </global>

  <integration>
    <name>shuffle</name>
    <hook_url>https://shuffler.io/api/v1/hooks/webhook_3b958c98-f707-43d6-84d0-07e95330fc87</hook_url>
    <rule_id>100002</rule_id>
    <alert_format>json</alert_format>
  </integration>

  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>12</email_alert_level>
  </alerts>

  <!-- Choose between "plain", "json", or "plain,json" for the format of internal logs -->
  <logging>
    <log_format>plain</log_format>
  </logging>

  <remote>
    <connection>secure</connection>
    <port>1514</port>
    <protocol>tcp</protocol>
    <queue_size>131072</queue_size>
  </remote>

  <!-- Policy monitoring -->
  <rootcheck>
    <disabled>no</disabled>

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark     M-] To Bracket   M-Q Previous     ^B Back          ^← Prev Word
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy         ^Q Where Was     M-W Next         ^F Forward       ^→ Next Word
```

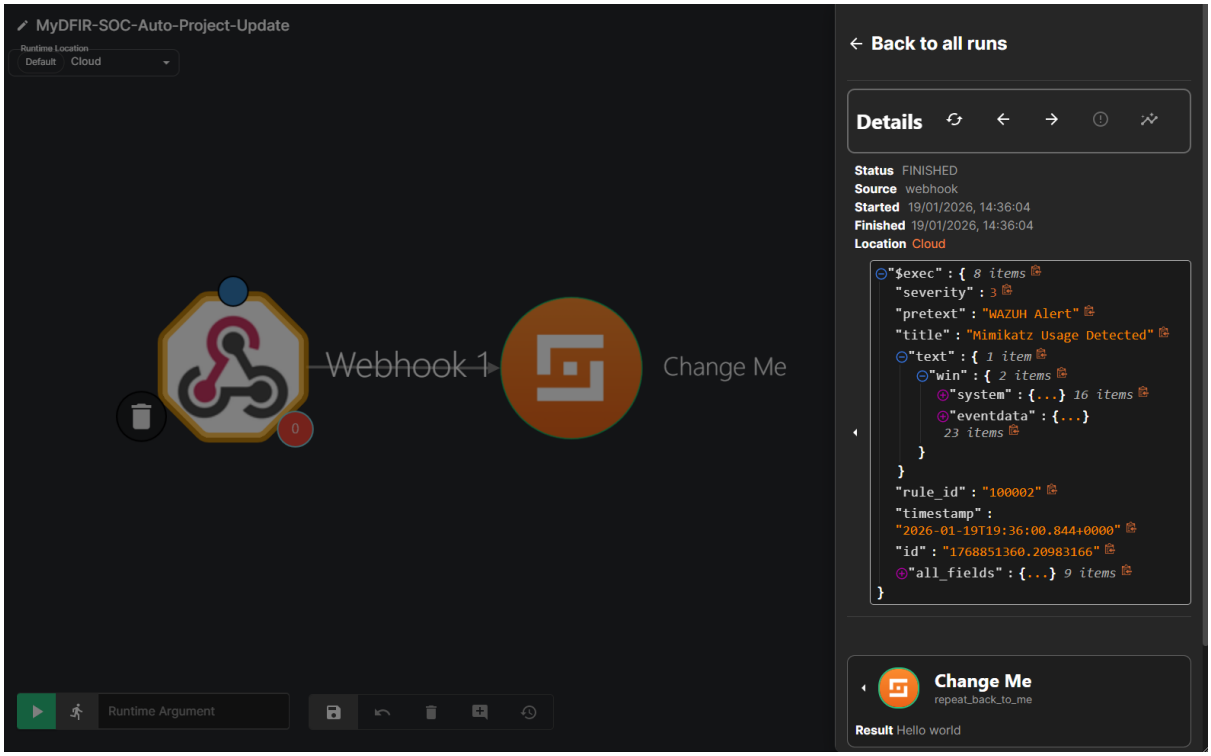
Adding an integration block to Wazuh's configuration file instructs the SIEM to forward alerts for rule 100002 to the Shuffle webhook, initiating the automated response workflow.

```
PS C:\Users\darsh\Downloads\mimikatz_trunk\x64> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

Executed mimikatz.exe again to see alert is triggered into shuffle.

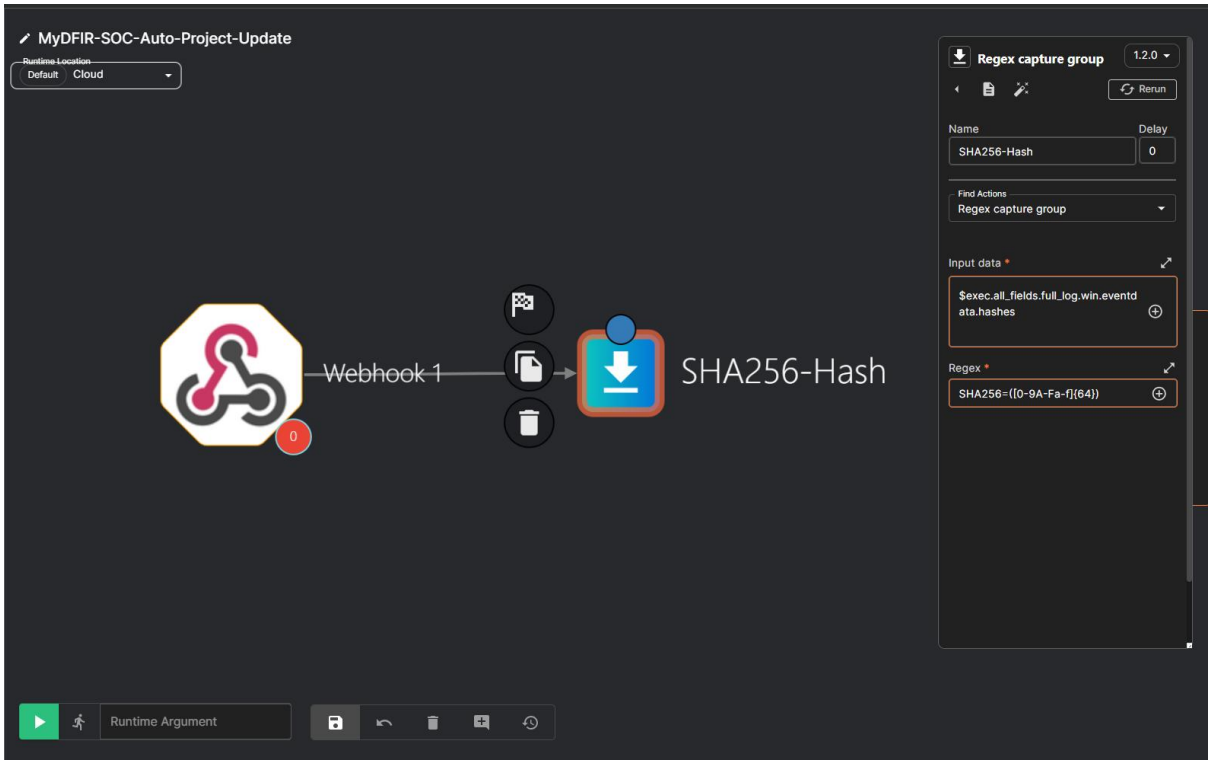


The webhook trigger appearing in Shuffle's execution history provides definitive proof that the integration is correctly configured and data is flowing between systems.

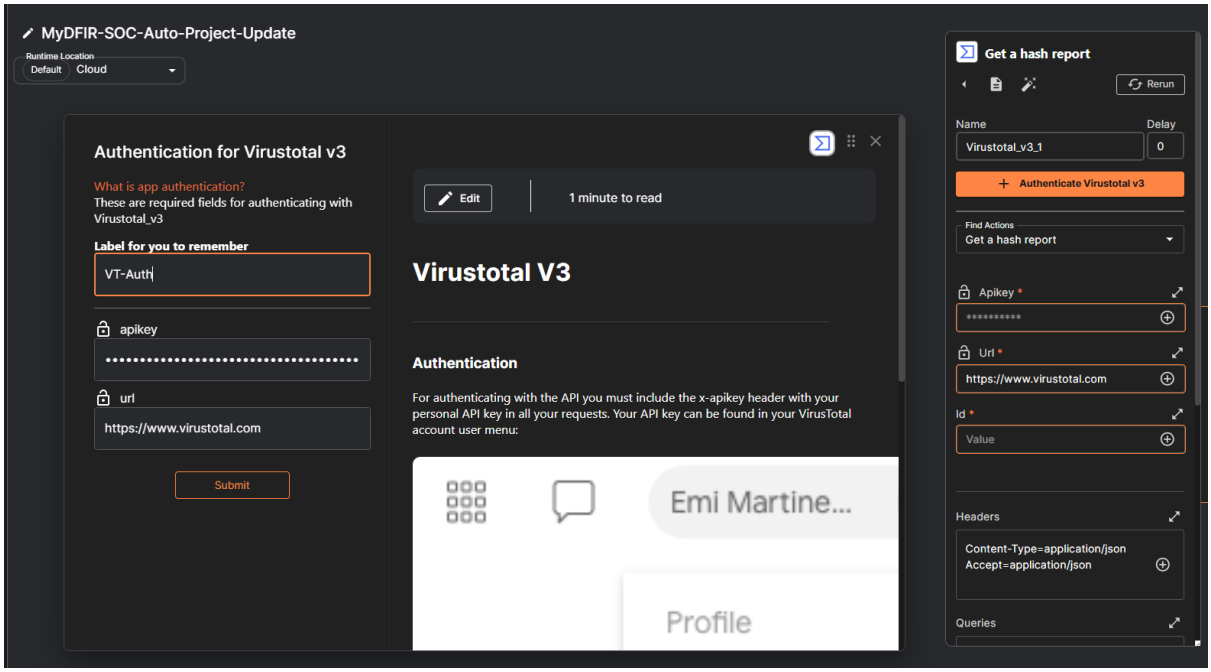


Examining the webhook payload reveals the full context of the alert sent by Wazuh, showcasing the rich dataset available for building automated actions.

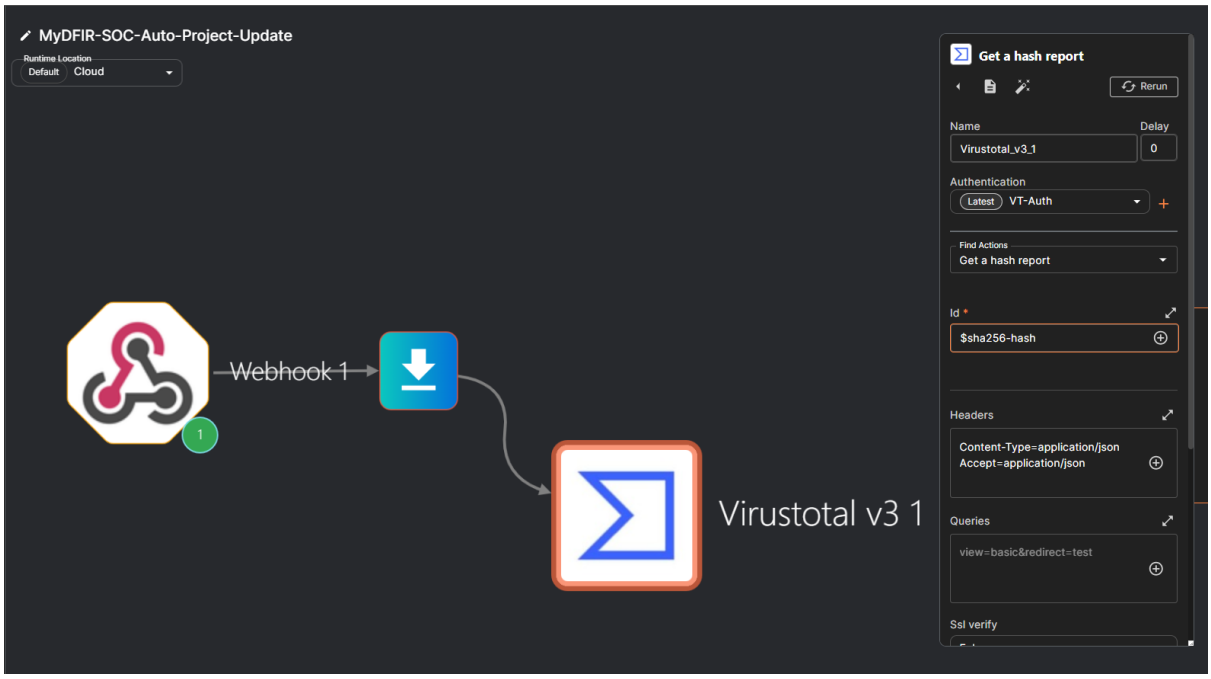
Automation



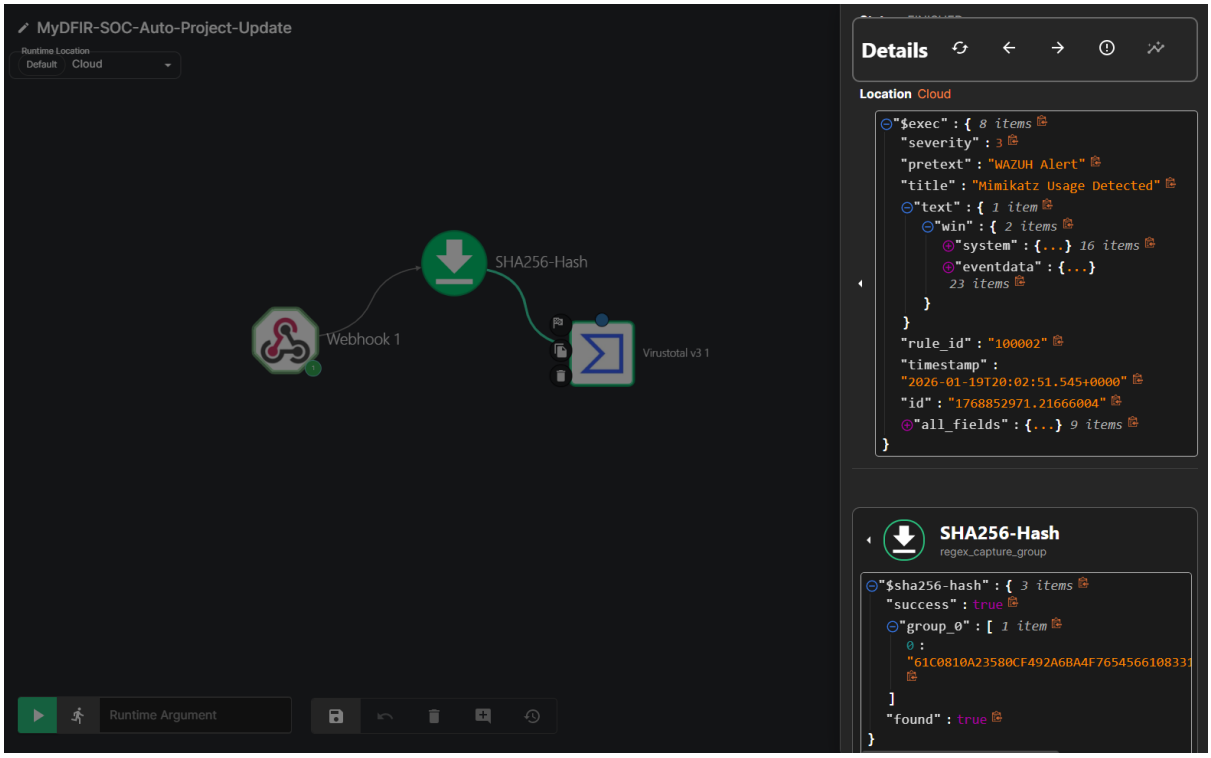
created regex capture group that extract hashes from data.



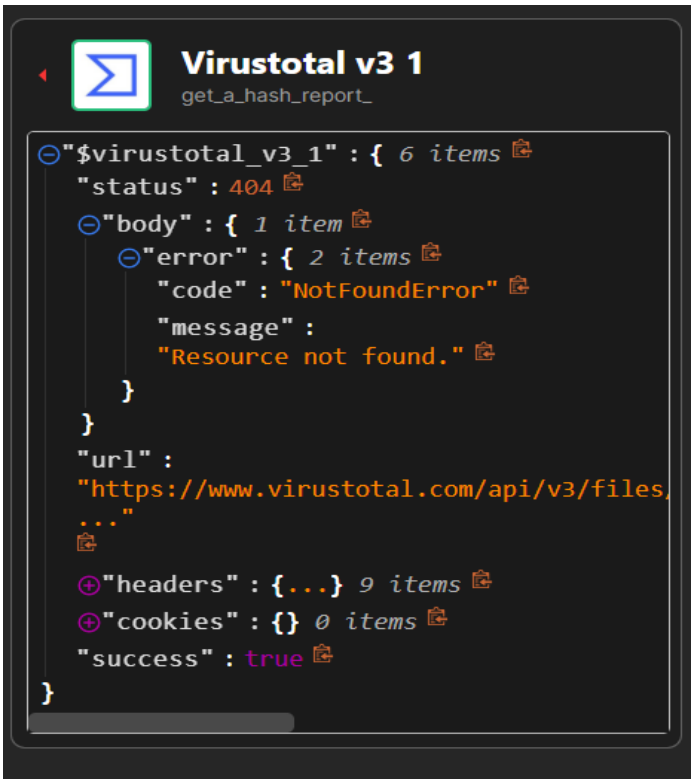
Configuring a Regex node to extract the SHA-256 hash from the alert data is a data preparation step, isolating a key indicator for threat intelligence enrichment.



Verfied virustotal API key and it will take action to get a hash report.



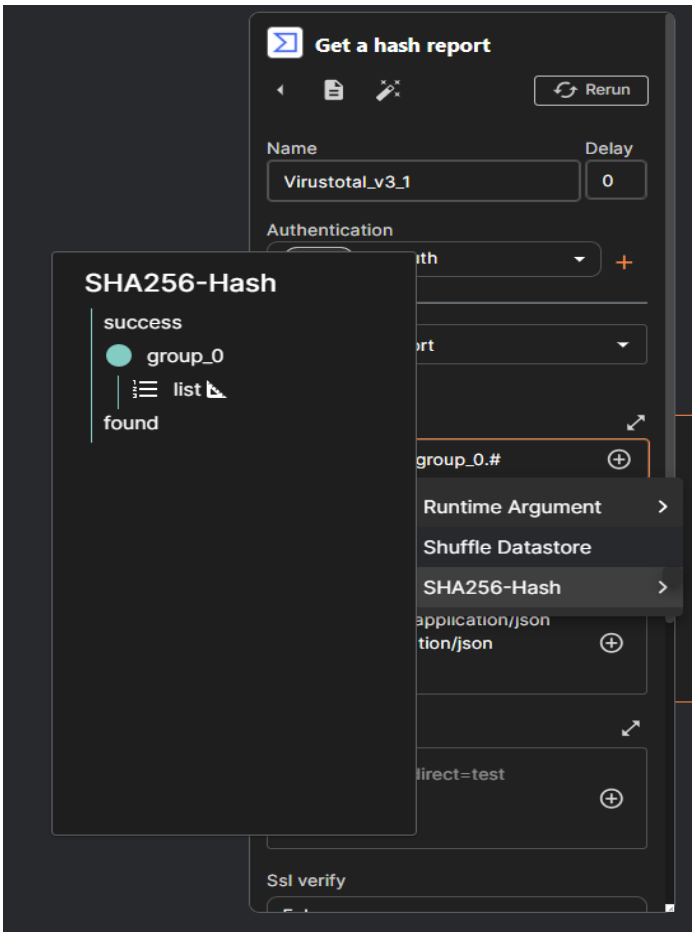
SHA-256 succesfully captured hash.



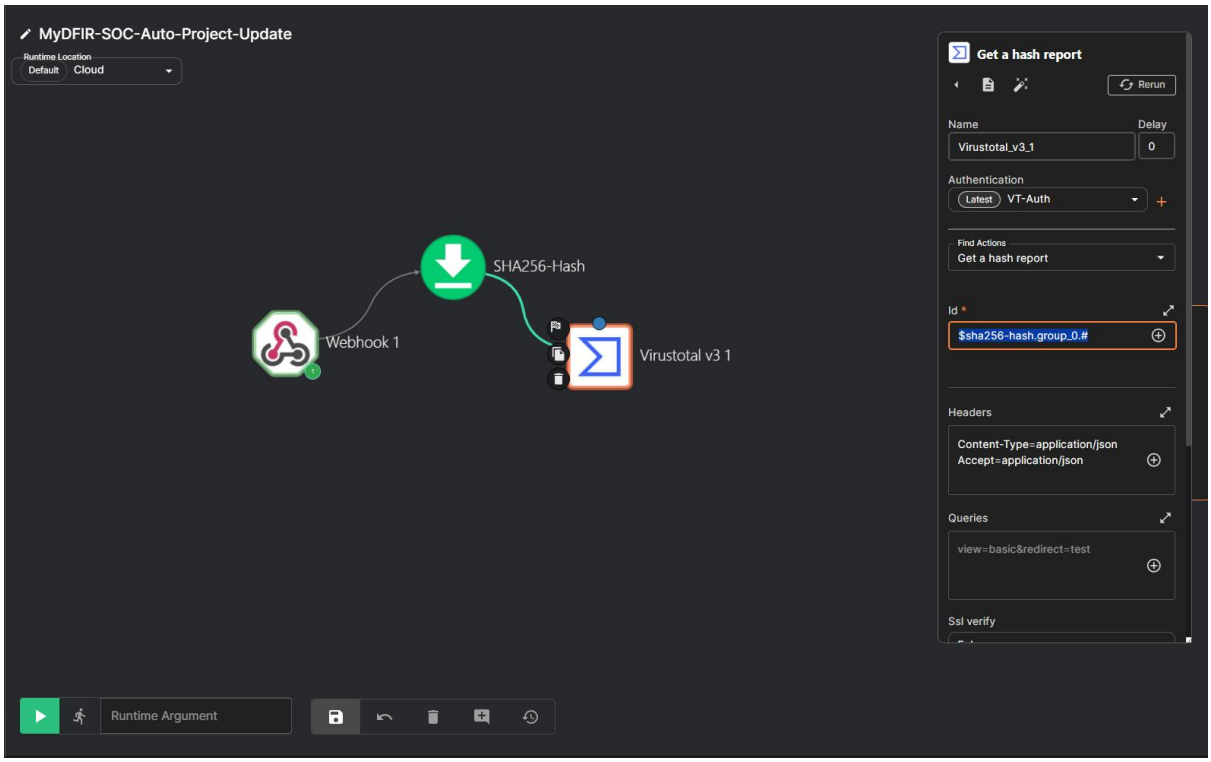
There is a some problem in finding hash report from virustotal. Let’s do some troubleshooting and find out what went wrong.

```
"url":"https://www.virustotal.com/api/v3/files/%7B'success':%20True,
%20'group_0':%20%5B'61C0810A23580CF492A..."
```

There is a problem within a URI.



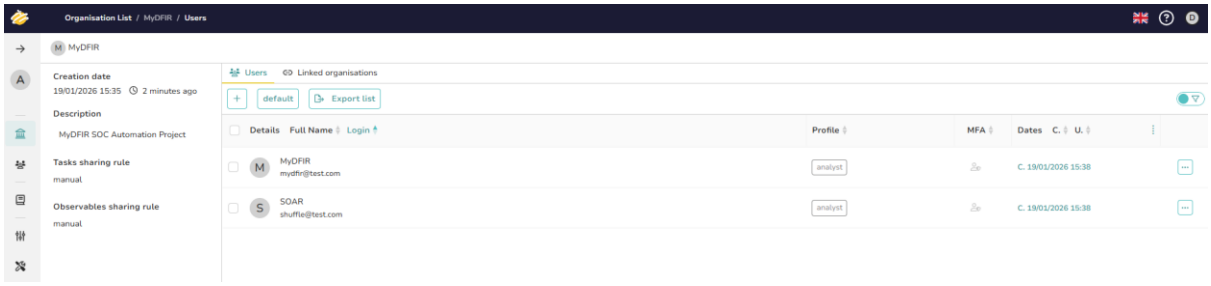
Here, I found that I have included “success”, “group_0”, “hash” and “found” in URI, that’s why virustotal is not generating a hash report. We just have to submit hash value from SHA-256 hash. So I changed value of ID = $sha256-hash.group_0.# (list) that sends only hash value to virustotal in URI. And saves workflow to take effect.



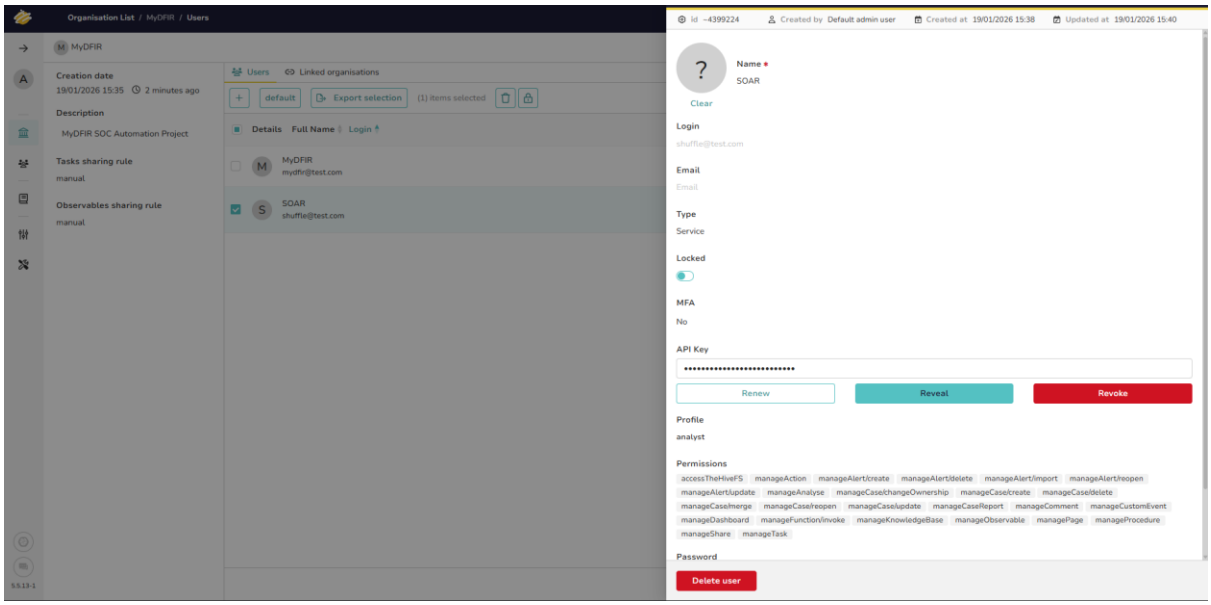
The fix involves refining the input to send only the plain hash value, resulting in a successful threat intelligence report.

I reruned workflow and find out virustotal runs successfully and it generated hash report.

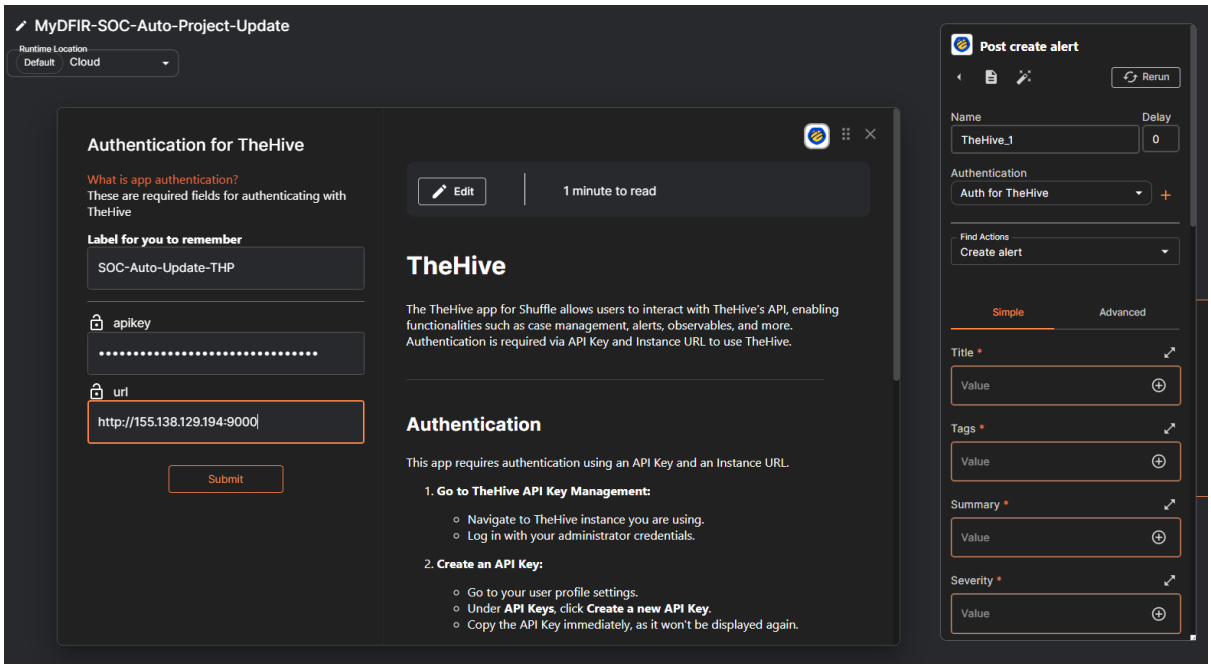
Added new organization to the Hive.



Created two user, one is normal user and another service user as analyst.



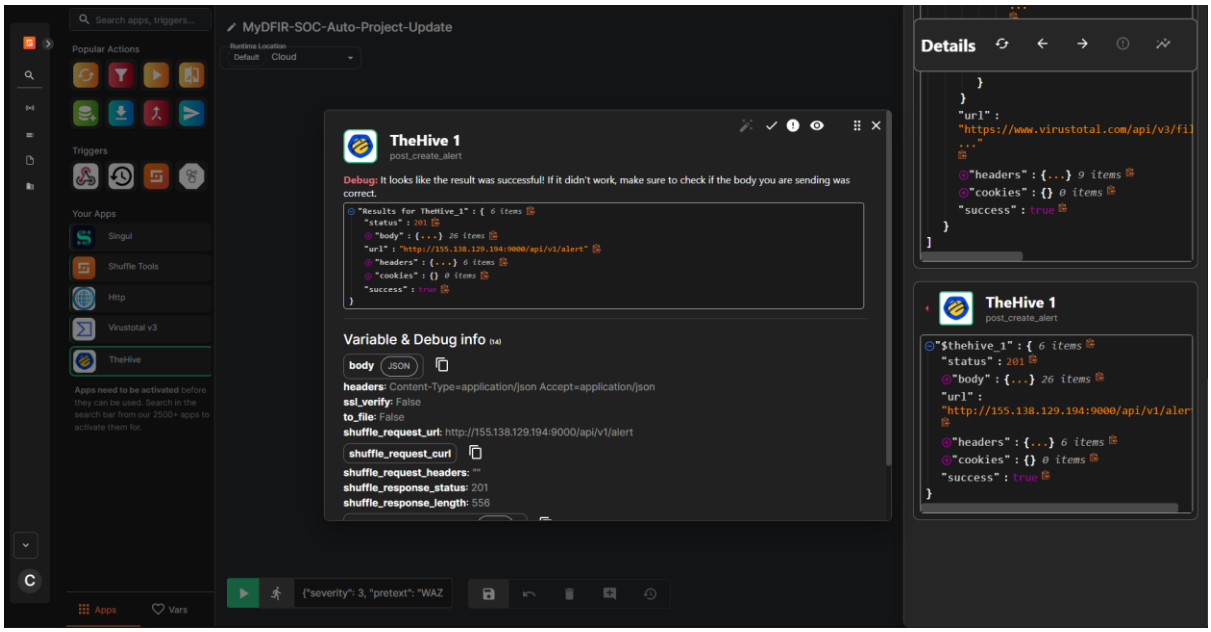
Create an service user API key and integrate it too shuffle.



Authenticating the Hive application within Shuffle using the API key establishes a secure connection between the automation workflow and the case management platform.

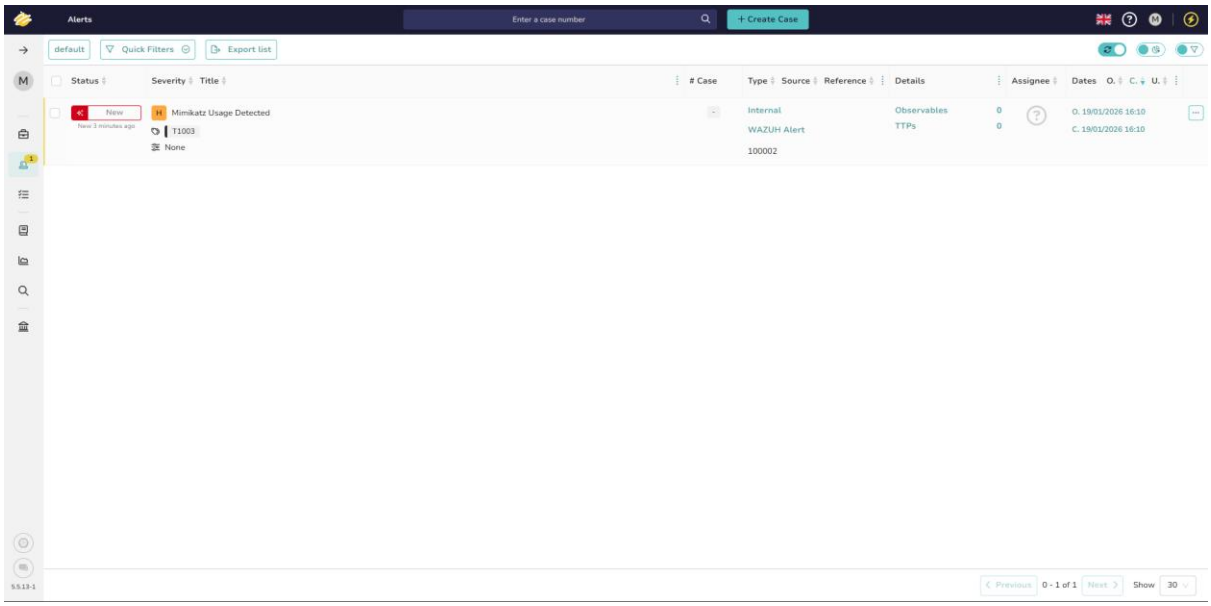
```
{
 "description": "$exec.title",
 "externallink": "${externallink}",
 "flag": false,
 "pap": 2,
 "severity": "3",
 "source": "$exec.pretext",
 "sourceRef": "$exec.rule_id",
 "status": "New",
 "summary": "Mimikatz Activity Detected on Host : $exec.rule_id$exec.text.win.system.computer",
 "tags": ["T1003"],
 "title": "$exec.title",
 "tlp": 2,
 "type": "Internal"
}
```

Configuring the "Create Alert" action with dynamic field mapping allows the workflow to automatically populate a new case in The Hive using data from the Wazuh alert and VirusTotal report.

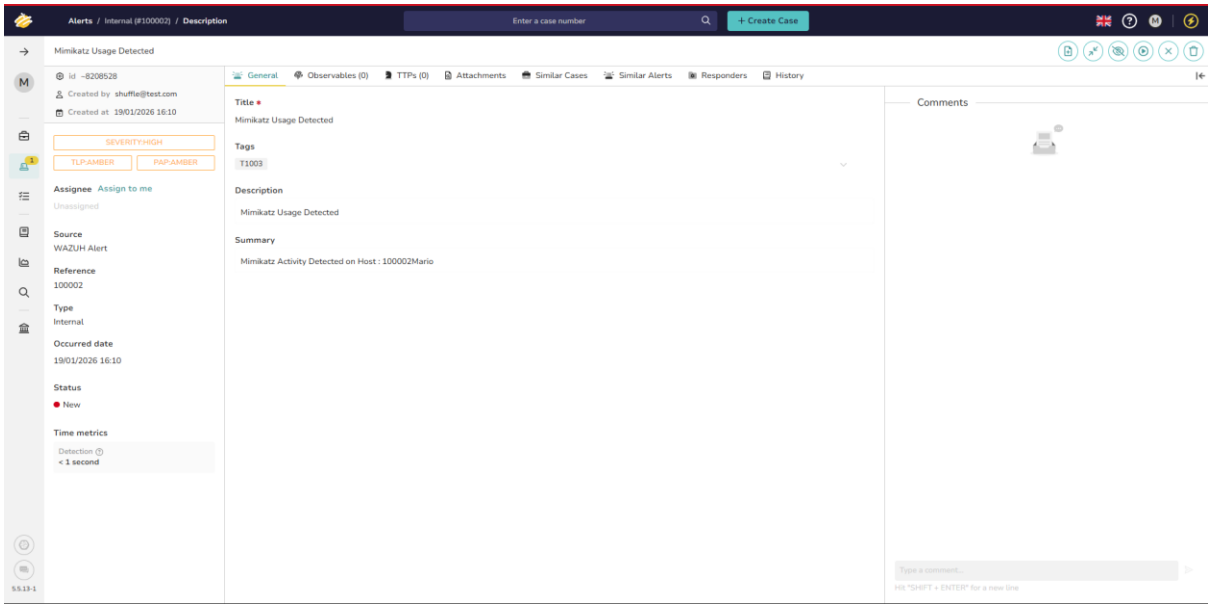


Successfully triggered alert to hive.

Now log in as normal user in Hive to see, if user got an alert or not.

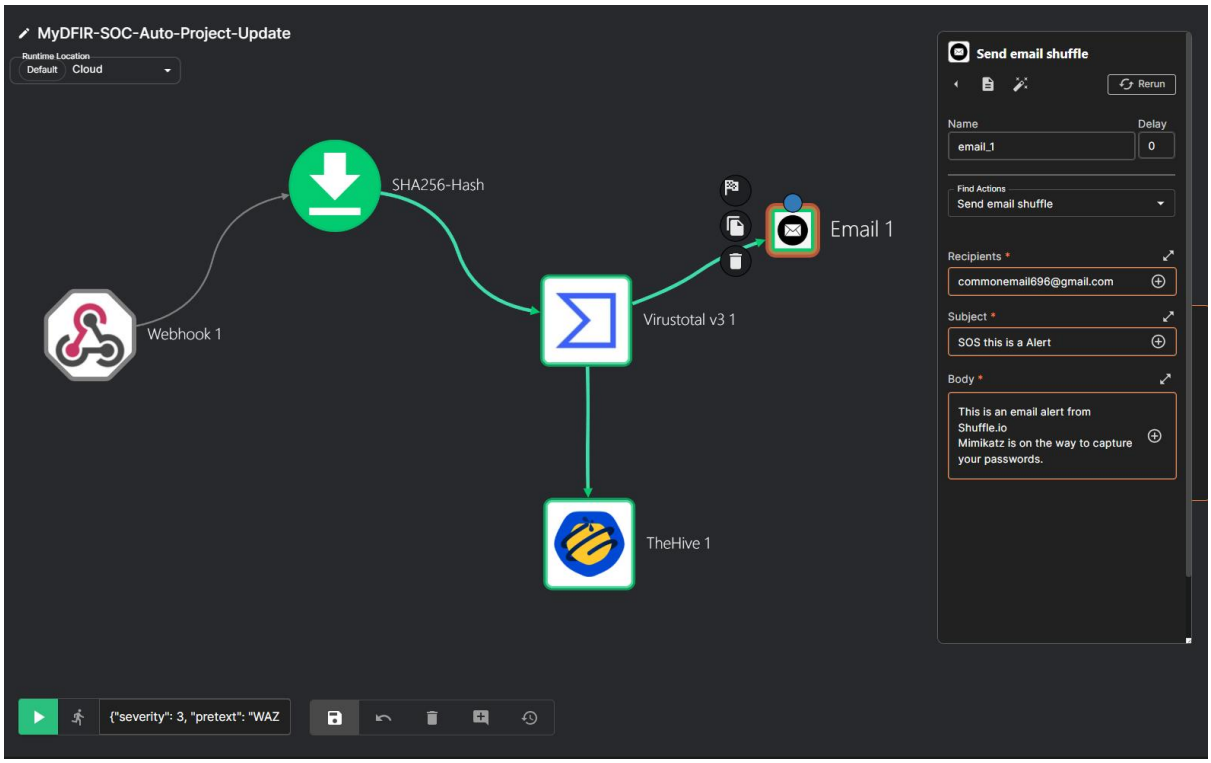


User successfuly get an alert to work on.

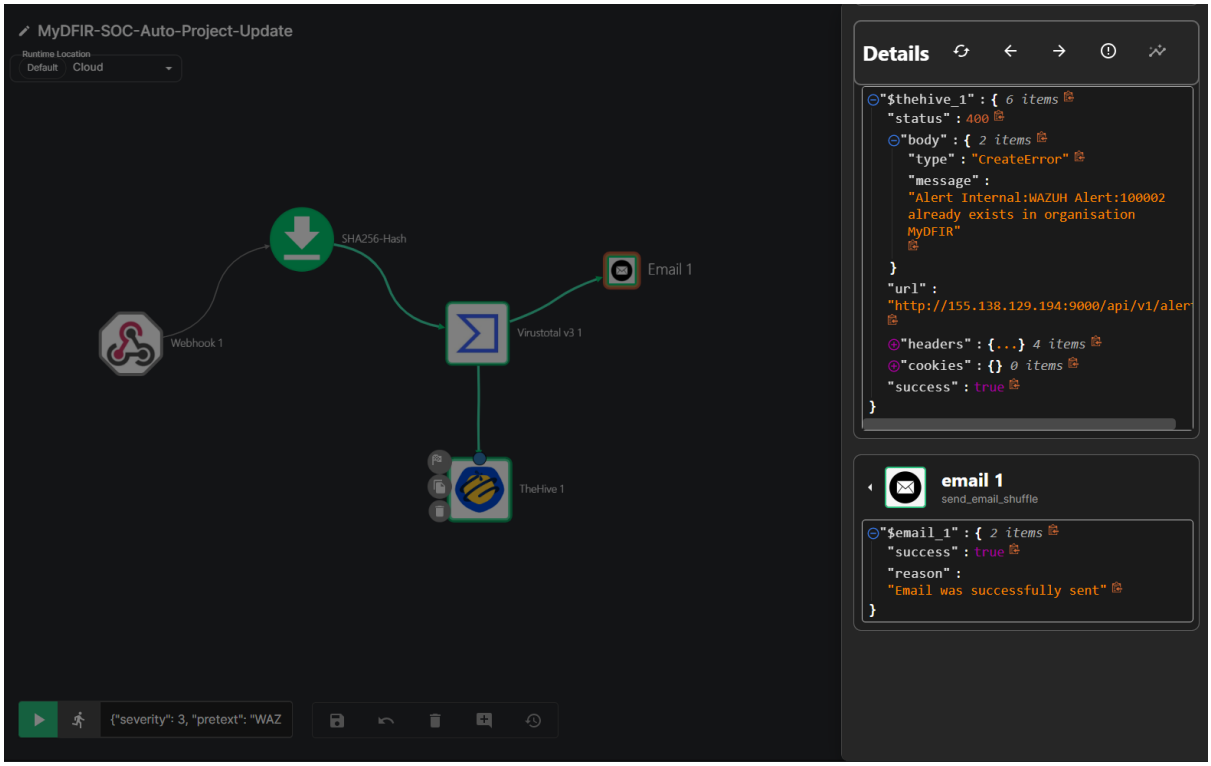


Alert details. Logging into The Hive as an analyst and viewing the auto-generated alert validates the entire workflow from the user's perspective, showing a ready-to-investigate case.

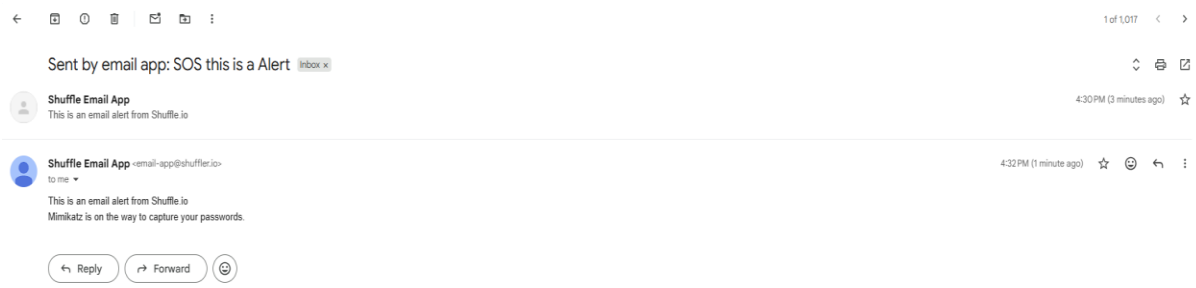
To send an email to SOC Analyst.



Adding and configuring an Email node in Shuffle creates a redundant notification channel, ensuring analysts receive prompt alerts outside the primary tool interface.



We can see that email is successfully delivered. Let’s check on inbox and verify.



The delivered email in the analyst's inbox represents the final output of the automated pipeline, completing the cycle from detection to analyst notification.

Conclusion & Key Learnings

Project Results

The project successfully established a closed-loop security cycle, validating the entire incident lifecycle from ingestion to notification. When a Mimikatz execution was detected on the Windows endpoint:

- **Wazuh** triggered a high-severity custom alert based on behavioral artifacts.
- **Shuffle** extracted the file hash and executed an automated reputation check via **VirusTotal.**
- **TheHive** consolidated the alert and intelligence into a centralized case for investigation.
- Email Notification ensured near-instant analyst awareness through a redundant communication channel.

Task	Manual Process (Analyst)	Automated Workflow (Shuffle/TheHive)	Improvement
Threat Enrichment	5–15 Minutes	< 2 Seconds	~450x Faster
Incident Logging	10–15 Minutes	Instant (API-driven)	100% Reduction
Notification	Dependent on Dashboard	Immediate (SMTP/API)	Instant Awareness
Total Response Time	15–30 Minutes	< 5 Seconds	99% MTTR Drop

Lessons Learned

- Strategic Log Ingestion & Normalization: I mastered the "Log-to-Alert" pipeline by applying Strategic Minimization—filtering data at the source to eliminate noise while preserving critical visibility. This taught me to normalize raw JSON telemetry into searchable fields, the foundation of detection engineering.

- Automation as an MTTR Force Multiplier: The project demonstrated a drastic reduction in Mean Time to Response (MTTR). Automating threat intelligence replaced minutes of manual investigation with sub-second execution. Layering multiple features (auto-case creation and enrichment) proves that orchestration allows a single analyst to handle the workload of an entire team.
- API Orchestration & Data Sanitization: Troubleshooting the integration between Wazuh and VirusTotal highlighted that data sanitization is the most critical step in SOAR—ensuring only clean, formatted indicators are passed between platforms to prevent workflow failures.
- Infrastructure Resilience: Beyond security logic, I gained practical experience in Service Hardening. Configuring backend dependencies like Cassandra and Elasticsearch reinforced the necessity of building detection tools on a high-availability, properly permissioned infrastructure.

Final Thoughts

This lab simulates a real-world enterprise environment, moving beyond simple detection into the "response" side of cybersecurity. It proves that automation is not just a luxury but a necessity for modern security teams to remain proactive against fast-moving threats.