Behaviour-Based Endpoint Threat Detection and Automated Containment Using Wazuh and Sysmon

A Falcon-Style EDR Simulation Lab

Author: Yatish Waghmare

MSc Cyber Security (Infrastructures Security)

2026

Yatish Waghmare

1 Purpose of the Lab

The purpose of this lab exercise is to design and evaluate a behaviour-based endpoint detection and response (EDR) environment that simulates key operational capabilities of **CrowdStrike Falcon Complete** using open-source technologies.

CrowdStrike Falcon Complete is a managed detection and response (MDR) platform that provides continuous monitoring, behavioural detection, automated response, and incident remediation across enterprise endpoints.

In order to understand how such enterprise platforms operate, this lab recreates several core Falcon Complete capabilities using the following tools:

- **Wazuh** SIEM and detection engine
- **Sysmon** endpoint telemetry collection
- Custom correlation rules for behavioural detection
- Automated host containment scripts
- Real-time SOC alerting using **Telegram**

Through simulated attacker activity on a monitored **Windows** system, the environment demonstrates how security telemetry is analysed, correlated, and acted upon automatically.

This provides a practical representation of the detection, investigation, and response workflow typically handled by Falcon Complete's managed security operations team.

Yatish Waghmare

2 Objectives

The primary objectives of this lab are to reproduce several security capabilities commonly provided by **CrowdStrike Falcon Complete**.

- **Simulate behavioural endpoint monitoring**
  Collect detailed endpoint telemetry using **Sysmon** and forward events to the **Wazuh** detection engine.
- **Replicate Falcon-style threat detection**
  Implement custom correlation rules capable of identifying suspicious behaviour such as encoded **PowerShell** execution and reconnaissance activity.
- **Demonstrate multi-stage attack correlation**
  Detect attacker behaviour sequences within a defined timeframe similar to Falcon's behavioural detection models.
- **Implement automated host containment**
  Automatically isolate compromised systems to prevent lateral movement and data exfiltration.
- **Simulate SOC alerting workflows**
  Send real-time alerts to a simulated SOC channel using **Telegram** messaging.

Overall, the objective of this lab is to recreate the core operational workflow of **CrowdStrike Falcon Complete** using open-source components.

Yatish Waghmare

3 Falcon Complete Capability Simulation

CrowdStrike Falcon Complete provides several enterprise security capabilities including behavioural detection, automated response, and continuous monitoring. This lab replicates many of those capabilities using **Wazuh** and custom response scripts.

Falcon Complete vs. Lab Implementation	
Endpoint Telemetry Collection	**Sysmon** + Windows Event Logs
Behaviour-based Detection	Custom **Wazuh** correlation rules
Multi-stage Attack Detection	Rule 990003 correlation
Host Isolation	Network adapter disable
Malicious Process Termination	**taskkill** automation
SOC Alert Notification	**Telegram API** alerts
Automated Incident Response	**Wazuh Active Response**

Although this lab uses open-source tools, the operational workflow closely mirrors the detection and response model used by **CrowdStrike Falcon Complete's** managed threat hunting teams.

Yatish Waghmare

4 Lab Architecture

The lab environment was designed to simulate the operational workflow of an enterprise **endpoint detection and response (EDR)** platform similar to **CrowdStrike Falcon Complete**. The architecture demonstrates how endpoint telemetry is collected, analysed, and used to trigger automated containment actions during a simulated security incident.

The environment consists of two virtual machines connected within the same host-only network:

VM Role	**Hostname**	**IP Address**
Windows Server Endpoint	DC1	192.168.56.128
Ubuntu Wazuh Manager	Wazuh-Manager	192.168.56.131

The **Windows Server** system acts as the monitored endpoint where simulated attacker activity is generated. The **Wazuh agent** installed on this system collects endpoint telemetry including Windows Security Event Logs and **Sysmon** process monitoring events.

These logs are forwarded to the **Wazuh manager**, which performs behavioural analysis using custom detection rules. When suspicious activity is identified, the system triggers automated containment actions through **Wazuh Active Response**.

Additionally, a notification mechanism was implemented to simulate a **Security Operations Centre (SOC)** alerting workflow. When containment actions are triggered, the endpoint sends a message through the **Telegram API** to notify the SOC of the security incident.

Figure 1 illustrates the architecture of the Falcon Complete simulation environment used in this lab.

Yatish Waghmare

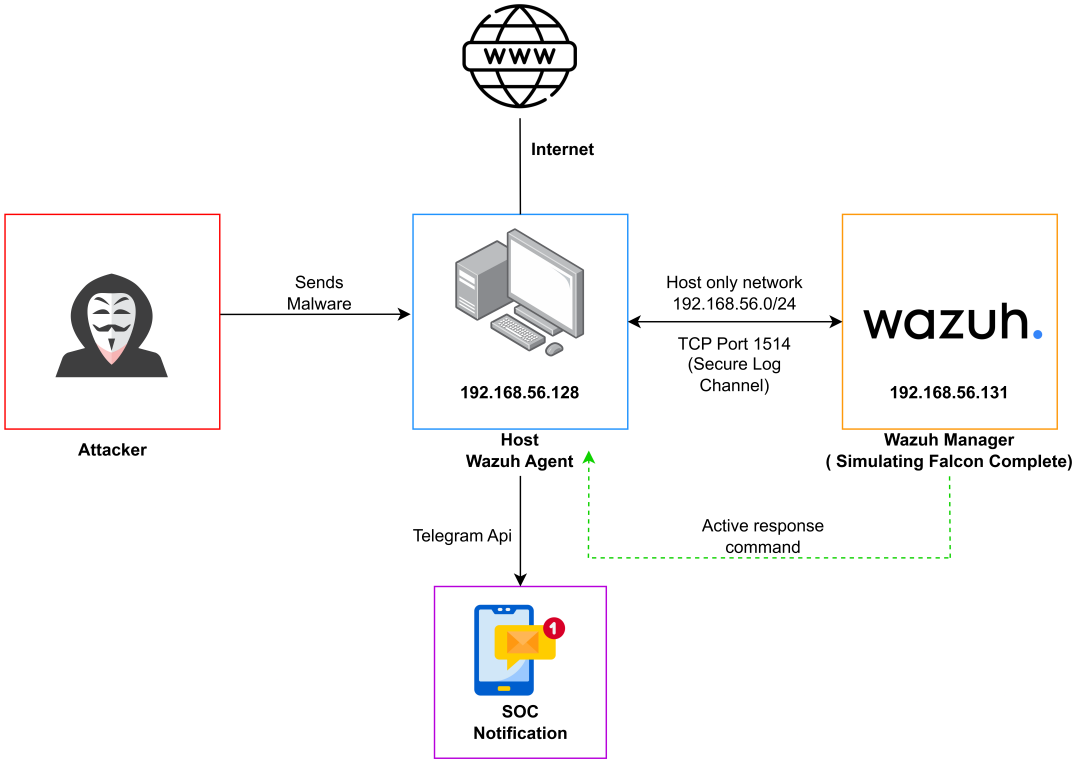


Figure 1: Falcon Complete simulation environment architecture

Component	**IP Address**	**Role**
Attacker	Simulated	Generates malicious activity targeting the endpoint
Windows Endpoint	192.168.56.128	Runs **Wazuh Agent** and **Sysmon**
Wazuh Manager	192.168.56.131	Detection engine simulating **Falcon Complete**
SOC Notification	Telegram API	Receives containment alerts

Yatish Waghmare

5 Lab Installation and Configuration

This section describes how the Wazuh detection environment was installed and configured in order to simulate the behaviour of CrowdStrike Falcon Complete using open-source tools.

The installation process includes deploying the Wazuh manager on an Ubuntu server and installing the Wazuh agent on a monitored Windows endpoint.

5.1 Installing the Wazuh Manager

```
curl -sO https://packages.wazuh.com/4.x/wazuh-install.sh
sudo bash wazuh-install.sh -a
```

After installation completed, the Wazuh manager service and dashboard were successfully started, confirming that the detection platform was operational.

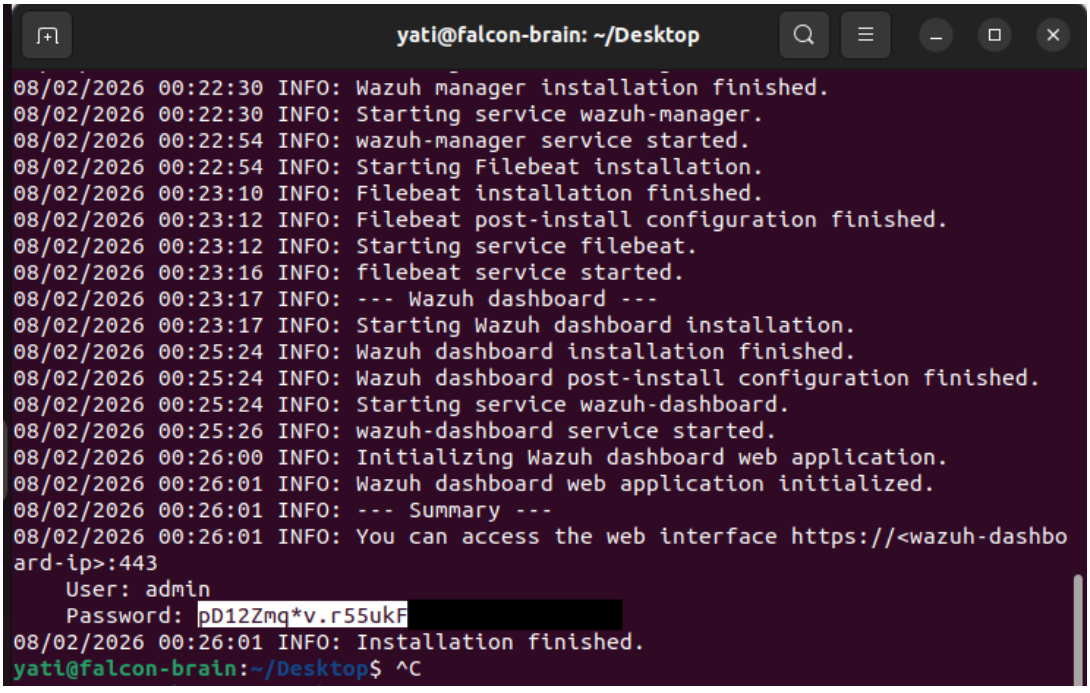


Figure 2: Wazuh Manager installation completed

Yatish Waghmare

5.2 Registering the Endpoint Agent

After installing the manager, a Windows endpoint was registered as a monitored agent. The agent registration was performed using the Wazuh agent management utility:

```
sudo /var/ossec/bin/manage_agents
```

A new agent entry was created with the following information:

- Agent Name: WIN-01
- Agent ID: 001

5.3 Extracting the Agent Authentication Key

After registering the agent, an authentication key was generated to allow the endpoint to securely connect to the Wazuh manager.

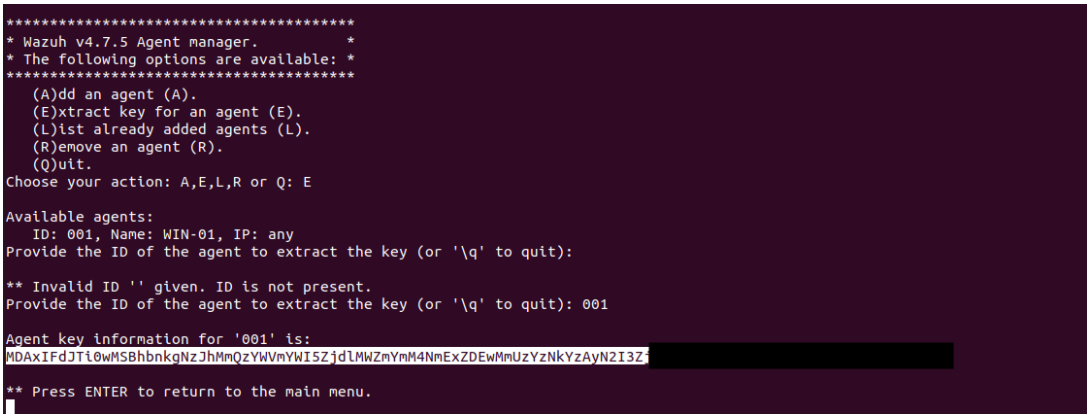


Figure 3: Extracting the Wazuh agent authentication key

The generated key is used during the Windows agent configuration process to authenticate the endpoint with the Wazuh manager.

5.4 Configuring the Wazuh Agent on the Windows Endpoint

The Wazuh agent was installed on the Windows Server endpoint and configured using the authentication key obtained from the Wazuh manager.

The manager IP address was configured as:

Yatish Waghmare

```
192.168.56.131
```

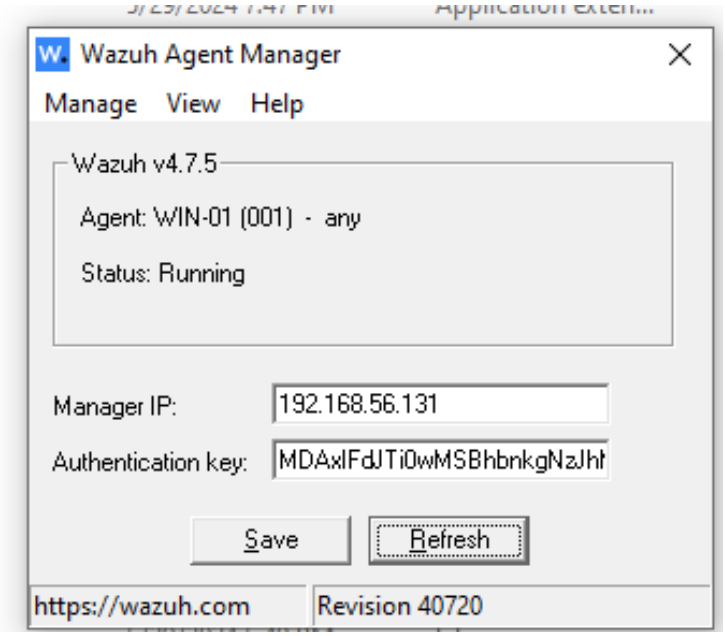


Figure 4: Wazuh agent connected to the manager

The screenshot shows that the Windows endpoint agent is running and successfully connected to the Wazuh manager.

5.5 Verifying Agent Communication

After configuration, the Wazuh agent service was started on the Windows endpoint and communication with the Wazuh manager was verified. The agent status appeared as **Running**, confirming successful connectivity between the monitored endpoint and the detection engine.

At this stage, the lab environment was ready for implementing behaviour-based detection and automated containment mechanisms to simulate the capabilities of **CrowdStrike Falcon Complete**.

Yatish Waghmare

6 Endpoint Telemetry Collection

Enterprise endpoint detection and response platforms such as **CrowdStrike Falcon Complete** rely on detailed endpoint telemetry to detect suspicious behaviour. In this lab environment, telemetry was collected using the **Wazuh agent**, Windows Event Logs, and **Sysmon**.

These data sources provide visibility into system activity including process creation, command execution, and security events.

6.1 Windows Security Event Monitoring

The Wazuh agent was configured to monitor Windows Security Event Logs. This configuration allows the detection engine to collect events related to authentication, privilege use, and system activity.

The following configuration was added to the Wazuh agent configuration file:

```
<localfile>
  <log_format>eventchannel</log_format>
  <location>Security</location>
</localfile>
```

This configuration enables the Wazuh agent to collect Windows Security Event Logs and forward them to the Wazuh manager for analysis.

6.2 Sysmon Installation

Sysmon (System Monitor) was installed on the Windows endpoint to provide detailed telemetry about system activity. Sysmon generates logs related to process creation, command-line execution, and network connections.

Sysmon was installed using the following command:

```
sysmon.exe -i sysmonconfig.xml
```

Once installed, Sysmon begins generating detailed system activity logs that are collected by the Wazuh agent.

Yatish Waghmare

6.3 Telemetry Forwarding

After collection, all endpoint telemetry is forwarded from the Windows endpoint to the Wazuh manager through the secure host-only network using **TCP port 1514**.

The Wazuh manager then analyses these logs using correlation rules to detect suspicious behaviour patterns that simulate the detection capabilities of **CrowdStrike Falcon Complete**.

7 Behaviour-Based Detection and Correlation Rules

Modern Endpoint Detection and Response (EDR) platforms such as **CrowdStrike Falcon Complete** rely on behaviour-based detection techniques to identify malicious activity. Instead of detecting individual signatures, these systems analyse sequences of suspicious actions that occur within a short time period.

To simulate this approach, custom correlation rules were implemented in the Wazuh manager configuration file:

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

These rules correlate multiple Sysmon and Windows security events to detect multi-stage attacker behaviour.

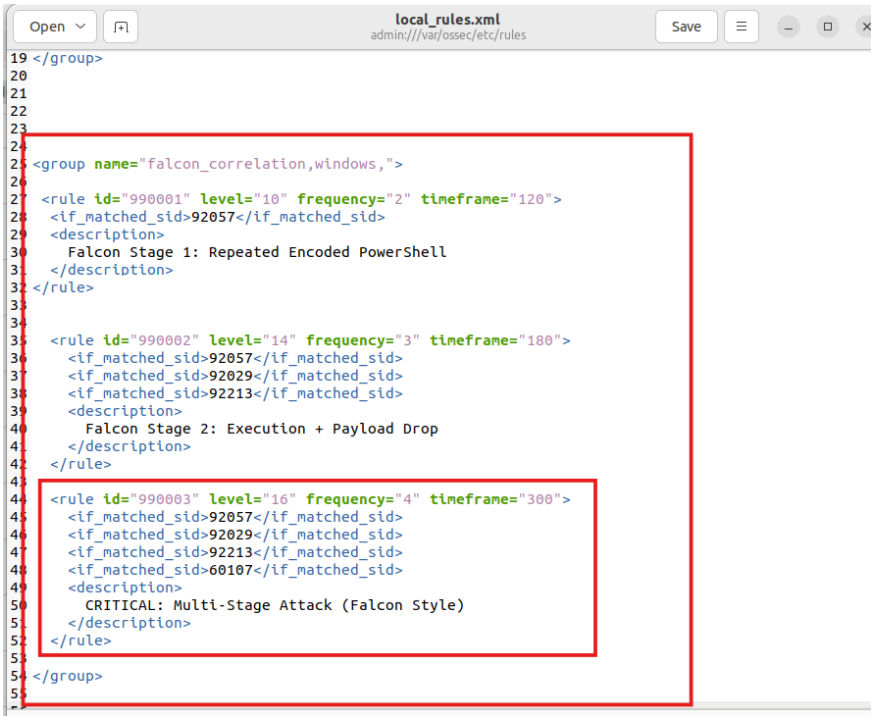


Figure 5: Custom Falcon-style correlation rules implemented in Wazuh

Yatish Waghmare

7.1 Stage 1: Repeated Encoded PowerShell

The first rule detects repeated execution of encoded PowerShell commands. Attackers commonly use encoded PowerShell to hide malicious scripts and evade detection.

```
<rule id="990001" level="10" frequency="2" timeframe="120">
  <if_matched_sid>92057</if_matched_sid>
  <description>
    Falcon Stage 1: Repeated Encoded PowerShell
  </description>
</rule>
```

This rule triggers when the encoded PowerShell detection rule (SID 92057) occurs twice within a 120-second timeframe. Such behaviour may indicate malicious script execution or automated attack activity.

7.2 Stage 2: Execution and Payload Activity

The second rule correlates additional suspicious activity such as command execution and payload delivery. This stage represents the attacker performing further actions after initial PowerShell execution.

```
<rule id="990002" level="14" frequency="3" timeframe="180">
  <if_matched_sid>92057</if_matched_sid>
  <if_matched_sid>92029</if_matched_sid>
  <if_matched_sid>92213</if_matched_sid>
  <description>
    Falcon Stage 2: Execution + Payload Drop
  </description>
</rule>
```

This rule triggers when multiple suspicious events occur within a short period of time. These correlated events indicate that the attacker may be attempting to execute commands and deploy additional malicious payloads.

Yatish Waghmare

7.3 Stage 3: Multi-Stage Attack Detection

The final rule represents a critical multi-stage attack detection. It combines multiple suspicious behaviours to identify a coordinated attack sequence.

```
<rule id="990003" level="16" frequency="4" timeframe="300">
  <if_matched_sid>92057</if_matched_sid>
  <if_matched_sid>92029</if_matched_sid>
  <if_matched_sid>92213</if_matched_sid>
  <if_matched_sid>60107</if_matched_sid>
  <description>
    CRITICAL: Multi-Stage Attack (Falcon Style)
  </description>
</rule>
```

This rule triggers when four suspicious events are detected within a 300-second timeframe. Once this condition is satisfied, the Wazuh manager generates a high-severity alert indicating a possible multi-stage attack.

Such correlation-based detection closely resembles the behaviour analysis techniques used by **CrowdStrike Falcon Complete**, where multiple attacker actions are analysed together to identify complex attack chains.

Yatish Waghmare

8 Automated Response and Host Containment

Enterprise endpoint detection platforms such as **CrowdStrike Falcon Complete** provide automated containment capabilities that allow security teams to stop attacks immediately after detection. Instead of relying solely on alerts, the system can automatically execute defensive actions on compromised hosts.

In this lab environment, automated containment was implemented using the **Wazuh Active Response** framework. When the multi-stage attack detection rule (990003) is triggered on the Wazuh manager running on the Ubuntu server, the manager sends an active response command to the monitored Windows endpoint.

8.1 Active Response Configuration

The containment command was defined on the monitored Windows endpoint by modifying the Wazuh agent configuration file located at:

```
C:\Program Files (x86)\ossec-agent\ossec.conf
```

The following command configuration was added:

```
<command>
  <name>contain-host</name>
  <executable>Contain-HOST.cmd</executable>
  <timeout_allowed>no</timeout_allowed>
</command>
```

This configuration registers the containment script with the Wazuh agent so that it can be executed when the Wazuh manager triggers an Active Response event.

When the correlation rule 990003 is triggered on the Wazuh manager, the manager sends a response command to the Windows endpoint. The Wazuh agent receives this instruction and executes the containment script locally on the host.

8.2 Containment Script Deployment

The containment scripts were deployed on the Windows endpoint inside the Wazuh Active Response directory:

```
C:\Program Files (x86)\ossec-agent\active-response\bin\
```

Yatish Waghmare

The following scripts were implemented:

- `Contain-HOST.cmd` – master containment script
- `Nat-Killer.cmd` – network isolation script
- `pw-kill.cmd` – process containment script

The master containment script coordinates the containment workflow and sequentially executes the helper scripts responsible for isolating the host and terminating malicious processes.

8.3 Master Containment Script

The `Contain-HOST.cmd` script acts as the primary containment script executed by the Wazuh agent.

```
@echo off
setlocal

set BASEDIR=C:\Program Files
   (x86)\ossec-agent\active-response\bin

echo ===== CONTAINMENT START =====

:: PRIORITY 1 - TELEGRAM ALERT
call "%BASEDIR%\q1.cmd"

ping 127.0.0.1 -n 3 >nul

:: PRIORITY 2 - NETWORK ISOLATION
call "%BASEDIR%\Nat-Killer.cmd"

ping 127.0.0.1 -n 2 >nul

:: PRIORITY 3 - TERMINATE ATTACKER SHELLS
call "%BASEDIR%\pw-kill.cmd"

echo ===== CONTAINMENT COMPLETE =====

endlocal
exit /b 0
```

Yatish Waghmare

This script executes the containment workflow in three stages:

- Sends a notification alert to the SOC channel
- Disables the external network adapter to isolate the host
- Terminates command-line interpreters used by the attacker

8.4 Network Isolation Script

The `Nat-Killer.cmd` script isolates the compromised endpoint by disabling the NAT network adapter.

```
@echo off
setlocal

set LOCK=C:\containment.lock
set ADAPTER=Ethernet0

if exist "%LOCK%" (
 exit /b 0
)

echo locked > "%LOCK%"

C:\Windows\Sysnative\netsh.exe interface set interface
   name="%ADAPTER%" admin=disable

endlocal
exit /b 0
```

A lock file is created to prevent repeated execution of the containment action.

8.5 Process Containment Script

The `pw-kill.cmd` script terminates command-line interpreters that may be used by attackers during post-exploitation activity.

```
@echo off
taskkill /F /IM powershell.exe
taskkill /F /IM pwsh.exe
taskkill /F /IM cmd.exe
exit /b 0
```

Yatish Waghmare

By terminating these processes, the attacker is prevented from continuing command execution on the compromised host.

8.6 Telegram Containment alert script

This script is responsible for sending a real-time containment alert to the Security Operations Center (SOC) through the Telegram Bot API.

```
@echo off
setlocal

powershell -ExecutionPolicy Bypass -Command ^
"$msg=' HOST CONTAINMENT STARTED: ' + $env:COMPUTERNAME + ' at '
   + (Get-Date); ^
try { ^
Invoke-RestMethod -Uri
   'https://api.telegram.org/botYOUR_NEW_TOKEN/sendMessage'
   -Method Post -Body @{chat_id='YOUR_CHAT_ID'; text=$msg} ^
} catch { }"

endlocal
exit /b 0
```

8.7 Containment Workflow

The automated containment workflow implemented in this lab follows the sequence below:

1. Suspicious activity is generated on the Windows endpoint.
2. Sysmon logs are collected by the Wazuh agent.
3. The Wazuh manager analyses the telemetry and triggers correlation rule `990003`.
4. The Active Response framework sends a containment command to the Windows endpoint.
5. The Wazuh agent executes `Contain-HOST.cmd`.
6. The containment script sends an alert notification, disables the NAT adapter, and terminates attacker command shells.

Yatish Waghmare

This automated response prevents attackers from continuing command execution and isolates the compromised endpoint from external communication.

The containment workflow implemented in this lab closely replicates the automated incident response capabilities provided by enterprise endpoint detection platforms such as **CrowdStrike Falcon Complete**.

9 Attack Simulation

To evaluate the behaviour-based detection capabilities of the implemented security environment, a controlled attack simulation was executed on the monitored Windows endpoint. The purpose of the simulation was to generate suspicious system activity that would trigger the custom detection rules configured in the Wazuh manager.

The simulation was implemented using an automated PowerShell script designed to mimic attacker behaviour commonly observed during post-exploitation. The script generates encoded PowerShell execution, hidden PowerShell processes, reconnaissance commands, and outbound network connections.

9.1 Automated Attack Simulation Script

The following PowerShell script was executed on the Windows endpoint to simulate attacker behaviour over a five-minute period.

```
Write-Host "=== 5 Minute Attack Simulation Started ==="

$startTime = Get-Date
$endTime = $startTime.AddMinutes(5)

$parentPID = $PID

while ((Get-Date) -lt $endTime) {

    # Encoded PowerShell Execution
    $payload = "Start-Sleep 2"
    $bytes = [System.Text.Encoding]::Unicode.GetBytes($payload)
    $encoded = [Convert]::ToBase64String($bytes)
    Start-Process powershell.exe -ArgumentList "-EncodedCommand
       $encoded"
    Start-Sleep 5

    # Hidden PowerShell Execution
```

Yatish Waghmare

```
    Start-Process powershell.exe -ArgumentList "-WindowStyle
       Hidden -Command '"Start-Sleep 3'""
    Start-Sleep 5

    # Recon Activity
    whoami | Out-Null
    whoami /priv | Out-Null
    hostname | Out-Null
    ipconfig | Out-Null
    Start-Sleep 5

    # Network Callout
    try {
        Invoke-WebRequest -Uri "http://example.com"
           -UseBasicParsing | Out-Null
    } catch {}

    Start-Sleep 5
}

Write-Host "=== 5 Minutes Reached     Hard Killing All
   Simulation Shells ==="

Get-Process powershell,pwsh -ErrorAction SilentlyContinue |
Where-Object { $_.Id -ne $parentPID } |
Stop-Process -Force

Write-Host "=== Simulation Fully Terminated ==="
```

9.2 Simulated Attacker Behaviour

The script automatically generates multiple behaviours typically associated with malicious activity:

- Encoded PowerShell execution used to obfuscate commands
- Hidden PowerShell processes used for stealth execution
- System reconnaissance commands such as `whoami`, `hostname`, and `ipconfig`
- Outbound network communication to simulate command-and-control activity

Yatish Waghmare

These actions generate telemetry events that are captured by Sysmon and forwarded to the Wazuh manager through the Wazuh agent.

9.3 Detection and Automated Containment

During the execution of the attack simulation script, multiple suspicious activities were generated within a short time window. These activities included encoded PowerShell execution, hidden PowerShell processes, system reconnaissance commands, and outbound network communication attempts.

The Sysmon service running on the Windows endpoint captured these events and forwarded the telemetry to the Wazuh manager through the Wazuh agent. The Wazuh manager analysed the incoming events using the custom behaviour-based correlation rules configured in the system.

Once the sequence of suspicious behaviours exceeded the defined correlation threshold, the custom correlation rule `990003` was triggered. This rule represents a simulated multi-stage attack scenario designed to mimic the behaviour detection mechanisms used by enterprise endpoint detection platforms.

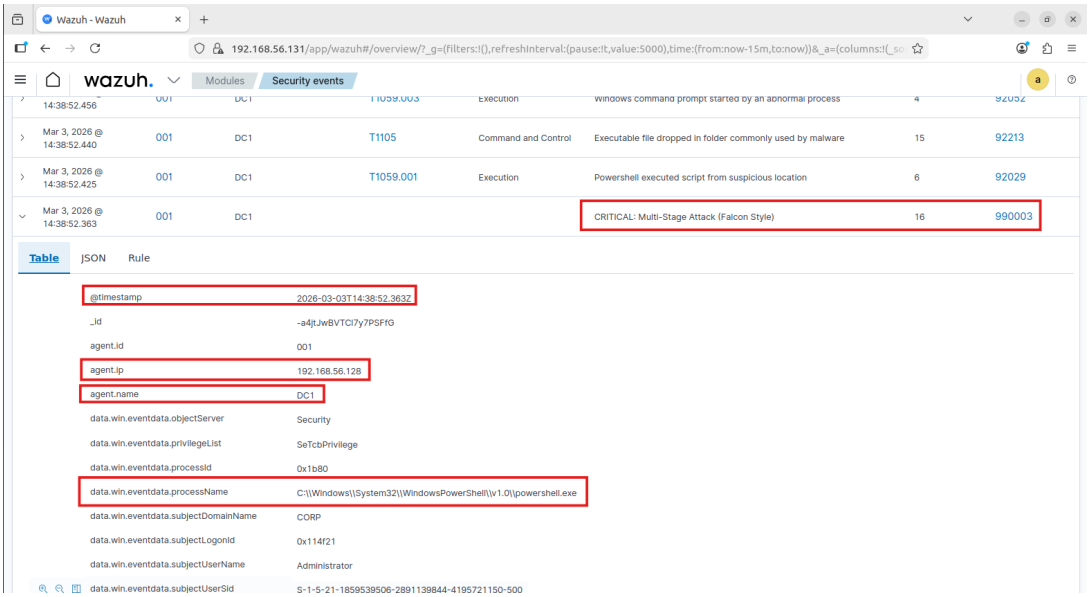


Figure 6: Critical multi-stage attack alert generated by Wazuh

The alert generated by Wazuh includes contextual information such as the timestamp of the event, the affected endpoint, the executing process, and the rule identifier associated with the detected behaviour.

Once the correlation rule was triggered, the Wazuh Active Response framework automati-

Yatish Waghmare

cally initiated the containment procedure. The Wazuh manager sent a response command to the Windows endpoint, instructing the Wazuh agent to execute the containment script `Contain-HOST.cmd`.

The containment script was executed locally on the Windows host and coordinated the containment workflow by invoking additional scripts responsible for network isolation and process termination. As a result, the compromised host was automatically isolated from the network and attacker command execution was terminated without requiring manual intervention.

9.4 System State Before and After the Attack

Before executing the attack simulation, the monitored Windows endpoint was operating normally and maintained an active network connection to the internal network. As shown in Figure 7, the endpoint was successfully connected to the network, allowing normal communication with the Wazuh manager.

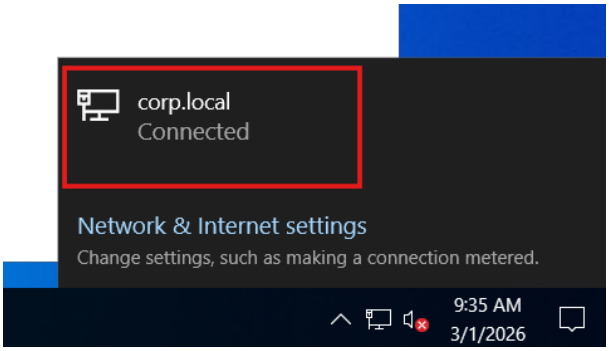


Figure 7: Normal network connectivity before attack simulation

After the attack simulation triggered the detection rule and the containment procedure was executed, the network adapter on the endpoint was automatically disabled. This action prevented the compromised host from maintaining network communication with other systems.

In addition to isolating the host from the network, the containment script terminated active PowerShell and command prompt processes used during the attack simulation. This ensured that the attacker could no longer execute commands on the compromised system.

Figure 8 shows the alert notification received by the security operations channel indicating that host containment had been initiated.

Yatish Waghmare

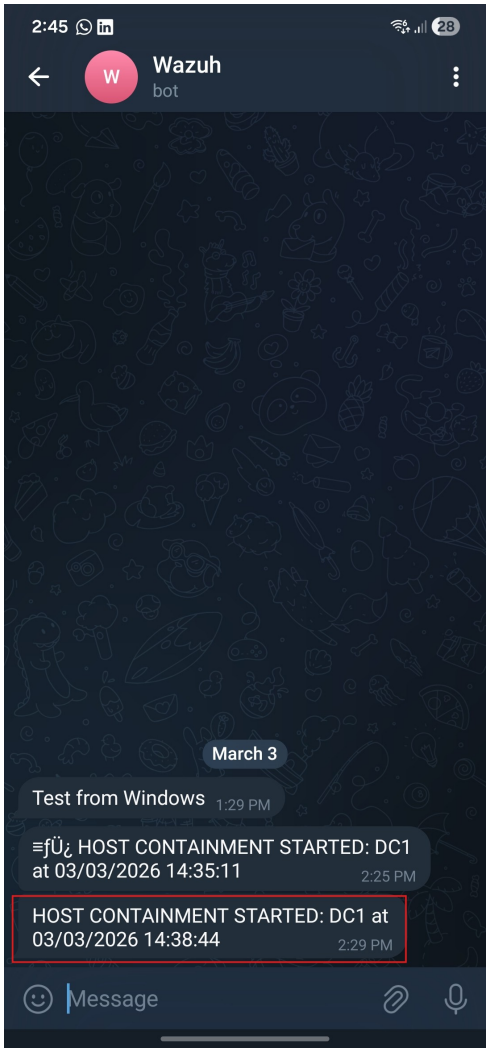


Figure 8: SOC notification confirming host containment

Following the containment process, the endpoint lost external network connectivity as shown in Figure 9. This confirms that the containment script successfully disabled the network interface, effectively isolating the compromised host.

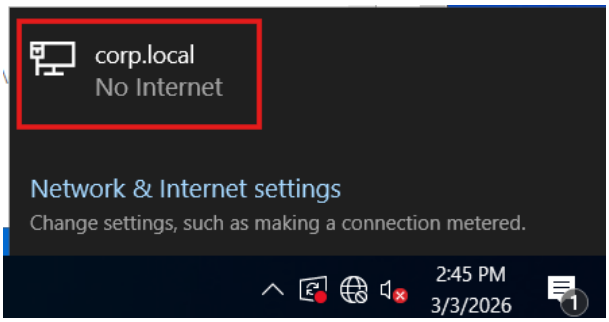


Figure 9: Endpoint network connection disabled after containment

Yatish Waghmare

This sequence demonstrates that the implemented detection and response system successfully identified the simulated attack, generated a critical alert, and automatically executed containment actions on the endpoint. The automated workflow replicates the detection and containment capabilities typically provided by enterprise endpoint detection platforms such as **CrowdStrike Falcon Complete**.

9.5 MITRE ATT&CK Technique Mapping

The attacker behaviours generated during the simulation can be mapped to techniques defined in the MITRE ATT&CK framework. This mapping helps contextualize the simulated attack within commonly observed adversary tactics and techniques.

Attack Behaviour	**MITRE Technique**	**Technique ID**
Encoded PowerShell Execution	Command and Scripting Interpreter	T1059.001
Hidden PowerShell Execution	Obfuscated / Hidden Execution	T1027
System Reconnaissance Commands	System Information Discovery	T1082
Network Callout Activity	Application Layer Protocol	T1071
Repeated Command Execution	Automated Collection / Execution	T1119

Table 1: MITRE ATT&CK mapping of simulated attacker behaviour

9.6 System State Before and After Attack Containment

Table 2 summarises the state of the monitored endpoint before the attack simulation and after the automated containment procedure was triggered.

System Behaviour	**Before Attack**	**After Containment**
Network Connectivity	Connected	Disabled / Isolated
PowerShell Processes	Normal execution	Terminated by containment script
Command Prompt Activity	Normal system usage	Command shells terminated
Wazuh Alerts	No critical alerts	Rule 990003 triggered
Active Response	Not triggered	Containment script executed
SOC Notification	No alerts	Telegram containment alert sent
Endpoint Status	Normal operation	Host isolated from external network

Table 2: Comparison of endpoint behaviour before and after automated containment

Yatish Waghmare

9.7 Malware Behaviour Verification

To further validate the behaviour of the attack simulation script, the PowerShell file used in the experiment was analysed using the VirusTotal malware analysis platform. The file was uploaded to the platform to observe how different security engines classify the script.

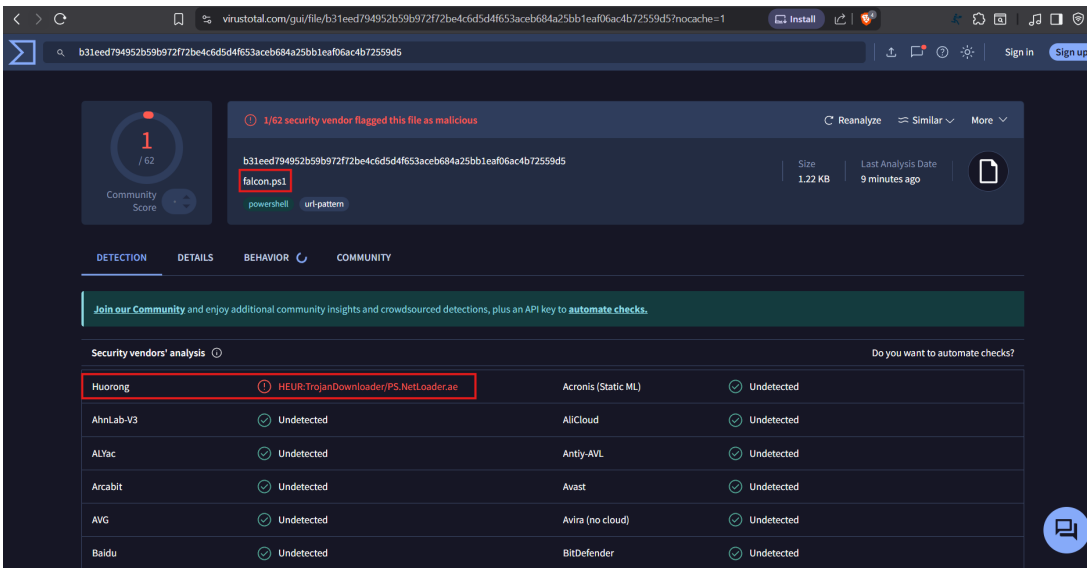


Figure 10: VirusTotal analysis of the attack simulation PowerShell script

The analysis indicates that one security engine classified the script as a potential PowerShell-based downloader, reflecting behaviour commonly associated with malicious activity. Although the script was created for laboratory simulation, this classification demonstrates that the generated behaviour resembles techniques used in real-world attacks, validating the effectiveness of the attack simulation.

9.8 Incident Response Timeline

The following sequence summarises the detection and response workflow observed during the experiment.

- Attack simulation script executed on the monitored Windows endpoint.
- Sysmon generated telemetry events including encoded PowerShell execution and reconnaissance activity.
- Wazuh agent forwarded the events to the Wazuh manager.
- Correlation rule `990003` was triggered indicating a multi-stage attack.

Yatish Waghmare

- The Wazuh Active Response framework executed the containment script on the endpoint.
- Telegram notification was sent to the Security Operations Center.
- Network adapter was disabled and attacker command processes were terminated.

10 Conclusion

This project implemented a behaviour-based endpoint detection and response laboratory using the Wazuh security platform to simulate core capabilities of enterprise MDR solutions such as **CrowdStrike Falcon Complete**. The environment consisted of a Windows Server endpoint instrumented with Sysmon and the Wazuh agent, with the Wazuh manager acting as the central detection and response engine.

Custom correlation rules were developed to detect suspicious behaviours including encoded PowerShell execution, reconnaissance commands, and multi-stage attack activity. A simulated attack script generated adversary-like techniques that produced telemetry captured by Sysmon and analysed by Wazuh.

When the correlation conditions were met, rule `990003` triggered automated containment through the Wazuh Active Response mechanism. The response disabled the external network adapter, terminated malicious processes, and sent a real-time SOC notification via Telegram.

The results demonstrate that behaviour-based detection combined with automated response can effectively identify and contain suspicious endpoint activity, replicating key operational principles of modern enterprise EDR platforms such as **CrowdStrike Falcon Complete**.

11 Future Improvements

Although the lab successfully demonstrated behaviour-based detection and automated containment, several enhancements could further improve the realism and capability of the environment.

First, additional telemetry sources such as network traffic monitoring and file integrity monitoring could be integrated to provide deeper visibility into attacker behaviour. This would allow the detection engine to correlate host-based and network-based indicators for more comprehensive threat detection.

Yatish Waghmare

Second, the detection logic could be expanded by implementing additional MITRE ATT&CK technique mappings. This would allow the lab environment to simulate a wider range of adversary techniques and improve the accuracy of behavioural correlation rules.

Third, automated response mechanisms could be extended to support additional containment strategies such as process quarantine, file deletion, registry rollback, or temporary user account isolation. These improvements would more closely replicate the response capabilities of enterprise EDR platforms.

Finally, integrating a Security Information and Event Management (SIEM) platform or threat intelligence feeds could enhance the detection pipeline by enabling centralized monitoring, threat enrichment, and advanced analytics.

These improvements would further strengthen the lab environment and provide a more comprehensive simulation of modern endpoint detection and response operations.

Yatish Waghmare

References

[1] Wazuh Inc., "Wazuh Documentation," 2024. [Online]. Available: https://documentation.wazuh.com/

[2] Microsoft, "Sysmon - System Monitor," Microsoft Learn, 2024. [Online]. Available: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon

[3] MITRE Corporation, "MITRE ATT&CK Framework," 2024. [Online]. Available: https://attack.mitre.org/

[4] CrowdStrike, "CrowdStrike Falcon Complete MDR," 2024. [Online]. Available: https://www.crowdstrike.com/en-us/resources/

[5] VirusTotal, "VirusTotal Malware Analysis Platform," 2024. [Online]. Available: https://www.virustotal.com/

[6] Telegram, "Telegram Bot API Documentation," 2024. [Online]. Available: https://core.telegram.org/bots/api