SOAR Workflow Description

1. Playbook: Login (Failed RDP / Suspicious Authentication)

Workflow Overview

This playbook automates the triage of suspicious login attempts (failed RDP logons or brute force attempts) detected by Wazuh.

Steps

1) Trigger – Playbook is initiated when Wazuh sends an alert through the webhook integration.
2) Parse Alert – The alert payload is parsed to extract rule ID, username, source IP, and host.
3) Rule Filtering – Only proceed if the alert corresponds to rule IDs 100010, 100012, or 100013 (custom login-related rules).
4) Reputation Check – The source IP is submitted to AbuseIPDB for reputation scoring.
    - If IP is malicious (high abuse score):
        - Escalate to Discord with detailed alert context (host, user, source IP, reputation result).
        - Mark the case as Escalated and Document.
    - If IP is not malicious:
        - Send a notification to Discord (informational/monitoring).
        - Mark the case as Closed and Document.

Outcome

Ensures all suspicious login attempts are checked against reputation data.

Reduces false positives by closing low-risk cases automatically.

Provides SOC analysts with actionable Discord alerts for confirmed malicious sources.

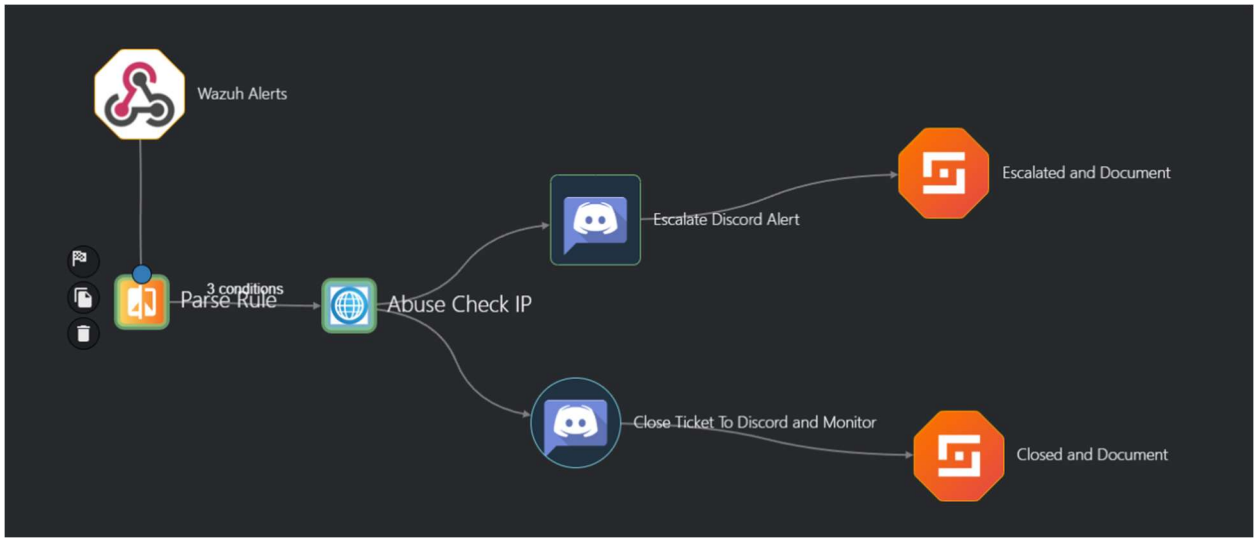

Figure 1: Playbook for Failed Login

2. Playbook: GPO Modified

Workflow Overview

This playbook investigates Group Policy Object (GPO) modification events to determine whether the change was authorized.

Steps

1) Trigger – Playbook is initiated when Wazuh sends an alert through the webhook.
2) Parse Alert – The alert payload is parsed and filtered to only proceed if the rule ID is 100014 (GPO Modified).
3) Active Directory Lookup – The username from the alert is checked in Active Directory to verify if the actor belongs to the authorized administrators group.
4) Decision Logic:
    a. If User is a valid admin
        i. Send a closure notification to Discord with details of the change
        ii. Mark the case as closed and document
    b. If user is not an authorized admin:
        i. Send an escalation alert to Discord with details of the unauthorized change
        ii. Mark the case as escalated and document

Outcome

- Validates whether GPO changes are legitimate or malicious.
- Provides immediate escalation for unauthorized modifications.
- Ensures all events are logged and documented for audit purposes.
- Doesn't catch if the threat attacker created a fake admin account since it will come legitimate using this playbook

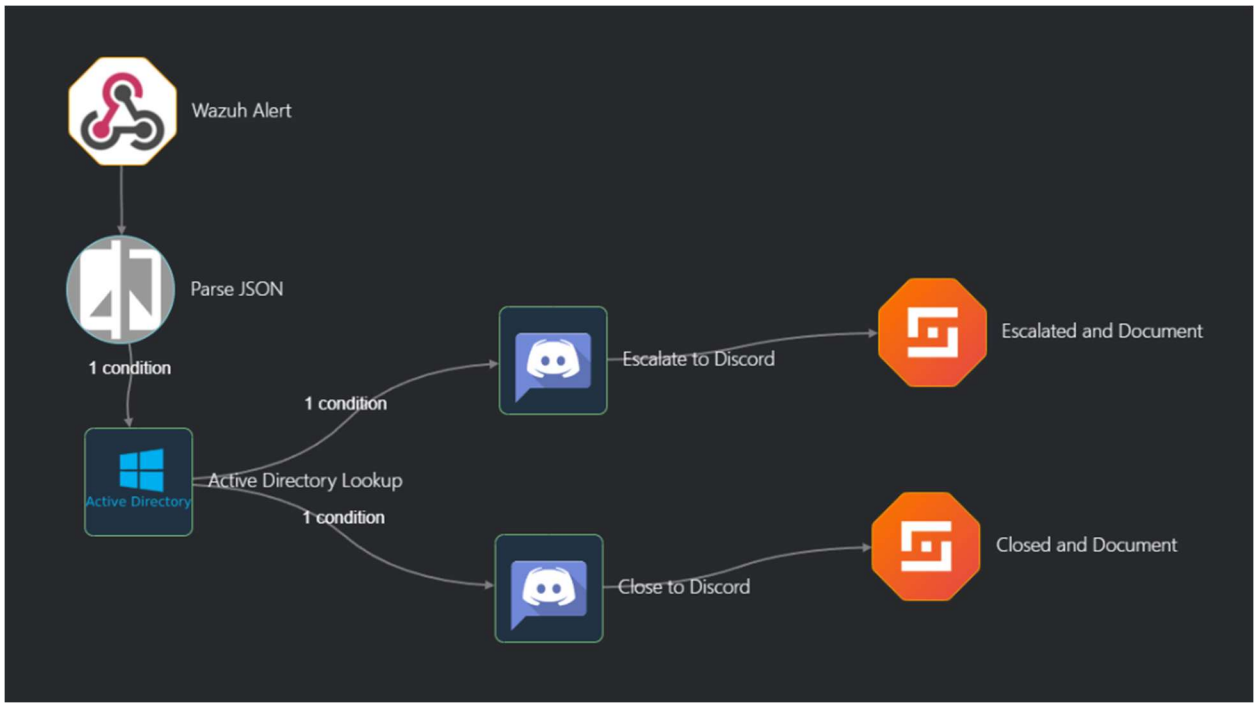

Figure 2: Playbook for GPO Modification

3. Playbook: Suspicious PowerShell Playbook

Workflow Overview

This playbook analyzes suspicious PowerShell execution events, with logic for decoding encoded commands, extracting IOCs, and checking IP reputation.

Steps

1) Trigger – Playbook is initiated when Wazuh sends an alert through the webhook.
2) Parse Alert – The alert payload is parsed and filtered to match the relevant Suspicious PowerShell detection rule.

3) Rule Validation – If the alert is an allowed rule, continue. Otherwise ignore.
4) Command Line Check
    a. If the alert does not contain a command line (cmd is missing or empty):
        i. Send a closure notification to Discord.
        ii. Mark the case as Closed and Documented.
    b. If a command line is present, continue.
5) Encoded Command Check
    a. If the PowerShell command contains -enc (encoded command):
        i. Decode the Base64 content.
        ii. Continue to IOC extraction.
    b. If no encoded command, skip decoding and proceed directly to IOC extraction.
6) IOC Extraction – Extract possible indicators (URLs, IPs, SHA256 hashes) from the decoded or raw command line using regex parsing.
7) IP Validation
    a. If no IPs are extracted:
        i. Escalate to Discord with available details.
        ii. Mark the case as Escalated and Documented.
    b. If IPs exist, continue to reputation check.
8) AbuseIPDB Lookup – Submit extracted IPs to AbuseIPDB.
    a. If IP is malicious: Escalate to Discord with IOC details, then mark as Escalated and Documented.
    b. If IP is not malicious: Send closure notification to Discord and mark as Closed and Documented.

Outcome

- Detects and validates suspicious PowerShell activity.
- Decodes obfuscated commands for deeper analysis.
- Extracts and enriches IOCs (URLs, IPs, hashes).
- Escalates malicious IPs while auto-closing low-risk cases.

Figure 3: Playbook for Suspicious PowerShell

4. Playbook: LSASS Access

Workflow Overview

This playbook investigates process access attempts to LSASS.exe, a common target for credential dumping, by validating the event and enriching with VirusTotal.

Steps

1) Trigger – Playbook is initiated when Wazuh sends an alert through the webhook.
2) Parse Alert – The alert payload is parsed and filtered to match the LSASS Access detection rule.
3) Rule Validation – If the alert is not from the correct rule, ignore. If valid, continue.
4) Target Process Check
    a. If the target process is lsass.exe, continue.
    b. If the target process is not LSASS, send a closure notification to Discord and mark as Closed and Documented.
5) Source Image Check

a. If the alert contains a Source Image field (process attempting LSASS access), continue.
b. If no Source Image is present, send a closure notification to Discord and mark as Closed and Documented.

6) Hash Extraction – Parse the SHA256 hash of the source process image.
7) VirusTotal Lookup – Submit the process hash to VirusTotal for reputation analysis.
    a. If malicious: Escalate to Discord with details and mark as Escalated and Documented.
    b. If clean: Send closure notification to Discord and mark as Closed and Documented.

Outcome

- Ensures only LSASS-related events are analyzed.
- Enriches suspicious processes via VirusTotal hash lookups.
- Escalates confirmed malicious process access while auto-closing benign cases.

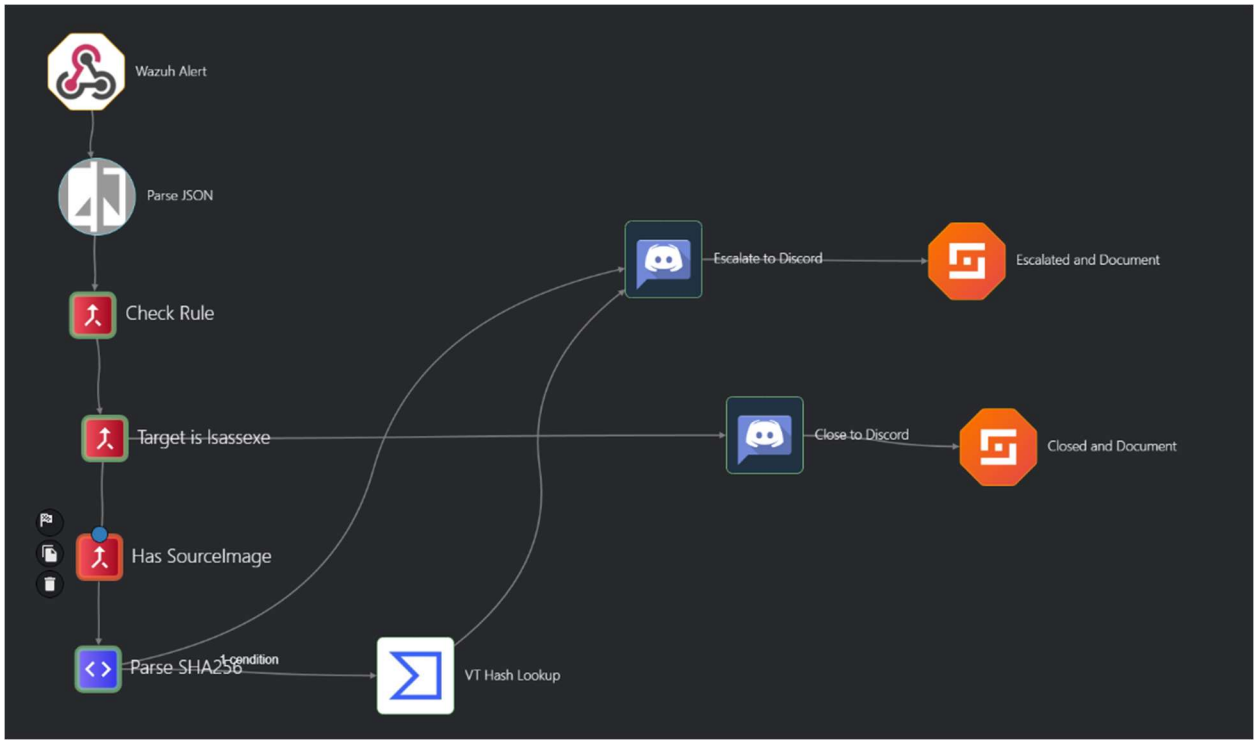

Figure 4: Playbook for LSASS Access

5. Playbook: VSS Deletion

Workflow Overview

This playbook detects and responds to potential shadow copy deletion attempts, which are commonly used in ransomware and anti-forensics activities.

Steps

1) Trigger – Playbook is initiated when Wazuh sends an alert through the webhook.
2) Parse Alert – Extract relevant fields such as process image, command line, and rule ID.
3) Rule Filtering – Only proceed if the alert corresponds to the VSS Deletion detection rule.
4) Command Line Check
    a. If no command line (cmd) is present:
        i. Send a closure notification to Discord.
        ii. Mark the case as Closed and Documented.
    b. If a command line exists, continue.
5) VSS Deletion Logic
    a. Match process and command line against known deletion patterns:
        i. vssadmin with both delete and shadow present in the command.
        ii. wmic with shadowcopy delete in the command.
6) Decision Logic
    a. If VSS deletion pattern matches:
        i. Escalate to Discord with full details (host, process, command line).
        ii. Mark as Escalated and Documented.
    b. If no pattern match:
        i. Send closure notification to Discord.
        ii. Mark as Closed and Documented.

Outcome

- Detects and escalates attempts to delete shadow copies.
- Differentiates between real VSS deletion commands and irrelevant process activity.
- Ensures incomplete alerts (no command line) are safely closed.

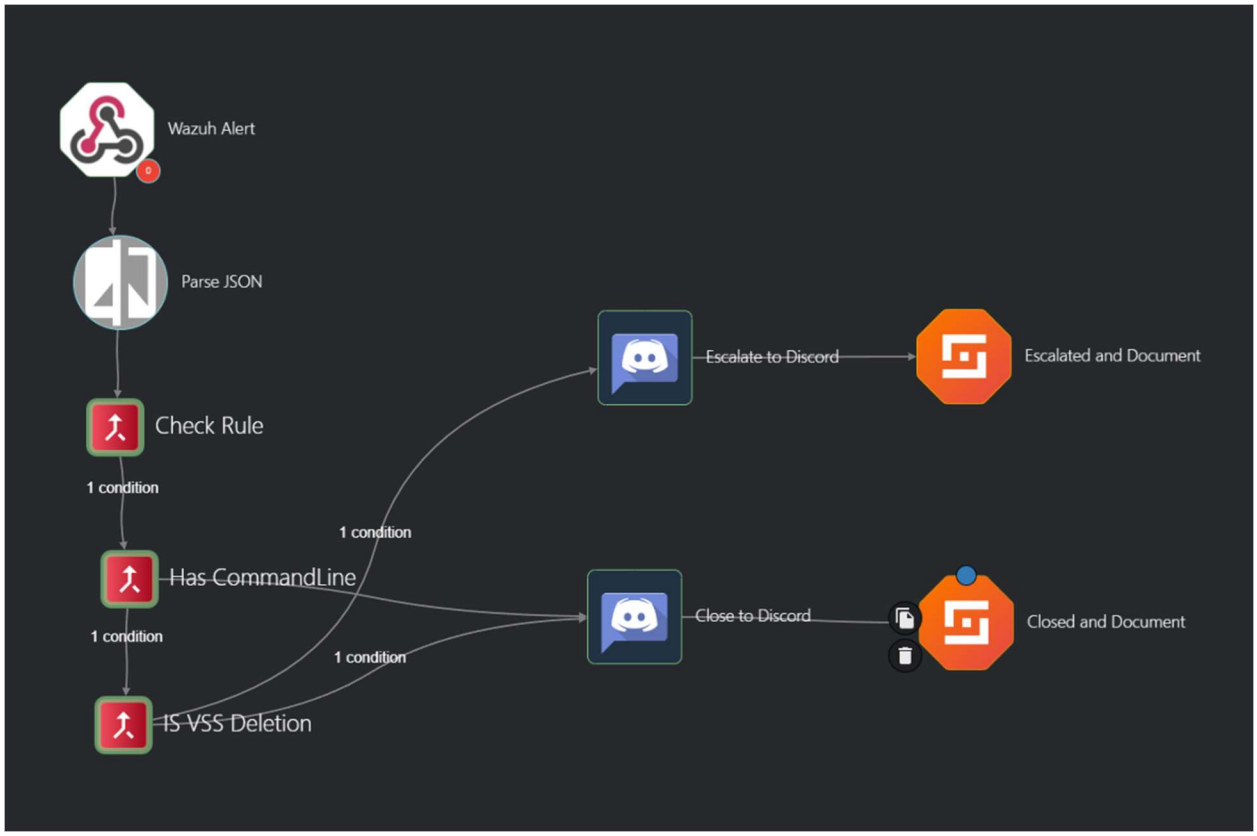

Figure 5: Playbook for VSS Deletion