Wazuh SIEM Deployment Plan

CONTENTS

1. Executive Summary
2. Current State
3. Previous SIEM Work
4. Infrastructure Overview
5. Machines to Monitor
6. Services to Monitor
7. Phase 1: Wazuh Manager on Darth (Day 1-2)
8. 1.1 Create SIEM VM on Darth
9. 1.2 Install Wazuh Manager
10. 1.3 Install Wazuh Dashboard
11. 1.4 Configure Manager
12. 1.5 Firewall Rules (darth)
13. Phase 2: Agent Deployment (Day 2-3)
14. 2.1 Agent Installation Script
15. 2.2 Deploy to miner (Production)
16. 2.3 Deploy to trooper2
17. 2.4 Verify Agent Connections
18. Phase 3: Security Rules & Detection (Day 3-5)
19. 3.1 SSH Monitoring
20. 3.2 Docker Monitoring
21. 3.3 Web Server Monitoring
22. 3.4 Database Monitoring
23. 3.5 Kubernetes Monitoring
24. 3.6 Auth Service Monitoring
25. Phase 4: Log Collection & Integration (Day 5-7)
26. 4.1 Logs to Collect
27. 4.2 Log Collection Config
28. 4.3 Docker Log Driver
29. 4.4 Kubernetes Audit Logs
30. Phase 5: Alerting & Notifications (Day 7-10)

31. 5.1 Email Alerts
32. 5.2 Slack Integration (Optional)
33. 5.3 Telegram Integration
34. Phase 6: Compliance & Auditing (Day 10-14)
35. 6.1 Compliance Frameworks
36. 6.2 File Integrity Monitoring (FIM)
37. 6.3 Security Configuration Assessment (SCA)
38. 6.4 Vulnerability Detection
39. 6.5 Active Response
40. Phase 7: Dashboard & Reporting (Day 14-21)
41. 7.1 Dashboard Setup
42. 7.2 Key Dashboards to Create
43. 7.3 Scheduled Reports
44. Timeline
45. Prerequisites
46. On darth (SIEM host)
47. On miner (production)
48. On trooper2 (AI VMs host)
49. Post-Deployment Checklist
50. Maintenance
51. Daily
52. Weekly
53. Monthly
54. References

Wazuh SIEM Deployment Plan

Created: 2026-03-27 **Author:** OpenClaw **Status:** Planning

---

Executive Summary

Deploy Wazuh SIEM on darth (the SIEM server) to provide centralized security monitoring, threat detection, and compliance auditing for the entire STSGym infrastructure.

Current State

Host	Role	Status	Monitoring
darth (the SIEM server)	K8s dev, 124GB RAM	Active	**New SIEM host**
trooper2 (the internal network)	AI VMs host	Active	Agent needed
miner (the production server)	Production VPS	Active	Agent needed

Previous SIEM Work

Documentation exists for a previous SIEM deployment plan: - `docs/infrastructure/vm-infrastructure.md` - Wazuh agent deployment - `docs/infrastructure/cicerone-infrastructure-deployment.md` - SIEM server setup - Planned: `report` VM (192.168.1.98) as Wazuh Manager - **Current Status:** No VMs running, no Wazuh installed

---

Infrastructure Overview

Machines to Monitor

Hostname	IP	Role	Services	Priority
darth	the SIEM server	K8s dev, SIEM host	Kind, Docker, Kubernetes	Critical
trooper2	the internal network	AI VMs host	libvirt, VMs, cicerone	High
miner	the production server	Production VPS	Docker, all stsgym services	Critical

Services to Monitor

Service	Type	Alert On
Docker containers	Infrastructure	Container death, restart loops
Nginx	Web server	5xx errors, rate limiting
PostgreSQL	Database	Connection failures, slow queries
Redis	Cache	Memory exhaustion, connection failures
Auth service	Security	Failed logins, account lockouts
fail2ban	Security	Ban events, SSH attempts
Kubernetes	Orchestration	Pod failures, node issues

Phase 1: Wazuh Manager on Darth (Day 1-2)

1.1 Create SIEM VM on Darth

VM Specifications:

Property	Value
Name	`siem-manager`
vCPU	4
RAM	16GB
Disk	100GB
Network	Bridge (192.168.1.0/24) or NAT
OS	Ubuntu 24.04 LTS

Alternative: Run Wazuh directly on darth host

Given darth has 124GB RAM and no VMs, we can run Wazuh directly on the host instead of in a VM. This simplifies deployment and reduces overhead.

Recommendation: Install Wazuh Manager directly on darth host.

1.2 Install Wazuh Manager

```
#!/bin/bash
# Script: install-wazuh-manager.sh
# Run on darth (the SIEM server)

set -e

echo "=== Installing Wazuh Manager ==="

# Add Wazuh repository
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | 
dearmor -o /usr/share/keyrings/wazuh.gpg
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] htt
packages.wazuh.com/4.x/apt/ stable main" | tee /etc/apt/
sources.list.d/wazuh.list

# Update and install
apt-get update
apt-get install -y wazuh-manager

# Enable and start
```

```
        systemctl enable --now wazuh-manager

        # Verify
        systemctl status wazuh-manager

        echo "=== Wazuh Manager installed ==="
```

1.3 Install Wazuh Dashboard

```
        #!/bin/bash
        # Script: install-wazuh-dashboard.sh

        set -e

        echo "=== Installing Wazuh Dashboard ==="

        # Install OpenSearch
        curl -s https://artifacts.opensearch.org/releases/bundl
opensearch/2.x/opensearch-2.x.repo | tee /etc/yum.repos.d/
opensearch-2.x.repo
        # For Ubuntu/Debian:
        wget -q https://artifacts.opensearch.org/releases/bundl
opensearch/2.x/opensearch-2.x.deb
        dpkg -i opensearch-2.x.deb

        # Install Wazuh Dashboard
        apt-get install -y wazuh-dashboard

        # Configure
        # Edit /etc/wazuh-dashboard/opensearch_dashboards.yml

        # Enable and start
        systemctl enable --now wazuh-dashboard

        echo "=== Wazuh Dashboard installed ==="
        echo "Access: https://wazuh.stsgym.com:5601"
        echo "Default: admin / admin"
```

1.4 Configure Manager

```
<!-- /var/ossec/etc/ossec.conf additions -->

<ossec_config>
  <!-- Network listening -->
  <remote>
    <connection>secure</connection>
    <port>1514</port>
    <protocol>tcp</protocol>
    <allowed-ips><internal-network-cidr></allowed-ips>
    <allowed-ips>192.168.0.0/16</allowed-ips>
    <allowed-ips><vpn-cidr></allowed-ips> <!-- Tailscal
  </remote>

  <!-- Registration service -->
  <auth>
    <disabled>no</disabled>
    <port>1515</port>
    <use_source_ip>no</use_source_ip>
    <force_insert>yes</force_insert>
    <force_time>0</force_time>
    <purge>yes</purge>
    <use_password>no</use_password>
    <limit_maxagents>yes</limit_maxagents>
    <ssl_verify_host>no</ssl_verify_host>
  </auth>

  <!-- Global settings -->
  <global>
    <jsonout_output>yes</jsonout_output>
    <logall>yes</logall>
    <logall_json>yes</logall_json>
  </global>

  <!-- Alerts -->
  <alerts>
    <log_alert_level>3</log_alert_level>
    <email_alert_level>12</email_alert_level>
```

```
  </alerts>
</ossec_config>
```

1.5 Firewall Rules (darth)

```
#!/bin/bash
# Script: configure-siem-firewall.sh

# Wazuh ports
ufw allow 1514/tcp comment "Wazuh agent connection"
ufw allow 1515/tcp comment "Wazuh agent registration"
ufw allow 55000/tcp comment "Wazuh API"
ufw allow 5601/tcp comment "Wazuh Dashboard"

# Reload firewall
ufw reload

echo "=== Firewall configured for Wazuh ==="
```

Phase 2: Agent Deployment (Day 2-3)

2.1 Agent Installation Script

```
#!/bin/bash
# Script: install-wazuh-agent.sh
# Run on each host to be monitored

MANAGER_IP="${MANAGER_IP}"  # darth

set -e

echo "=== Installing Wazuh Agent ==="

# Add Wazuh repository
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | 
dearmor -o /usr/share/keyrings/wazuh.gpg
echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] htt
packages.wazuh.com/4.x/apt/ stable main" | tee /etc/apt/
```

```
sources.list.d/wazuh.list

        # Update and install
        apt-get update
        apt-get install -y wazuh-agent

        # Configure manager IP
        sed -i "s|<address>MANAGER_IP</address>|<address>$
{MANAGER_IP}</address>|g" /var/ossec/etc/ossec.conf

        # Enable and start
        systemctl enable --now wazuh-agent

        # Verify connection
        systemctl status wazuh-agent

        echo "=== Wazuh Agent installed and connected ==="
```

2.2 Deploy to miner (Production)

```
        # SSH to miner
        ssh wez@${PRODUCTION_HOST}

        # Install agent
        curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | 
dearmor -o /usr/share/keyrings/wazuh.gpg
        echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] htt
packages.wazuh.com/4.x/apt/ stable main" | sudo tee /etc/apt/
sources.list.d/wazuh.list
        sudo apt-get update
        sudo apt-get install -y wazuh-agent

        # Configure for darth SIEM (via Tailscale)
        sudo sed -i 's|<address>MANAGER_IP</address>|<address>$
{MANAGER_IP}</address>|g' /var/ossec/etc/ossec.conf

        # Enable and start
        sudo systemctl enable --now wazuh-agent
```

2.3 Deploy to trooper2

```
# SSH to trooper2 (localhost on darth, or via network)
# trooper2 is on the internal network

# Install agent (same steps as above)
# Configure manager IP as the SIEM server (darth)
```

2.4 Verify Agent Connections

```
# On darth (SIEM manager)
/var/ossec/bin/agent_control -l

# Expected output:
#   ID: 001, Name: miner, IP: the production server, Ac
#   ID: 002, Name: trooper2, IP: the internal network, 
#   ID: 003, Name: darth, IP: the SIEM server, Active
```

Phase 3: Security Rules & Detection (Day 3-5)

3.1 SSH Monitoring

```
<!-- /var/ossec/etc/rules/local_rules.xml -->

<group name="ssh,authentication">
  <!-- Multiple SSH failures -->
  <rule id="100001" level="5">
    <if_sid>5716</if_sid>
    <match>Failed password</match>
    <description>SSH authentication failure</descriptio
    <group>authentication_failed</group>
  </rule>

  <!-- Brute force detection (5 failures in 1 minute) -
  <rule id="100002" level="10" frequency="5" timeframe=
    <if_matched_sid>100001</if_matched_sid>
    <description>SSH brute force attack detected</
```

```
description>
            <group>authentication_attack</group>
          </rule>

          <!-- Successful root login (alert) -->
          <rule id="100003" level="12">
            <if_sid>5700</if_sid>
            <match>Accepted password for root</match>
            <description>Root SSH login - investigate</descript
            <group>root_login</group>
          </rule>

          <!-- Successful sudo -->
          <rule id="100004" level="3">
            <match>sudo:</match>
            <regex>session opened for user</regex>
            <description>Sudo session opened</description>
            <group>sudo</group>
          </rule>
        </group>
```

3.2 Docker Monitoring

```
        <!-- Docker container monitoring -->

        <group name="docker">
          <!-- Container started -->
          <rule id="100100" level="3">
            <field name="program">dockerd</field>
            <match>Container started</match>
            <description>Docker container started</description>
          </rule>

          <!-- Container died -->
          <rule id="100101" level="7">
            <field name="program">dockerd</field>
            <match>Container died</match>
            <description>Docker container died unexpectedly</
description>
```

```
    <group>container_failure</group>
  </rule>

  <!-- Container restart loop -->
  <rule id="100102" level="12" frequency="3" timeframe=
    <if_matched_sid>100101</if_matched_sid>
    <description>Docker container in restart loop</
description>
    <group>container_failure</group>
  </rule>
</group>
```

3.3 Web Server Monitoring

```
<!-- Nginx/HTTP monitoring -->

<group name="web">
  <!-- 5xx errors -->
  <rule id="100200" level="5">
    <match>" 5\d{2} "</match>
    <description>HTTP 5xx server error</description>
    <group>web_error</group>
  </rule>

  <!-- Multiple 401/403 from same IP (scan) -->
  <rule id="100201" level="10" frequency="10" timeframe
    <if_matched_sid>100200</if_matched_sid>
    <same_source_ip />
    <description>Multiple HTTP errors from same IP (pos
scan)</description>
    <group>web_scan</group>
  </rule>

  <!-- SQL injection attempt -->
  <rule id="100202" level="12">
    <match>UNION SELECT|OR 1=1|AND 1=1|--'|;</match>
    <description>SQL injection attempt detected</descri
    <group>web_attack,sql_injection</group>
  </rule>
```

```
    <!-- Path traversal -->
    <rule id="100203" level="12">
      <match>\.\./|\.\.%2f|%2e%2e</match>
      <description>Path traversal attempt detected</
description>
      <group>web_attack,path_traversal</group>
    </rule>
  </group>
```

3.4 Database Monitoring

```
  <!-- PostgreSQL monitoring -->

  <group name="database">
    <!-- Connection failure -->
    <rule id="100300" level="7">
      <field name="program">postgres</field>
      <match>FATAL: connection</match>
      <description>PostgreSQL connection failure</descrip
    </rule>

    <!-- Authentication failure -->
    <rule id="100301" level="5">
      <field name="program">postgres</field>
      <match>FATAL: password authentication failed</match
      <description>PostgreSQL authentication failure</
description>
      <group>database_auth</group>
    </rule>
  </group>
```

3.5 Kubernetes Monitoring

```
  <!-- Kubernetes monitoring -->

  <group name="kubernetes">
    <!-- Pod failed -->
```

```
    <rule id="100400" level="7">
      <match>Pod ".*" failed</match>
      <description>Kubernetes pod failed</description>
      <group>k8s_pod_failure</group>
    </rule>

    <!-- Node not ready -->
    <rule id="100401" level="10">
      <match>Node .* status is now: NotReady</match>
      <description>Kubernetes node not ready</description>
      <group>k8s_node_failure</group>
    </rule>

    <!-- OOM killed -->
    <rule id="100402" level="7">
      <match>OOMKilled</match>
      <description>Container OOM killed</description>
      <group>k8s_oom</group>
    </rule>
  </group>
```

3.6 Auth Service Monitoring

```
  <!-- Auth service (stsgym.com) monitoring -->

  <group name="auth">
    <!-- Failed login -->
    <rule id="100500" level="5">
      <match>login_failed</match>
      <description>Auth service login failure</description>
      <group>auth_failure</group>
    </rule>

    <!-- Account locked -->
    <rule id="100501" level="10">
      <match>account_locked</match>
      <description>Account locked due to failed attempts</
description>
      <group>auth_lockout</group>
```

```
  </rule>

  <!-- Admin action -->
  <rule id="100502" level="8">
    <match>admin_action</match>
    <description>Admin action performed</description>
    <group>admin_audit</group>
  </rule>
</group>
```

Phase 4: Log Collection & Integration (Day 5-7)

4.1 Logs to Collect

Log	Path	Purpose
Syslog	`/var/log/syslog`	System events
Auth	`/var/log/auth.log`	SSH, sudo events
Docker	`/var/log/docker.log`	Container events
Nginx	`/var/log/nginx/*.log`	Web access/errors
PostgreSQL	`/var/log/postgresql/*.log`	Database events
Kubernetes	`/var/log/pods/`	Pod logs
Auth service	`/var/log/auth-service/`	Login events
fail2ban	`/var/log/fail2ban.log`	Ban events

4.2 Log Collection Config

```
<!-- /var/ossec/etc/ossec.conf - Log collection -->

<ossec_config>
  <!-- System logs -->
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/syslog</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
```

```
  </localfile>

  <!-- Docker -->
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/docker.log</location>
  </localfile>

  <!-- Nginx -->
  <localfile>
    <log_format>nginx_access</log_format>
    <location>/var/log/nginx/access.log</location>
  </localfile>

  <localfile>
    <log_format>nginx_error</log_format>
    <location>/var/log/nginx/error.log</location>
  </localfile>

  <!-- fail2ban -->
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/fail2ban.log</location>
  </localfile>

  <!-- Kubernetes (if kind) -->
  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/kubernetes/*.log</location>
  </localfile>
</ossec_config>
```

4.3 Docker Log Driver

Configure Docker to send logs to syslog (forwarded to Wazuh):

```
# /etc/docker/daemon.json
{
  "log-driver": "syslog",
  "log-opts": {
```

```
    "syslog-address": "unix:///var/run/syslog.sock",
    "tag": "{{.Name}}/{{.ID}}"
  }
}

# Restart Docker
systemctl restart docker
```

4.4 Kubernetes Audit Logs

```
# kind-config.yaml - Enable audit logs
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: ClusterConfiguration
    apiServer:
      extraArgs:
        audit-log-path: /var/log/kubernetes/audit.log
        audit-log-maxage: "30"
        audit-log-maxbackup: "10"
        audit-log-maxsize: "100"
```

Phase 5: Alerting & Notifications (Day 7-10)

5.1 Email Alerts

```
<!-- /var/ossec/etc/ossec.conf -->

<ossec_config>
  <global>
    <email_notification>yes</email_notification>
    <smtp_server>mail.stsgym.com</smtp_server>
    <email_from>wazuh@stsgym.com</email_from>
    <email_to>wlrobbi@gmail.com</email_to>
  </global>
```

```
    <alerts>
      <email_alert_level>12</email_alert_level>
    </alerts>
  </ossec_config>
```

5.2 Slack Integration (Optional)

```
  # Install Wazuh Slack integration
  cd /var/ossec/integrations
  curl -s https://raw.githubusercontent.com/wazuh/wazuh/m
integrations/slack.py > slack

  # Configure
  chmod +x slack

  # Add to ossec.conf
  <integration>
    <name>slack</name>
    <hook_url>https://hooks.slack.com/services/YOUR/WEBHO
URL</hook_url>
    <alert_format>json</alert_format>
  </integration>
```

5.3 Telegram Integration

```
  <!-- Custom Telegram integration -->
  <integration>
    <name>custom-telegram</name>
    <hook_url>https://api.telegram.org/bot<BOT_TOKEN>/
sendMessage</hook_url>
    <alert_format>json</alert_format>
  </integration>
```

Phase 6: Compliance & Auditing (Day 10-14)

6.1 Compliance Frameworks

Framework	Requirements	Wazuh Modules
NIST 800-53	Audit logs, access control, incident response	File integrity, configuration assessment
CIS Benchmarks	System hardening	SCA (Security Configuration Assessment)
GDPR	Data protection, breach notification	Log monitoring, encryption checks
PCI-DSS	Access monitoring, vulnerability scanning	Vulnerability detection, log analysis

6.2 File Integrity Monitoring (FIM)

```
        <!-- /var/ossec/etc/ossec.conf -->

        <syscheck>
          <!-- Critical system files -->
          <directories check_all="yes">/etc/passwd,/etc/shadow,/
sudoers,/etc/ssh/sshd_config</directories>

          <!-- Web configurations -->
          <directories check_all="yes">/etc/nginx/sites-enabled/
directories>

          <!-- Docker configs -->
          <directories check_all="yes">/etc/docker/daemon.json</
directories>

          <!-- Application configs -->
          <directories check_all="yes">/home/wez/auth-service/</
directories>

          <!-- Ignore frequent changes -->
          <ignore>/var/log</ignore>
          <ignore>/proc</ignore>
          <ignore>/sys</ignore>
```

```
  <!-- Frequency -->
  <frequency>21600</frequency> <!-- Every 6 hours -->
</syscheck>
```

6.3 Security Configuration Assessment (SCA)

```
# Wazuh includes SCA policies for:
# - CIS Ubuntu Linux Benchmark
# - NIST 800-53
# - PCI-DSS
# - GDPR

# View loaded policies
/var/ossec/bin/wazuh-control status

# SCA results are visible in Dashboard under:
# Security Events > Integrity Monitoring
```

6.4 Vulnerability Detection

```
<!-- /var/ossec/etc/ossec.conf -->

<vulnerability-detector>
  <enabled>yes</enabled>
  <interval>5m</interval>
  <ignore_time>6h</ignore_time>
  <run_on_start>yes</run_on_start>

  <!-- Ubuntu CVE database -->
  <provider name="canonical">
    <enabled>yes</enabled>
    <os>trusty</os>
    <os>xenial</os>
    <os>bionic</os>
    <os>focal</os>
    <os>jammy</os>
    <update_interval>1h</update_interval>
```

```
    </provider>
  </vulnerability-detector>
```

6.5 Active Response

```
  <!-- /var/ossec/etc/ossec.conf -->

  <active-response>
    <!-- Block IP on brute force -->
    <command name="firewall-drop">
      <executable>firewall-drop.sh</executable>
      <timeout_allowed>yes</timeout_allowed>
    </command>

    <!-- Trigger on SSH brute force -->
    <active-response>
      <command>firewall-drop</command>
      <location>local</location>
      <rules_id>100002</rules_id>
      <timeout>3600</timeout>
    </active-response>

    <!-- Trigger on web scan -->
    <active-response>
      <command>firewall-drop</command>
      <location>local</location>
      <rules_id>100201</rules_id>
      <timeout>1800</timeout>
    </active-response>
  </active-response>
```

Phase 7: Dashboard & Reporting (Day 14-21)

7.1 Dashboard Setup

Access Wazuh Dashboard at: `https://wazuh.stsgym.com:5601`

Default credentials: `admin` / `admin` (change immediately)

7.2 Key Dashboards to Create

1. **Security Overview**
   - Alert volume by severity
   - Top attacked hosts
   - Geographic attack map
   - Trend over time
2. **SSH Monitoring**
   - Failed logins by IP
   - Brute force attempts
   - Successful logins
   - Geographic distribution
3. **Web Security**
   - 4xx/5xx errors
   - SQL injection attempts
   - Path traversal
   - Rate limiting triggers
4. **Docker & Kubernetes**
   - Container starts/stops
   - Pod health
   - Resource usage anomalies
   - Restart loops
5. **Compliance**
   - CIS benchmark scores
   - FIM changes
   - Vulnerability counts
   - Policy violations

7.3 Scheduled Reports

```
        # Daily security summary
        # Sent at 06:00 UTC
        # Includes:
        # - Alert summary (24h)
        # - New vulnerabilities
        # - FIM changes
        # - Active response actions

        # Weekly compliance report
        # Sent Monday 06:00 UTC
        # Includes:
        # - CIS benchmark compliance
        # - Policy violations
        # - Remediation recommendations
```

Timeline

Phase	Description	Duration	Start	End
1	Wazuh Manager on darth	2 days	Day 1	Day 2
2	Agent deployment	1 day	Day 2	Day 3
3	Security rules	2 days	Day 3	Day 5
4	Log collection	2 days	Day 5	Day 7
5	Alerting setup	3 days	Day 7	Day 10
6	Compliance & auditing	4 days	Day 10	Day 14
7	Dashboards & reporting	7 days	Day 14	Day 21

Total Duration: 21 days

Prerequisites

On darth (SIEM host)

- [ ] Ubuntu 24.04 LTS installed
- [ ] 4+ vCPUs available
- [ ] 16GB+ RAM available
- [ ] 100GB+ disk space
- [ ] Network accessible from all monitored hosts
- [ ] Ports 1514, 1515, 55000, 5601 open

On miner (production)

- [ ] SSH access to the production server
- [ ] Tailscale connection (the VPN address)
- [ ] Root/sudo access

On trooper2 (AI VMs host)

- [ ] SSH access to the internal network
- [ ] Root/sudo access

Post-Deployment Checklist

- [ ] Wazuh Manager running on darth
- [ ] Dashboard accessible at https://wazuh.stsgym.com:5601
- [ ] Agents connected from all hosts
- [ ] Security rules active (test with failed SSH)
- [ ] Log collection verified
- [ ] Email alerts working
- [ ] FIM configured for critical files
- [ ] SCA policies loaded
- [ ] Vulnerability detection enabled
- [ ] Active response tested
- [ ] Dashboards created
- [ ] Scheduled reports configured

Maintenance

Daily

- Check dashboard for critical alerts
- Review active response actions
- Verify log collection

Weekly

- Review FIM changes
- Update vulnerability database
- Check compliance scores
- Update rules if needed

Monthly

- Full vulnerability scan

- Review and tune alert thresholds
- Update Wazuh if new version available
- Audit active response rules

References

- Wazuh Documentation: https://documentation.wazuh.com/
- Wazuh Rules: https://github.com/wazuh/wazuh/tree/master/rules
- CIS Ubuntu Benchmark: https://www.cisecurity.org/benchmark/ubuntu_linux
- NIST 800-53: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

---

Last Updated: 2026-03-27