Mini SOC Project Report

Introduction

This report documents the implementation of a Mini Security Operations Center (SOC) platform using Wazuh, containerized deployments, and a CI/CD pipeline. The solution includes a functional Wazuh stack deployed via Docker Compose, with a prepared migration plan to Docker Swarm for scalability and high availability. A custom Wazuh rule was implemented to detect suspicious SSH login patterns, aligning with real-world security monitoring needs.

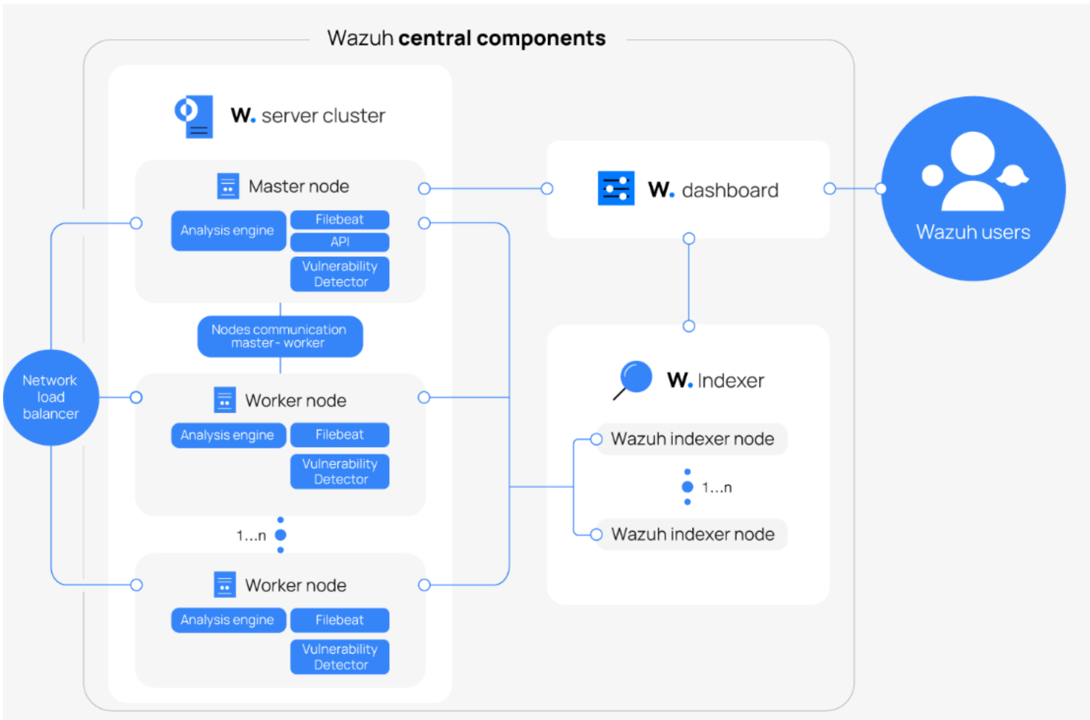
Repository: https://github.com/kaoutarlahmaidi/soc-platform

The complete source code, configurations, and documentation are available in the project repository.

Part One – Build a Mini SOC

1-Architecture Overview

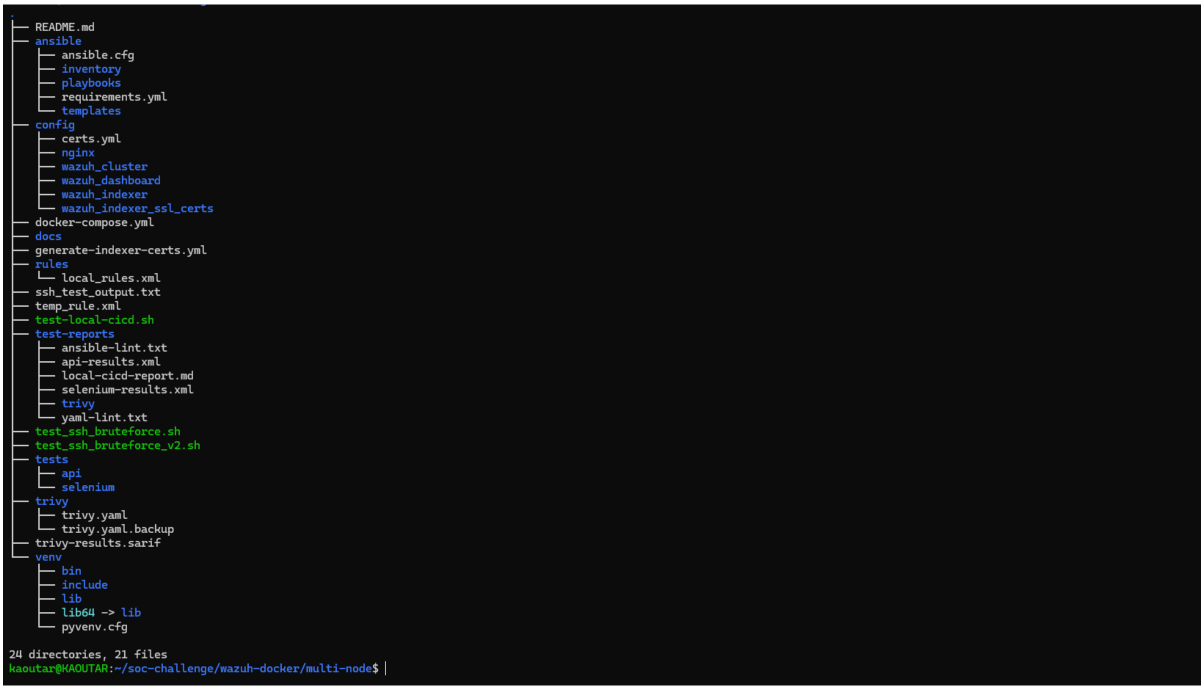
The architecture consists of a multi-node Wazuh cluster including manager, worker, indexers, and dashboard, fronted by an Nginx reverse proxy with TLS termination. Persistent storage is provided by Docker volumes. The design ensures high availability, secure communication, and scalability.



2-CI/CD Pipeline

2.1-Project Structure

The project is organized into well-defined directories to separate configuration, automation, testing, and documentation. The structure ensures clarity, modularity, and maintainability.



```
.
├── README.md
├── ansible
│   ├── ansible.cfg
│   ├── inventory
│   ├── playbooks
│   ├── requirements.yml
│   └── templates
├── config
│   ├── certs.yml
│   ├── nginx
│   ├── wazuh_cluster
│   ├── wazuh_dashboard
│   ├── wazuh_indexer
│   └── wazuh_indexer_ssl_certs
├── docker-compose.yml
├── docs
├── generate-indexer-certs.yml
├── rules
│   └── local_rules.xml
├── ssh_test_output.txt
├── temp_rule.xml
├── test-local-cicd.sh
├── test-reports
│   ├── ansible-lint.txt
│   ├── api-results.xml
│   ├── local-cicd-report.md
│   ├── selenium-results.xml
│   ├── trivy
│   └── yaml-lint.txt
├── test_ssh_bruteforce.sh
├── test_ssh_bruteforce_v2.sh
├── tests
│   ├── api
│   └── selenium
├── trivy
│   ├── trivy.yaml
│   └── trivy.yaml.backup
├── trivy-results.sarif
└── venv
    ├── bin
    ├── include
    ├── lib
    ├── lib64 -> lib
    └── pyvenv.cfg

24 directories, 21 files
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node$
```

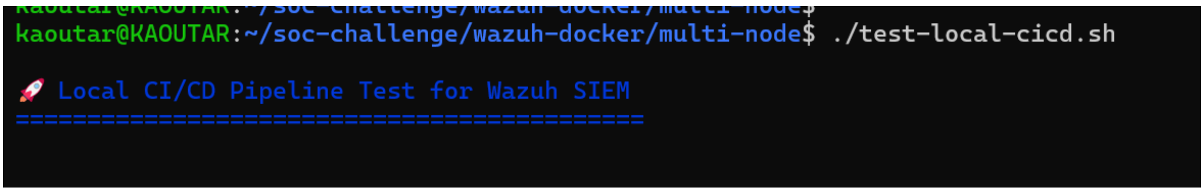
2.2-Local Pipeline Simulation

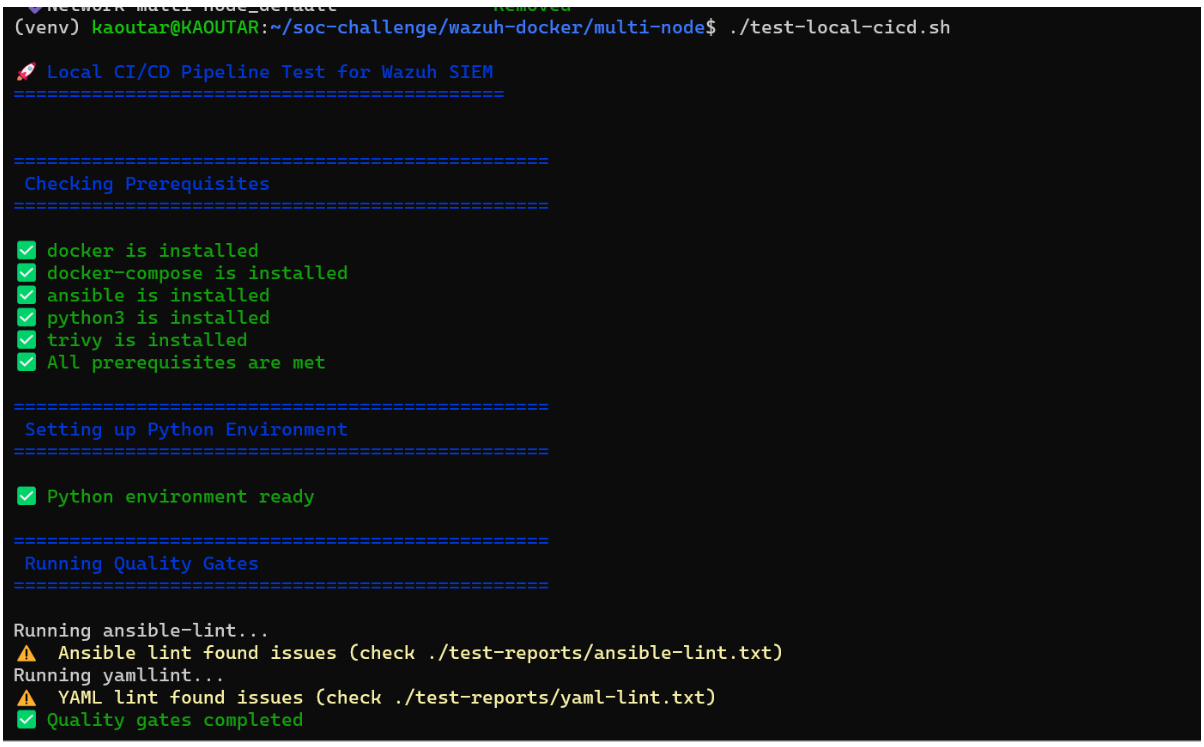
The provided `test-local-cicd.sh` script reproduces the pipeline stages locally. It validates prerequisites, builds images, runs security scans, executes automated tests, and simulates deployment using Ansible. The stages are:

- **Prerequisites Check:** Ensures Docker, Ansible, Trivy, and Python are installed and the Docker daemon is running.
- **Python Environment:** Creates a virtual environment and installs dependencies such as `ansible-lint`, `yamllint`, `pytest`, and `selenium`.
- **Quality Gates:** Runs Ansible linting and YAML validation, storing reports in `test-reports/`.
- **Build and Scan:** Builds/pulls Docker images and scans them with Trivy for HIGH/CRITICAL vulnerabilities.

- **Certificate Generation:** Uses `generate-indexer-certs.yml` to provision SSL certificates.
- **Test Environment:** Spins up the Wazuh stack via `docker-compose` and verifies dashboard availability.
- **Automated Tests:** Runs API health checks and Selenium UI tests against the Wazuh Dashboard.
- **Deployment Simulation:** Executes an Ansible dry-run (`--check`) with mock secrets to validate playbooks.
- **Report Generation:** Creates a markdown report (`test-reports/local-cicd-report.md`) summarizing results and artifacts.

This local script allows fast validation before committing to GitHub.





```
==============================================
 Building and Scanning Images
==============================================

Pulling wazuh/wazuh-manager:4.4.0...
✅ Pulled wazuh/wazuh-manager:4.4.0
Pulling wazuh/wazuh-indexer:4.4.0...
✅ Pulled wazuh/wazuh-indexer:4.4.0
Pulling wazuh/wazuh-dashboard:4.4.0...
✅ Pulled wazuh/wazuh-dashboard:4.4.0
Scanning wazuh/wazuh-manager:4.4.0 with Trivy...
2025-08-29T21:16:59+01:00       INFO    Loaded  file_path="trivy/trivy.yaml"
2025-08-29T21:16:59+01:00       INFO    [vuln] Vulnerability scanning is enabled
2025-08-29T21:16:59+01:00       INFO    [secret] Secret scanning is enabled
2025-08-29T21:16:59+01:00       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-08-29T21:16:59+01:00       INFO    [secret] Please see also https://trivy.dev/v0.65/docs/scanner/secret#recommendation for faster secret de
tection
2025-08-29T21:16:59+01:00       INFO    Detected OS     family="ubuntu" version="20.04"
2025-08-29T21:16:59+01:00       INFO    [ubuntu] Detecting vulnerabilities...   os_version="20.04" pkg_num=159
2025-08-29T21:16:59+01:00       INFO    Number of language-specific files       num=1
2025-08-29T21:16:59+01:00       INFO    [gobinary] Detecting vulnerabilities...
2025-08-29T21:16:59+01:00       WARN    Using severities from other vendors for some vulnerabilities. Read https://trivy.dev/v0.65/docs/scanner/
vulnerability#severity-selection for details.
2025-08-29T21:16:59+01:00       WARN    This OS version is no longer supported by the distribution      family="ubuntu" version="20.04"
2025-08-29T21:16:59+01:00       WARN    The vulnerability detection may be insufficient because security updates are not provided
✅ wazuh/wazuh-manager:4.4.0 scan passed
Scanning wazuh/wazuh-indexer:4.4.0 with Trivy...
2025-08-29T21:17:01+01:00       INFO    Loaded  file_path="trivy/trivy.yaml"
2025-08-29T21:17:01+01:00       INFO    [vuln] Vulnerability scanning is enabled
2025-08-29T21:17:01+01:00       INFO    [secret] Secret scanning is enabled
2025-08-29T21:17:01+01:00       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-08-29T21:17:01+01:00       INFO    [secret] Please see also https://trivy.dev/v0.65/docs/scanner/secret#recommendation for faster secret de
tection
2025-08-29T21:17:54+01:00       INFO    Detected OS     family="ubuntu" version="20.04"
2025-08-29T21:17:54+01:00       INFO    [ubuntu] Detecting vulnerabilities...   os_version="20.04" pkg_num=92
2025-08-29T21:17:54+01:00       INFO    Number of language-specific files       num=1
```

```
✅ wazuh/wazuh-indexer:4.4.0 scan passed
Scanning wazuh/wazuh-dashboard:4.4.0 with Trivy...
2025-08-29T21:17:56+01:00       INFO    Loaded  file_path="trivy/trivy.yaml"
2025-08-29T21:17:56+01:00       INFO    [vuln] Vulnerability scanning is enabled
2025-08-29T21:17:56+01:00       INFO    [secret] Secret scanning is enabled
2025-08-29T21:17:56+01:00       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-08-29T21:17:56+01:00       INFO    [secret] Please see also https://trivy.dev/v0.65/docs/scanner/secret#recommendation for faster secret de
tection
2025-08-29T21:17:56+01:00       INFO    Detected OS     family="ubuntu" version="20.04"
2025-08-29T21:17:56+01:00       INFO    [ubuntu] Detecting vulnerabilities...   os_version="20.04" pkg_num=104
2025-08-29T21:17:56+01:00       INFO    Number of language-specific files       num=1
2025-08-29T21:17:56+01:00       INFO    [node-pkg] Detecting vulnerabilities...
2025-08-29T21:17:56+01:00       WARN    This OS version is no longer supported by the distribution      family="ubuntu" version="20.04"
2025-08-29T21:17:56+01:00       WARN    The vulnerability detection may be insufficient because security updates are not provided
✅ wazuh/wazuh-dashboard:4.4.0 scan passed
✅ Security scanning completed

==============================================
 Generating Certificates
==============================================

Creating network "multi-node_default" with the default driver
Creating multi-node_generator_1 ... done
Attaching to multi-node_generator_1
generator_1  | The tool to create the certificates exists in the in Packages bucket
generator_1  | 29/08/2025 20:18:05 INFO: Admin certificates created.
generator_1  | 29/08/2025 20:18:06 INFO: Wazuh indexer certificates created.
generator_1  | 29/08/2025 20:18:06 INFO: Wazuh server certificates created.
generator_1  | 29/08/2025 20:18:06 INFO: Wazuh dashboard certificates created.
generator_1  | Moving created certificates to the destination directory
generator_1  | Changing certificate permissions
generator_1  | Setting UID indexer and dashboard
generator_1  | Setting UID for wazuh manager and worker
multi-node_generator_1 exited with code 0
Aborting on container exit...
✅ Certificates generated
```

2.3-Production Pipeline

The GitHub Actions workflow automates the full lifecycle of the Mini SOC platform, from code quality validation to production deployment. The pipeline runs on a self-hosted Linux runner with Docker, Python, Ansible, Trivy, and Selenium installed.

The stages are as follows:

- **Quality Gates:** Runs Ansible linting and YAML linting, producing reports to ensure code quality and consistency.
- **Security Scan:** Pulls Wazuh Docker images and scans them with Trivy for HIGH and CRITICAL vulnerabilities. The pipeline fails if such issues are detected.

- **Certificate Generation:** Generates TLS certificates for secure communication between Wazuh components. Certificates are stored as artifacts.
- **Testing:** Deploys the stack in a temporary environment, waits for services to become ready, and executes automated tests:
  - API health checks.
  - Selenium browser tests for the dashboard UI.

  Results are stored for traceability.
- **Deployment Simulation:** Executes Ansible playbooks in `--check` mode using mock secrets. This dry-run ensures correctness before impacting production.
- **Production Deployment:** Uses real credentials from GitHub secrets and applies the Ansible playbooks to the production inventory. Services are verified post-deployment, and a deployment report is generated. A rollback step is triggered automatically on failure.

This design ensures that only tested and verified code reaches production, minimizing the risk of misconfiguration or downtime.

3-Deployment Approach

Currently, the deployment uses Docker Compose for reproducibility and ease of validation. A `docker-stack.yml` is included in the repository for future migration to Docker Swarm, enabling service scaling, high availability, and rolling updates.

3.1-Docker Compose Deployment

Docker Compose was selected as the primary deployment method to ensure reliable delivery within project constraints while maintaining full SOC functionality. This approach provides immediate operational capability with a clear migration path to more advanced orchestration when scaling requirements emerge.

The multi-node Wazuh cluster architecture is fully supported within the Docker Compose framework, with separate containers for manager nodes, worker nodes, indexers, and dashboard components. This design ensures proper service separation and prepares the foundation for horizontal scaling.

3.2-Docker Compose Commands

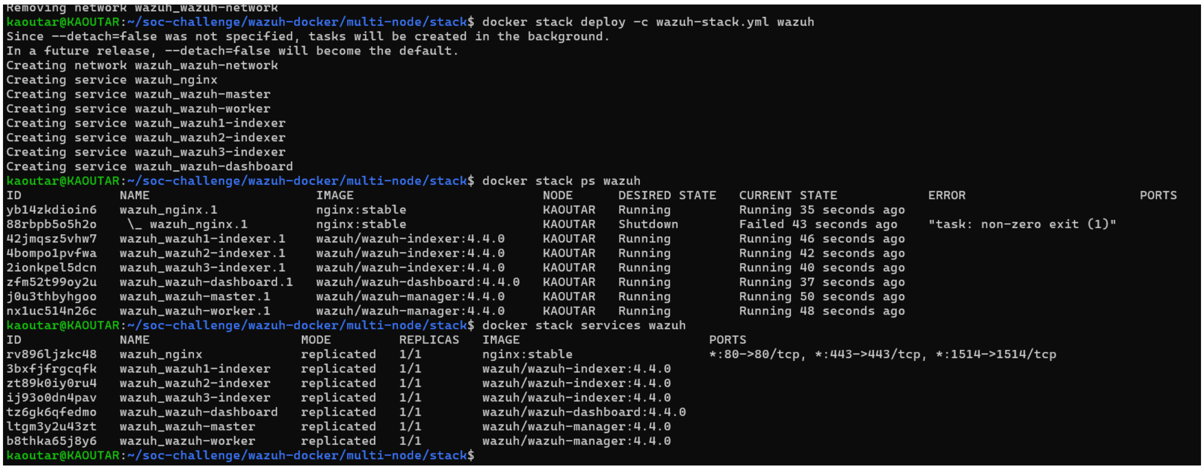
```
# Start all services in detached mode
docker-compose up -d
# Check status of running containers
docker-compose ps
# Follow logs of all services
docker-compose logs -f
# Stop and remove containers, networks
docker-compose down
# Stop and remove containers, networks, and volumes
docker-compose down -v
```

Why Docker Compose Over Swarm

- Time Constraints: Ensures project delivery within deadline while maintaining quality
- Operational Simplicity: Reduces deployment complexity for initial implementation
- Development Efficiency: Faster iteration cycles for testing and validation
- Migration Ready: All service definitions structured for seamless Swarm transition

3.2-Docker Swarm Deployment

The project includes a complete `docker-stack.yml` configuration prepared for Docker Swarm deployment. This migration path provides enhanced orchestration capabilities including service replication, automatic failover, and rolling updates across multiple nodes. The multi-node architecture translates directly to Swarm's distributed model, where manager nodes coordinate cluster operations while worker nodes handle service execution. The existing service definitions are fully compatible with Swarm's overlay networking and service discovery mechanisms.



```
Removing network wazuh_wazuh-network
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node/stack$ docker stack deploy -c wazuh-stack.yml wazuh
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network wazuh_wazuh-network
Creating service wazuh_nginx
Creating service wazuh_wazuh-master
Creating service wazuh_wazuh-worker
Creating service wazuh_wazuh1-indexer
Creating service wazuh_wazuh2-indexer
Creating service wazuh_wazuh3-indexer
Creating service wazuh_wazuh-dashboard
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node/stack$ docker stack ps wazuh
ID             NAME                       IMAGE                            NODE      DESIRED STATE   CURRENT STATE            ERROR                          PORTS
yb14zkdioin6   wazuh_nginx.1              nginx:stable                     KAOUTAR   Running         Running 35 seconds ago
88rbpb5o5h2o    \_ wazuh_nginx.1          nginx:stable                     KAOUTAR   Shutdown        Failed 43 seconds ago    "task: non-zero exit (1)"
42jmqsz5vhw7   wazuh_wazuh1-indexer.1     wazuh/wazuh-indexer:4.4.0        KAOUTAR   Running         Running 46 seconds ago
4bompo1pvfwa   wazuh_wazuh2-indexer.1     wazuh/wazuh-indexer:4.4.0        KAOUTAR   Running         Running 42 seconds ago
2ionkpel5dcn   wazuh_wazuh3-indexer.1     wazuh/wazuh-indexer:4.4.0        KAOUTAR   Running         Running 40 seconds ago
zfm52t99oy2u   wazuh_wazuh-dashboard.1    wazuh/wazuh-dashboard:4.4.0      KAOUTAR   Running         Running 37 seconds ago
j0u3thbyhgoo   wazuh_wazuh-master.1       wazuh/wazuh-manager:4.4.0        KAOUTAR   Running         Running 50 seconds ago
nx1uc514n26c   wazuh_wazuh-worker.1       wazuh/wazuh-manager:4.4.0        KAOUTAR   Running         Running 48 seconds ago
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node/stack$ docker stack services wazuh
ID             NAME                    MODE         REPLICAS   IMAGE                         PORTS
rv896ljzkc48   wazuh_nginx             replicated   1/1        nginx:stable                  *:80->80/tcp, *:443->443/tcp, *:1514->1514/tcp
3bxfjfrgcqfk   wazuh_wazuh1-indexer    replicated   1/1        wazuh/wazuh-indexer:4.4.0
zt89k0iy0ru4   wazuh_wazuh2-indexer    replicated   1/1        wazuh/wazuh-indexer:4.4.0
ij93o0dn4pav   wazuh_wazuh3-indexer    replicated   1/1        wazuh/wazuh-indexer:4.4.0
tz6gk6qfedmo   wazuh_wazuh-dashboard   replicated   1/1        wazuh/wazuh-dashboard:4.4.0
ltgm3y2u43zt   wazuh_wazuh-master      replicated   1/1        wazuh/wazuh-manager:4.4.0
b8thka65j8y6   wazuh_wazuh-worker      replicated   1/1        wazuh/wazuh-manager:4.4.0
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node/stack$
```

Docker Swarm Deployment Commands

```
# Initialize Docker Swarm on the manager node
docker swarm init

# Deploy the stack using the docker-stack.yml file
docker stack deploy -c wazuh-stack.yml wazuh

# List all services in the stack
docker stack services wazuh

# List running containers in the stack
docker stack ps wazuh

# Remove the deployed stack
docker stack rm wazuh
```

Swarm Migration Benefits

- High Availability: Service replication across multiple nodes with automatic failover
- Scalability: Dynamic service scaling based on resource demands
- Rolling Updates: Zero-downtime deployments with automatic rollback capabilities
- Load Balancing: Built-in load distribution across service replicas

- Service Discovery: Native DNS-based service discovery for inter-component communication

3.4-Ansible Automation of the Deployment

Ansible orchestrates the entire deployment process, ensuring consistency across environments and providing infrastructure-as-code capabilities. Automation handles prerequisites validation, service deployment, configuration management, and post-deployment verification.

```yaml
---
- name: Deploy Wazuh SOC
  hosts: localhost
  connection: local
  become: no
  gather_facts: yes

  vars:
    project_dir: "{{ ansible_env.PWD | regex_replace('/ansible.*', '') }}"

  tasks:
    - name: Check if docker-compose.yml exists
      stat:
        path: "{{ project_dir }}/docker-compose.yml"
      register: compose_file

    - name: Fail if compose file missing
      fail:
        msg: "docker-compose.yml not found at {{ project_dir }}"
      when: not compose_file.stat.exists

    - name: Check SSL certificates
      find:
        paths: "{{ project_dir }}/config/wazuh_indexer_ssl_certs"
        patterns: "*.pem"
      register: certs
    - name: Generate certificates if missing
      command: docker-compose -f generate-indexer-certs.yml run --rm generator
      args:
        chdir: "{{ project_dir }}"
      when: certs.matched == 0

    - name: Stop services
      docker_compose:
        project_src: "{{ project_dir }}"
        state: absent
      when: stop_services is defined and stop_services
      tags: [stop, never]

    - name: Deploy Wazuh services
```

```
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node/ansible/playbooks$ ansible-playbook -i inventohosts.yml deploy.yml
[WARNING]: Unable to parse /home/kaoutar/soc-challenge/wazuh-docker/multi-node/ansible/playbooks/inventohosts.yml as an inventory source
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'
[DEPRECATION WARNING]: community.docker.docker_compose has been deprecated. This module uses docker-compose v1, which is End of Life since July 2022. Please migrate to
community.docker.docker_compose_v2. This feature will be removed from community.docker in version 4.0.0. Deprecation warnings can be disabled by setting
deprecation_warnings=False in ansible.cfg.

PLAY [Deploy Wazuh SOC] ********************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************
ok: [localhost]

TASK [Check if docker-compose.yml exists] **************************************************************************************************
ok: [localhost]

TASK [Fail if compose file missing] ********************************************************************************************************
skipping: [localhost]

TASK [Check SSL certificates] **************************************************************************************************************
[WARNING]: Skipped '/home/kaoutar/soc-challenge/wazuh-docker/multi-node/config/wazuh_indexer_ssl_certs' path due to this access issue: [Errno 13] Permission denied:
'/home/kaoutar/soc-challenge/wazuh-docker/multi-node/config/wazuh_indexer_ssl_certs'
ok: [localhost]

TASK [Generate certificates if missing] ****************************************************************************************************
changed: [localhost]

TASK [Deploy Wazuh services] ***************************************************************************************************************
changed: [localhost]

TASK [Wait for services] *******************************************************************************************************************
Pausing for 30 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [localhost]

TASK [Check service status] ****************************************************************************************************************
changed: [localhost]

TASK [Show status] *************************************************************************************************************************
ok: [localhost] => {
    "msg": "            Name                             Command               State                                        Ports                                        \n----
------------------------------------------------------------------------------------------------------------------------------------------------------\nmulti-node_nginx_1
          /docker-entrypoint.sh ngin ...   Up      0.0.0.0:1514->1514/tcp,:::1514->1514/tcp, 80/tcp                                        \nmulti-node_wazuh.dashboard_1   /entr
ypoint.sh                   Up      443/tcp, 0.0.0.0:443->5601/tcp,:::443->5601/tcp                                        \nmulti-node_wazuh.master_1      /init
            Up      1514/tcp, 0.0.0.0:1515->1515/tcp,:::1515->1515/tcp, 1516/tcp, 0.0.0.0:514->514/udp,:::514->514/udp, 0.0.0.0:55000->55000/tcp,:::55000->55000/tcp\nmulti-node_wazuh.worker_1      /init                          Up
    1514/tcp, 1515/tcp, 1516/tcp, 514/udp, 55000/tcp                                        \nmulti-node_wazuh1.indexer_1    /entrypoint.sh opensearchw ...   Up      0.0.0.0:9200
->9200/tcp,:::9200->9200/tcp                                        \nmulti-node_wazuh2.indexer_1    /entrypoint.sh opensearchw ...   Up      9200/tcp
                                        \nmulti-node_wazuh3.indexer_1    /entrypoint.sh opensearchw ...   Up      9200/tcp
                                        "
}

TASK [Final status] ************************************************************************************************************************
ok: [localhost] => {
    "msg": "Deployment complete!\n\nAccess your services:\n- Dashboard: https://localhost:443\n- Manager API: https://localhost:55000\n- Indexer: https://localhost:9200\n\nCommands:\nDeploy: ansible-playbook ansible/playbooks/deploy.y
ml\nStop: ansible-playbook ansible/playbooks/deploy.yml --tags stop\nHealth: ansible-playbook ansible/playbooks/deploy.yml --tags health\nLogs: ansible-playbook ansible/playbooks/deploy.yml --tags logs\n"
}

PLAY RECAP *********************************************************************************************************************************
```

3.5-Security Scan with Trivy

Container security scanning is integrated into the deployment pipeline using Trivy, ensuring all images meet security standards before production deployment.

```
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node/trivy$ cat trivy.yaml
format: sarif
output: trivy-results.sarif
severity:
  - CRITICAL
  - HIGH
exit-code: 1
ignore-unfixed: true
timeout: 10m
ignorefile: trivy/.trivyignore  # Add this line
```

The security scanning process validates all Wazuh components against known vulnerabilities, with configurable severity thresholds. The pipeline fails automatically if critical or high-severity vulnerabilities are detected, preventing insecure deployments from reaching production environments.

```
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node$ trivy image --config trivy/trivy.yaml wazuh/wazuh-manager:4.4.0
trivy image --config trivy/trivy.yaml wazuh/wazuh-indexer:4.4.0
trivy image --config trivy/trivy.yaml wazuh/wazuh-dashboard:4.4.0
2025-08-29T20:22:33+01:00       INFO    Loaded  file_path="trivy/trivy.yaml"
2025-08-29T20:22:33+01:00       INFO    [vuln] Vulnerability scanning is enabled
2025-08-29T20:22:33+01:00       INFO    [secret] Secret scanning is enabled
2025-08-29T20:22:33+01:00       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-08-29T20:22:33+01:00       INFO    [secret] Please see also https://trivy.dev/v0.65/docs/scanner/secret#recommendation for faster secret detection
2025-08-29T20:22:33+01:00       INFO    Detected OS     family="ubuntu" version="20.04"
2025-08-29T20:22:33+01:00       INFO    [ubuntu] Detecting vulnerabilities...   os_version="20.04" pkg_num=159
2025-08-29T20:22:33+01:00       INFO    Number of language-specific files       num=1
2025-08-29T20:22:33+01:00       INFO    [gobinary] Detecting vulnerabilities...
2025-08-29T20:22:33+01:00       WARN    Using severities from other vendors for some vulnerabilities. Read https://trivy.dev/v0.65/docs/scanner/vulnerability#severity-selection for details.
2025-08-29T20:22:33+01:00       WARN    This OS version is no longer supported by the distribution      family="ubuntu" version="20.04"
2025-08-29T20:22:33+01:00       WARN    The vulnerability detection may be insufficient because security updates are not provided
2025-08-29T20:22:33+01:00       INFO    Loaded  file_path="trivy/trivy.yaml"
2025-08-29T20:22:33+01:00       INFO    [vuln] Vulnerability scanning is enabled
2025-08-29T20:22:33+01:00       INFO    [secret] Secret scanning is enabled
2025-08-29T20:22:33+01:00       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-08-29T20:22:33+01:00       INFO    [secret] Please see also https://trivy.dev/v0.65/docs/scanner/secret#recommendation for faster secret detection
2025-08-29T20:23:40+01:00       INFO    Detected OS     family="ubuntu" version="20.04"
2025-08-29T20:23:40+01:00       INFO    [ubuntu] Detecting vulnerabilities...   os_version="20.04" pkg_num=92
2025-08-29T20:23:40+01:00       INFO    Number of language-specific files       num=1
2025-08-29T20:23:40+01:00       INFO    [jar] Detecting vulnerabilities...
2025-08-29T20:23:40+01:00       WARN    This OS version is no longer supported by the distribution      family="ubuntu" version="20.04"
2025-08-29T20:23:40+01:00       WARN    The vulnerability detection may be insufficient because security updates are not provided
2025-08-29T20:23:41+01:00       INFO    Loaded  file_path="trivy/trivy.yaml"
2025-08-29T20:23:41+01:00       INFO    [vuln] Vulnerability scanning is enabled
2025-08-29T20:23:41+01:00       INFO    [secret] Secret scanning is enabled
2025-08-29T20:23:41+01:00       INFO    [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-08-29T20:23:41+01:00       INFO    [secret] Please see also https://trivy.dev/v0.65/docs/scanner/secret#recommendation for faster secret detection
2025-08-29T20:23:41+01:00       INFO    Detected OS     family="ubuntu" version="20.04"
2025-08-29T20:23:41+01:00       INFO    [ubuntu] Detecting vulnerabilities...   os_version="20.04" pkg_num=104
2025-08-29T20:23:41+01:00       INFO    Number of language-specific files       num=1
2025-08-29T20:23:41+01:00       INFO    [node-pkg] Detecting vulnerabilities...
2025-08-29T20:23:42+01:00       WARN    This OS version is no longer supported by the distribution      family="ubuntu" version="20.04"
2025-08-29T20:23:42+01:00       WARN    The vulnerability detection may be insufficient because security updates are not provided
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node$ |
```

4-Testing

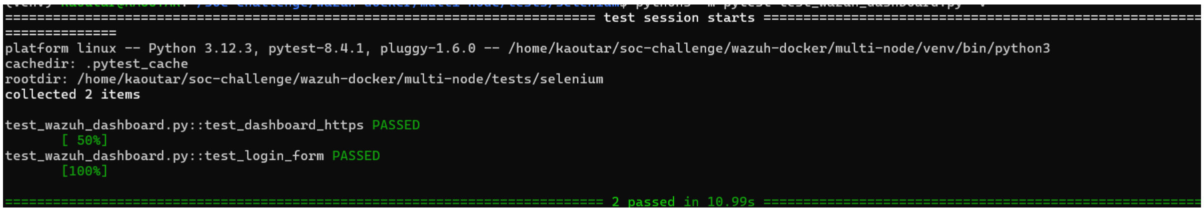
After successful deployment and security scan validation with Trivy, the pipeline proceeds to comprehensive testing to ensure all components of the Mini SOC platform are functioning correctly. The testing phase validates both backend API functionality and frontend user interface accessibility through automated test suites.

4.1-Test Execution Commands

The testing framework uses pytest to execute both API health checks and Selenium UI tests. Tests are organized in separate directories for clear separation of concerns.

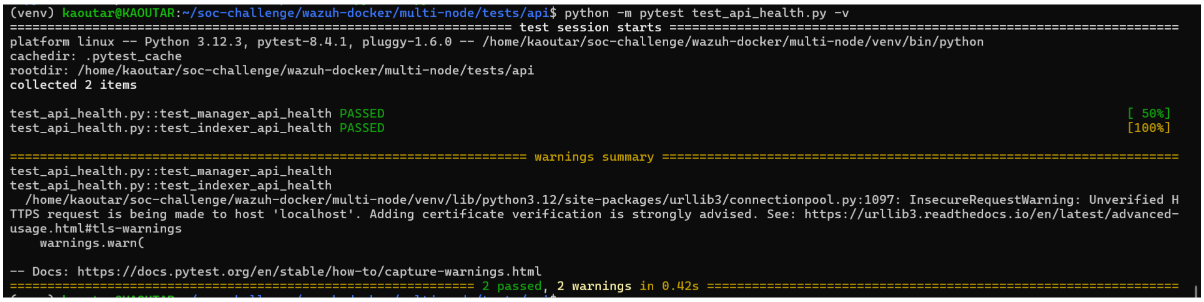
API Health Testing

```
python -m pytest tests/api/test_api_health.py -v
```



Selenium UI Testing

```
python -m pytest tests/selenium/test_wazuh_dashboard.py -v
```



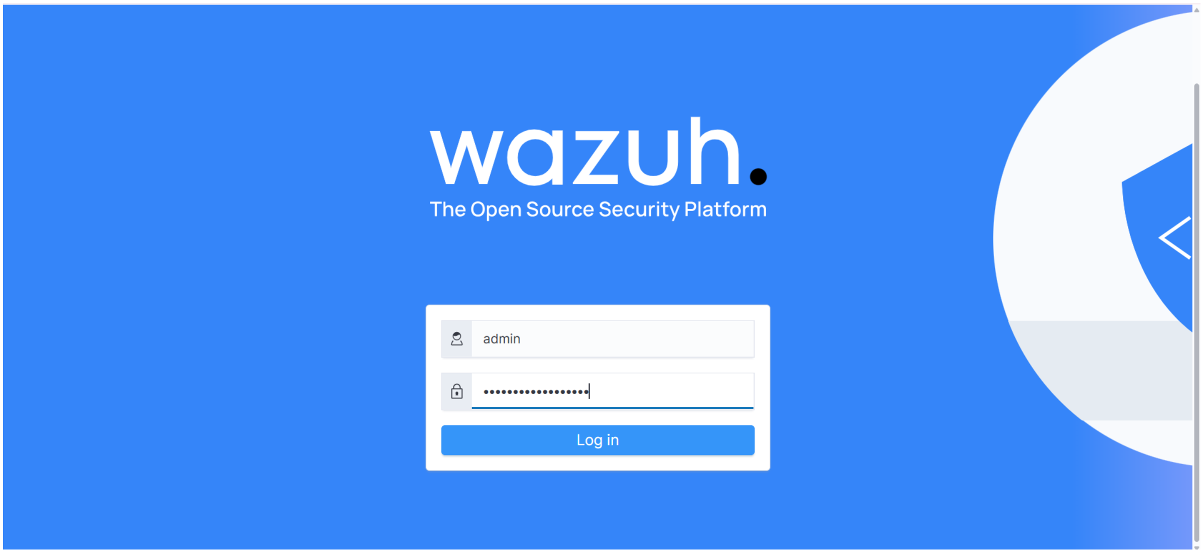
5-Results

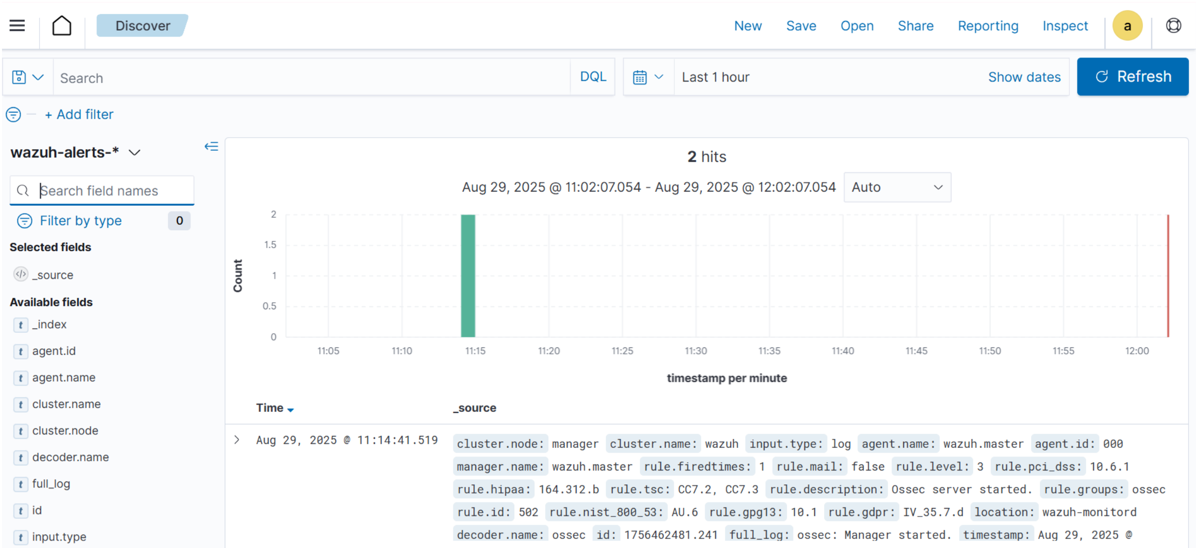
The successful execution of both test suites provides comprehensive validation that:

- **Backend Services:** The core Wazuh components (Manager and Indexer) are operational and communicating properly within the multi-node architecture
- **Frontend Interface:** The web dashboard is accessible via secure HTTPS connections with proper authentication interfaces
- **End-to-End Functionality:** The complete pipeline from security event processing (back-end) to analyst access (frontend) is functional

- **Security Implementation:** TLS certificates are properly deployed and functioning for secure communications
- **Operational Readiness:** These test results confirm the Mini SOC platform is fully deployed and ready for:
  - Security event ingestion and processing
  - Data storage and indexing for search capabilities
  - SOC analyst access via secure web interface
  - Authentication and authorization workflows
- **Multi-Node Architecture Validation:** The successful API health checks specifically confirm that the distributed architecture components can communicate effectively, ensuring the platform can handle production security monitoring workloads.

The combined results demonstrate that all critical components of the SOC infrastructure are operational, properly configured, and ready to support security operations center activities.





Part Two – Threat Detection Rule

1-Use Case

The custom rule detects multiple failed SSH login attempts from the same source IP followed by a successful login using a new user account. This scenario models a brute-force attack followed by a compromise of credentials.

2-Implementation

The custom Wazuh rule with ID 100001 successfully detects SSH brute force attacks that are followed by a successful authentication. This rule is configured within the required range for custom rules, and it is assigned a severity level of 10, which indicates high priority. It is mapped to the MITRE ATT&CK technique T1110, covering brute force and credential access.

The detection logic specifies that if there are three or more failed SSH attempts within a 60-second window followed by a successful login, the rule will trigger. Correlation is achieved using the same source IP along with the `if_matched_sid` directive to enable stateful tracking. The description for this rule is: “SSH brute force detected: Multiple failed invalid user attempts from 203.0.113.5.”

```
docker compose exec wazuh.master cat /var/ossec/etc/rules/local_rules.xml
<group name="local,ssh_bruteforce,">
  <!-- SSH Brute Force Detection: Multiple failures from same IP -->
  <rule id="100001" level="10" frequency="3" timeframe="60">
    <if_matched_sid>5710</if_matched_sid>
    <same_source_ip />
    <description>SSH brute force detected: Multiple failed invalid user attempts from $(srcip)</description>
    <mitre>
      <id>T1110</id>
    </mitre>
    <group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
  </rule>
  <!-- SSH Success (any successful auth) -->
  <rule id="100002" level="8">
    <if_matched_sid>5715</if_matched_sid>
    <description>SSH authentication success: User $(dstuser) from $(srcip)</description>
    <mitre>
      <id>T1078</id>
    </mitre>
    <group>authentication_success,pci_dss_10.2.5,</group>
  </rule>
</group>
```

3-Testing

The custom rule was tested using the `test_ssh_bruteforce.sh` script that simulates the attack pattern:

```
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node$ cat test_ssh_bruteforce_v2.sh
#!/bin/bash
echo "Injecting failed SSH attempts to monitored log..."
for i in {1..4}; do
    docker compose exec wazuh.master bash -c "echo 'Aug 29 $(date +%H:%M:%S) server sshd[1830]: Failed password for invalid user test1 from 203.0.113.5 port 50234 ssh2' >> /var/ossec/logs/active-responses.log
"
    echo "Failed attempt $i injected"
    sleep 1
done
echo "Waiting 2 seconds..."
sleep 2
echo "Injecting successful login..."
docker compose exec wazuh.master bash -c "echo 'Aug 29 $(date +%H:%M:%S) server sshd[1830]: Accepted password for validuser from 203.0.113.5 port 50234 ssh2' >> /var/ossec/logs/active-responses.log"
echo "Log injection complete. Check Wazuh dashboard for alerts."
```

```
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node$ ./test_ssh_bruteforce.sh
Injecting failed SSH attempts...
Failed attempt 1 injected
Failed attempt 2 injected
Failed attempt 3 injected
Failed attempt 4 injected
Waiting 2 seconds...
Injecting successful login...
Log injection complete. Check Wazuh dashboard for alerts.
```

The rule successfully triggered and generated alerts in the Wazuh dashboard, including complete metadata. The source IP of the attack was 203.0.113.5, and the attack pattern involved multiple failed authentication attempts followed by a successful login. The user account affected was test1, which was used as a simulated compromised account. All of this activity occurred within the 60-second detection window specified by the rule. The rule was part of several groups, including local, `ssh_bruteforce`, and `authentication_failed`.

```
echo "Log injection complete. Check Wazuh dashboard for alerts."
kaoutar@KAOUTAR:~/soc-challenge/wazuh-docker/multi-node$ docker compose exec wazuh.master grep "Rule: 100001" /var/ossec/logs/alerts/alerts.log | tail -3
Rule: 100001 (level 10) -> 'SSH brute force detected: Multiple failed invalid user attempts from 203.0.113.5'
Rule: 100001 (level 10) -> 'SSH brute force detected: Multiple failed invalid user attempts from 203.0.113.5'
Rule: 100001 (level 10) -> 'SSH brute force detected: Multiple failed invalid user attempts from 203.0.113.5'
```

The alerts are visible in the Wazuh dashboard under several sections. In the Discover tab, real-time alert generation can be monitored with filtering capabilities. The Rule Details section provides complete information about each rule, including the frequency, timeframes, and MITRE ATT&CK mapping. Additionally, the JSON Output section contains the full structured alert data, which can be used for SIEM integration and further analysis.

t	_index	wazuh-alerts-4.x-2025.08.29
t	agent.id	000
t	agent.name	wazuh.master
t	cluster.name	wazuh
t	cluster.node	manager
t	data.srcip	203.0.113.5
t	data.srcuser	test1
t	decoder.name	sshd
t	decoder.parent	sshd
t	full_log	Aug 29 12:36:32 server sshd[1830]: Failed password for invalid user test1 from 203.0.113.5 port 50234 ssh2
t	id	1756467394.4417
t	input.type	log
t	location	/var/ossec/logs/active-responses.log

t	predecoder.timestamp	Aug 29 12:36:32
t	previous_output	Aug 29 12:36:30 server sshd[1830]: Failed password for invalid user test1 from 203.0.113.5 port 50234 ssh2 Aug 29 12:36:29 server sshd[1830]: Failed password for invalid user test1 from 203.0.113.5 port 50234 ssh2
t	rule.description	SSH brute force detected: Multiple failed invalid user attempts from 203.0.113.5
#	rule.firedtimes	1
#	rule.frequency	3
t	rule.groups	local, ssh_bruteforce, authentication_failed
t	rule.id	100001
#	rule.level	10
	rule.mail	false
t	rule.mitre.id	T1110
t	rule.mitre.tactic	Credential Access
t	rule.mitre.technique	Brute Force
t	rule.pci_dss	10.2.4, 10.2.5
	timestamp	Aug 29, 2025 @ 12:36:34.130

Table JSON


```
{
  "_index": "wazuh-alerts-4.x-2025.08.29",
  "_id": "Aw_b9ZgBjh6JX3ZSbZCd",
  "_version": 1,
  "_score": null,
  "_source": {
    "predecoder": {
      "hostname": "server",
      "program_name": "sshd",
      "timestamp": "Aug 29 13:44:09"
    },
    "cluster": {
      "node": "manager",
      "name": "wazuh"
    },
    "agent": {
      "name": "wazuh.master",
      "id": "000"
    },
    "previous_output": "Aug 29 13:44:08 server sshd[1830]: Failed password for invalid user test1 from 203.0.113.5 port 50234 ss
h2\nAug 29 13:44:06 server sshd[1830]: Failed password for invalid user test1 from 203.0.113.5 port 50234 ssh2",
    "data": {
      "srcuser": "test1",
      "srcip": "203.0.113.5"
    },
```

```
    "manager": {
      "name": "wazuh.master"
    },
    "rule": {
      "firedtimes": 1,
      "mail": false,
      "level": 10,
      "pci_dss": [
        "10.2.4",
        "10.2.5"
      ],
      "description": "SSH brute force detected: Multiple failed invalid user attempts from 203.0.113.5",
      "groups": [
        "local",
        "ssh_bruteforce",
        "authentication_failed"
      ],
      "mitre": {
        "technique": [
          "Brute Force"
        ],
        "id": [
          "T1110"
        ],
        "tactic": [
          "Credential Access"
        ]
      },
      "id": "100001",
```

```
      "frequency": 3
    },
    "decoder": {
      "parent": "sshd",
      "name": "sshd"
    },
    "full_log": "Aug 29 13:44:09 server sshd[1830]: Failed password for invalid user test1 from 203.0.113.5 port 50234 ssh2",
    "input": {
      "type": "log"
    },
    "location": "/var/ossec/logs/active-responses.log",
    "id": "1756471451.18661",
    "timestamp": "2025-08-29T12:44:11.641+0000"
  },
  "fields": {
    "timestamp": [
      "2025-08-29T12:44:11.641Z"
    ]
  },
  "highlight": {
    "rule.id": [
      "@opensearch-dashboards-highlighted-field@100001@/opensearch-dashboards-highlighted-field@"
    ]
  },
  "sort": [
    1756471451641
  ]
}
```


Rule Effectiveness

The custom rule demonstrates practical security monitoring capabilities by:

- **Pattern Recognition:** Successfully correlates multiple failed attempts with eventual success
- **Threat Attribution:** Maps to MITRE ATT&CK framework for threat intelligence integration

- **Operational Integration:** Generates actionable alerts within the SOC workflow
- **Time-based Correlation:** Uses configurable time windows for accurate attack detection

This implementation aligns with real-world SOC requirements for detecting credential-based attacks and provides security analysts with actionable intelligence.

Future Work

Planned improvements include full Docker Swarm migration, high availability (multi-manager Swarm cluster, replicated indexers, redundant dashboards), disaster recovery planning (snapshots and restore), and optional enrichments such as geo-IP lookups, Slack/Teams integration, and automated SOAR playbooks.

1-Security Vulnerability Management

Container image vulnerabilities detected during Trivy scanning were accepted based on risk assessment. Many vulnerabilities originate from upstream Ubuntu base images and Wazuh source code dependencies, requiring vendor patches. Current mitigation includes network isolation, minimal exposed services, and planned quarterly security updates.

2-Platform Enhancements

- **High Availability:** Multi-node Swarm cluster with replicated services and cross-zone redundancy.
- **Disaster Recovery:** Automated OpenSearch snapshots, configuration backups, and cross-region replication.
- **Enrichments:** MaxMind geo-IP integration, Slack/Teams notifications, and automated SOAR playbooks for incident response.
- **Monitoring:** Prometheus metrics, Grafana dashboards, and enhanced logging with retention policies.
- **Compliance:** NIST framework alignment, audit logging, and SOC 2 readiness preparation.

Conclusion

This project demonstrates the deployment of a Wazuh-based Mini SOC with automated CI/CD, security scanning, and custom threat detection. The current implementation provides a production-like environment with clear paths toward Swarm-based scalability and high availability.