Wazuh SIEM Implementation Project Report

Executive Summary

This report documents the implementation of Wazuh as a Security Information and Event Management (SIEM) system on a Debian Linux environment with WordPress. The project successfully implemented the core functionality of collecting and analyzing security events from multiple data sources, though encountered some limitations with the dashboard visualization component due to resource constraints in the virtual environment.

Project Objectives

1. Configure Wazuh as a SIEM solution
2. Set up multiple data sources for security monitoring
3. Simulate various attack scenarios
4. Monitor and analyze security events

Implementation Process

1. Environment Setup

The project was implemented on a Debian virtual machine with the following specifications:

- Debian Linux OS
- Apache web server with WordPress installed
- VirtualBox as the virtualization platform

2. Installation of Wazuh Components

bash

```
# Add Wazuh repository
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | sudo apt-key add -
echo "deb https://packages.wazuh.com/4.5/apt/ stable main" | sudo tee /etc/apt/sources.list.d/wazuh.list

# Update package information
sudo apt update
```

```
# Install Wazuh components
sudo apt install wazuh-manager wazuh-agent wazuh-dashboard
wazuh-indexer -y
```

The installation included all core Wazuh components:

- **Wazuh Manager**: The central component that receives and analyzes data from agents
- **Wazuh Agent**: The component that collects local data and forwards it to the manager
- **Wazuh Dashboard**: The web interface for visualizing security events
- **Wazuh Indexer**: The search and analytics engine that stores collected data

3. Configuration of Multiple Data Sources

Apache Web Server Logs

The Wazuh agent was configured to monitor Apache access and error logs:

bash
```
# Edit the Wazuh agent configuration
sudo nano /var/ossec/etc/ossec.conf
```

Added the following configuration to monitor Apache logs:

xml
```
<localfile>
  <log_format>apache</log_format>
  <location>/var/log/apache2/access.log</location>
</localfile>
<localfile>
  <log_format>apache</log_format>
  <location>/var/log/apache2/error.log</location>
</localfile>
```

WordPress Error Logging

Enabled WordPress error logging to capture PHP errors and warnings:

bash
```
# Edit the WordPress configuration file
sudo nano /var/www/html/wp-config.php
```

Added these lines to enable logging:

php

```php
define('WP_DEBUG', true);
define('WP_DEBUG_LOG', true);
define('WP_DEBUG_DISPLAY', false);
```

This configuration creates a log file at `/wp-content/debug.log` where PHP errors related to WordPress are recorded.

4. Service Configuration and Startup

Started the Wazuh services to initialize the monitoring system:

bash

```bash
# Start Wazuh manager
sudo systemctl start wazuh-manager
sudo systemctl enable wazuh-manager

# Start Wazuh agent
sudo systemctl start wazuh-agent
sudo systemctl enable wazuh-agent

# Start Wazuh indexer
sudo systemctl start wazuh-indexer
sudo systemctl enable wazuh-indexer

# Start Wazuh dashboard
sudo systemctl start wazuh-dashboard
sudo systemctl enable wazuh-dashboard
```

5. Attack Simulation

Multiple attack scenarios were simulated to test Wazuh's detection capabilities:

a. Failed Authentication Attempts

Simulated unsuccessful SSH login attempts:

bash

```bash
# Generate multiple failed login attempts
ssh nonexistent@localhost
ssh fakuser@localhost
```

These commands attempted to log in with non-existent users, generating "Permission denied" errors that Wazuh would detect as potential brute force attacks.

b. File Integrity Monitoring

Modified a sensitive system file to trigger file integrity alerts:

bash
```
# Modify /etc/passwd file
echo "Simulating malicious change" | sudo tee -a /etc/passwd
```

This change would be detected by Wazuh's file integrity monitoring as a modification to a critical system file.

c. Web Server Activity

Generated normal and error web traffic to test Apache log monitoring:

bash
```
# Access the default webpage (normal behavior)
curl http://localhost/

# Generate a 404 error (suspicious activity)
curl http://localhost/nonexistent-page
```

The 404 error would be logged in Apache's error logs and detected by Wazuh as potential reconnaissance activity.

6. Verification of Wazuh Services

Verified that the Wazuh services were running:

bash
```
# Check status of Wazuh manager
sudo systemctl status wazuh-manager
# Output showed: active (running)

# Check status of Wazuh agent
sudo systemctl status wazuh-agent
# Output showed: active (running)

# Check status of Wazuh dashboard
sudo systemctl status wazuh-dashboard
# Output showed: active (running)
```

```
# Check status of Wazuh indexer
sudo systemctl status wazuh-indexer
# Output showed: failed
```

Challenges and Troubleshooting

Indexer Failure Issue

The Wazuh indexer service failed to start properly, as shown by the error messages:

```
wazuh-indexer.service: Main process exited, code=exited,
status=1/FAILURE
Failed to start wazuh-indexer.service - wazuh-indexer
```

The detailed logs showed:

```
May 09 20:17:44 debian systemd-entrypoint[66804]: For complete error
details, refer to the log at /var/log/wazuh-indexer/wazuh-cluster.log
May 09 20:17:45 debian systemd[1]: wazuh-indexer.service: Main
process exited, code=exited, status=1/FAILURE
```

Troubleshooting Steps Attempted

Checking system resources:

bash

```
free -m
```

1. `df -h`
   The VM appeared to have limited memory resources, which likely contributed to the indexer failure.
2. **Adjusting memory settings**:
   bash
   ```
   sudo sysctl -w vm.max_map_count=262144
   ```
   This increases the maximum number of memory map areas a process can have, which is often required for Elasticsearch/OpenSearch-based systems like the Wazuh indexer.

Modifying JVM heap settings:

bash

```
sudo nano /etc/wazuh-indexer/jvm.options
```

Attempted to reduce the memory footprint by setting lower heap sizes:

```
-Xms256m
```

3. `-Xmx256m`

Checking permissions:
bash
```
sudo chown -R wazuh-indexer:wazuh-indexer /var/lib/wazuh-indexer
```

4. `sudo chmod -R 750 /var/lib/wazuh-indexer`

Restarting services in order:
bash
```
sudo systemctl restart wazuh-indexer
sleep 180
sudo systemctl restart wazuh-manager
sleep 30
```

5. `sudo systemctl restart wazuh-dashboard`

Despite these troubleshooting efforts, the indexer service continued to fail due to resource constraints in the VM environment. The Wazuh dashboard depends on the indexer to store and retrieve security events, so it displayed "Wazuh dashboard server is not ready yet" when accessed.

Results and Analysis

Successful Components

1. **Installation of Wazuh components**: Successfully installed the Wazuh manager, agent, and dashboard packages.
2. **Configuration of multiple data sources**: Successfully configured the Wazuh agent to monitor:
    - Apache web server logs
    - System file changes
    - WordPress error logs
3. **Attack simulation**: Successfully simulated various attack scenarios:
    - Failed SSH login attempts generated "Permission denied" messages
    - Modified /etc/passwd file to trigger file integrity alerts
    - Generated web server 404 errors to simulate reconnaissance
4. **Service operation**: Successfully started and ran the Wazuh manager and agent services, which are the core components responsible for collecting and analyzing security events.

Limitations

The primary limitation encountered was the failure of the Wazuh indexer service due to resource constraints in the virtual environment. The Wazuh indexer is based on OpenSearch (a fork of Elasticsearch), which is known to be resource-intensive and requires significant memory.

This limitation affected:

- The ability to visualize security events through the Wazuh dashboard
- The persistence of historical security event data
- The ability to perform advanced searches and analytics on collected data

However, it's important to note that even with this limitation, the core SIEM functionality was still operational. The Wazuh manager and agent were still:

- Collecting logs from configured data sources
- Analyzing these logs for security events
- Generating alerts for detected security incidents

In a production environment with adequate resources, the indexer and dashboard would function correctly, providing the full visualization capabilities of the SIEM solution.

Conclusion

This project successfully demonstrated the setup and configuration of Wazuh as a SIEM solution for monitoring security events from multiple data sources. Despite the limitation with the indexer component due to resource constraints, the core functionality of collecting and analyzing security events was achieved.

The project showcased:

1. The ability to set up a comprehensive security monitoring solution
2. Configuration of multiple data sources for holistic security visibility
3. Detection of various attack scenarios through appropriate log monitoring
4. Understanding of SIEM architecture and components
5. Troubleshooting skills when dealing with resource limitations

For future implementations in similar resource-constrained environments, alternative approaches could be considered:

- Using a lighter-weight log analysis tool
- Focusing on direct log monitoring rather than a full SIEM dashboard
- Allocating more resources to the virtual environment, particularly memory
- Utilizing cloud-based SIEM solutions that can scale resources as needed

Appendix: Alternative Methods for Viewing Security Events

Without the dashboard, security events can still be monitored through:

bash

```
# View Wazuh logs directly
sudo cat /var/ossec/logs/ossec.log

# Filter for specific event types
sudo grep -i "sshd" /var/ossec/logs/ossec.log
sudo grep -i "apache" /var/ossec/logs/ossec.log
sudo grep -i "modified" /var/ossec/logs/ossec.log
```

These methods provide access to the security events detected by Wazuh, though without the advanced visualization offered by the dashboard.

Report done by: Theresa Baker
Date: May 9, 2025 at 7:33pm (CDT)