

Design and Evaluation of a Wazuh-Based Security Monitoring Framework for a Simulated OT Environment

Priyank Kumar Kartikey¹, Apoorva Khare²

1-PG Student, 2-Asst. Prof. CSE Dept.

Shri Ram Institute of Technology, Jabalpur

Abstract

Security monitoring is becoming more critical for systems that were primarily set up for reliable process control; industrial systems are increasingly connected. This project focuses on the feasibility of a small simulated operational technology (OT) setup with a monitoring solution based on Wazuh. A Windows PC was employed to serve as a software PLC, Kali Linux for a security-testing machine and Wazuh for the central monitoring platform. The simulated PLC transmitted data of the simulated temperature, pressure, tank level, motor status, and valve position as Modbus TCP and provided changing values. A variety of practical tests were run: Failed Windows authentication, PowerShell activity, Linux file-integrity events, Modbus communication checks, and Nmap service discovery. Wazuh obviously recognized the Windows authentication and PowerShell testing and also got file-integrity events from Kali. Modbus communication was successfully set up without any problems. But the execution of Nmap was not finally detected as a Wazuh alert and, in the same way, the EICAR antivirus test did not trigger a Wazuh event. The study also highlights the value of Wazuh for endpoint-focused visibility in an OT environment, which can be complemented by more network and OT-aware monitoring if comprehensive visibility in an industrial environment is desired.

Keywords

Operational Technology, Industrial Control Systems, Wazuh, SIEM, PLC, Modbus TCP, Cybersecurity Monitoring, File Integrity Monitoring, PowerShell, Network Security.

1. Introduction

Operational Technology (OT) systems are different from normal office IT systems because they are closely connected with physical processes. A failure of an industrial system can cause a loss of production, safety or availability of a critical service. Therefore, it is necessary to implement security monitoring carefully in OT. For OT security, NIST notes that characteristics like performance, reliability, availability and safety must be taken into account—not just IT security controls, which may need tweaking [1]. Meanwhile, enterprise and remote-access networks are becoming more commonly integrated with modern OT networks, making suspicious activity even more critical to monitor. The purpose of this project was to investigate this problem by developing a small simulated OT environment. A Windows-based computer with a software PLC was used and they communicated using Modbus TCP. Controlled security tests were conducted using Kali Linux and endpoint events were captured and analyzed by Wazuh. The intent was not to say that Wazuh can provide full protection for an industrial control system but to see what kind of security information can be derived from a real open-source monitoring platform.

1.2 Problem Statement

An issue with OT cybersecurity is that you can't monitor the network and endpoints without disrupting operations. There are many devices, both industrial ones, that do not have enough resources to support a traditional security agent. Meanwhile, engineering workstations and Windows or Linux operating system support can provide crucial security evidence including application logs, file changes, command runs and login failures. A monitoring solution must then offer valuable data from systems on which it is technically feasible to be monitored, whilst acknowledging systems where endpoint monitoring is inadequate. This project tackles that issue by using a simulated OT environment. The primary issue is to obtain meaningful security visibility into a software PLC and around the systems supporting it with Wazuh. The project also intentionally documents unsuccessful (or incomplete) detection attempts. This is significant because the results of a research experiment should not be reported as all successes. Even if an activity is done and not reported to the SIEM, the activity can be a sign of a monitoring gap and help guide future activities.

2. Related Work

Security requirements, in OT environments, have been the subject of much discussion in standards and security guidance. The relevance of NIST SP 800-82 Rev. 3 is particularly significant as it addresses the industrial environment, the importance of adjusting security controls and the fact that availability, reliability, and safety could be a different priority than conventional IT [1]. Wazuh is an open-source security monitoring architecture that has the agents gather information from the endpoint and send it to a central server for analysis [2][3]. It has the capability of collecting logs and File Integrity Monitoring that can detect changes in monitored files and can be very helpful for investigation [4]. Wazuh also offers ways to gather Windows events such as security and application events [5]. PowerShell activity is important from the attacker-behavior point of view because MITRE ATT&CK categorizes PowerShell as Command and Scripting Interpreter, technique T1059.001 [6]. MITRE ATT&CK T1078 [7] covers the valid-account behavior. These references are used for background when choosing test scenarios: authentication and PowerShell. The contribution of this project is to explore these capabilities together with a simulated PLC and Modbus TCP environment.

3. System Architecture

The laboratory was built using a VirtualBox host-only network so that the experiments remained separated from physical industrial equipment. The OT network used the 192.168.56.0/24 address range. The Windows computer at 192.168.56.1 acted as the software PLC. The PLC application listened for Modbus TCP communication on port 502 and generated changing values representing a simple industrial process. Kali Linux used 192.168.56.103 and acted as the security-testing endpoint. The Wazuh Manager was located at 192.168.56.102 and served as the central point for security-event analysis and visualization. The arrangement allowed the researcher to test communication and security activity without connecting the simulator to real industrial machinery. Wazuh agents were used on the Windows and Kali systems so that operating-system and endpoint events could be sent to the manager. One important architectural limitation is that the Wazuh endpoint agent is not, by itself, a dedicated OT network sensor. Therefore, the architecture is best understood as a security-monitoring layer around a simulated OT process. A future version could add passive network monitoring and protocol-aware analysis for Modbus traffic.

Component	Purpose	Address
Windows software PLC	Simulated OT process and Modbus server	192.168.56.1
Kali Linux	Security testing and Modbus client	192.168.56.103
Wazuh Manager	Central monitoring and alert analysis	192.168.56.102
Host-only OT network	Isolated laboratory communication	192.168.56.0/24

4. Implementation Methodology

The implementation was carried out gradually so that each part of the laboratory could be checked before the next experiment was started. First, the host-only network was verified and communication between the systems was tested. The Windows software PLC was then started and checked for normal process behavior. The PLC used Modbus TCP on port 502 and generated changing process values rather than a fixed set of numbers. After the PLC was stable, Wazuh agents on Windows and Kali were checked to confirm that events were reaching the Wazuh Manager. Security tests were then performed individually. The Windows endpoint was used for authentication and PowerShell tests, while Kali was used for Linux endpoint monitoring, Modbus communication, and controlled Nmap reconnaissance. Linux auditd was also investigated as a possible source of command-execution information. During this work, auditd was found to be installed but inactive; starting the service restored normal audit-event generation. A simple execution watch then produced an EXECVE audit record, confirming that the audit subsystem itself was working. The experiments were deliberately kept reversible, and the software PLC was returned to its normal running state after the controlled tests.

4.1 OT Baseline and Normal Operation

A successful communication was made with the simulated PLC before the evaluation of security events. Successfully connected Kali Linux to Windows PLC (192.168.56.1), and the test results of ICMP are as follows: 4 replies, no packet loss. If the software PLC was in operation, then a Modbus TCP port 502 was available. A PyModbus client worked successfully and connected to the PLC and read five process values from it on Kali. Temperature, pressure, tank level, motor state and valve position were these values. These were variable over time because the simulator continually changed the process and made it a more realistic baseline than a static response. It was essential to establish a baseline because it was a reference of normal activity prior to putting security in place. It also validated that the virtual network, Windows PLC application, Modbus service and Kali client all worked well. The baseline needs to be part of the final paper as it is evidence that later security observations were performed on a previously known working environment. However, since no equipment was connected no process behavior was lost in the software simulation.

```
C>python plc_dynamic.py
=====
SIMULATED OT PLC
=====
Modbus TCP
02

-----
Industrial Process
-----
Pressure x10
x10
Level
Temperature
Position
-----
Running...
Values change every 3 seconds.
Press any key to stop.
=====
70.1 C | Pressure=3.0 bar | Tank=59% | Motor=ON | Valve=48%
70.2 C | Pressure=2.9 bar | Tank=60% | Motor=ON | Valve=49%
70.3 C | Pressure=3.0 bar | Tank=60% | Motor=ON | Valve=47%
70.2 C | Pressure=3.1 bar | Tank=60% | Motor=ON | Valve=47%
70.1 C | Pressure=3.1 bar | Tank=59% | Motor=ON | Valve=49%
70.3 C | Pressure=3.1 bar | Tank=60% | Motor=ON | Valve=51%
```

Figure 1. Simulated OT PLC operating with dynamic process values.

68.9 C	Pressure=3.1 bar	Tank=70%	Motor=ON	Valve=47%
68.7 C	Pressure=3.2 bar	Tank=71%	Motor=ON	Valve=48%
68.5 C	Pressure=3.2 bar	Tank=71%	Motor=ON	Valve=50%
68.7 C	Pressure=3.3 bar	Tank=70%	Motor=ON	Valve=51%
68.9 C	Pressure=3.3 bar	Tank=71%	Motor=ON	Valve=52%
68.7 C	Pressure=3.4 bar	Tank=72%	Motor=ON	Valve=53%
68.7 C	Pressure=3.4 bar	Tank=73%	Motor=ON	Valve=55%
68.6 C	Pressure=3.3 bar	Tank=72%	Motor=ON	Valve=57%
68.8 C	Pressure=3.3 bar	Tank=71%	Motor=ON	Valve=55%
68.8 C	Pressure=3.3 bar	Tank=71%	Motor=ON	Valve=53%
68.8 C	Pressure=3.3 bar	Tank=72%	Motor=ON	Valve=55%
68.9 C	Pressure=3.3 bar	Tank=72%	Motor=ON	Valve=53%
68.9 C	Pressure=3.3 bar	Tank=71%	Motor=ON	Valve=53%
68.7 C	Pressure=3.3 bar	Tank=70%	Motor=ON	Valve=53%
68.5 C	Pressure=3.2 bar	Tank=71%	Motor=ON	Valve=51%
68.4 C	Pressure=3.2 bar	Tank=72%	Motor=ON	Valve=49%
68.6 C	Pressure=3.2 bar	Tank=71%	Motor=ON	Valve=47%
68.8 C	Pressure=3.2 bar	Tank=71%	Motor=ON	Valve=49%
69.0 C	Pressure=3.2 bar	Tank=70%	Motor=ON	Valve=51%
69.0 C	Pressure=3.3 bar	Tank=70%	Motor=ON	Valve=50%
68.9 C	Pressure=3.2 bar	Tank=70%	Motor=ON	Valve=51%
69.1 C	Pressure=3.2 bar	Tank=69%	Motor=ON	Valve=50%
69.3 C	Pressure=3.3 bar	Tank=70%	Motor=ON	Valve=50%
69.4 C	Pressure=3.2 bar	Tank=71%	Motor=ON	Valve=49%
69.2 C	Pressure=3.2 bar	Tank=71%	Motor=ON	Valve=47%
69.0 C	Pressure=3.2 bar	Tank=72%	Motor=ON	Valve=46%
68.9 C	Pressure=3.2 bar	Tank=72%	Motor=ON	Valve=44%
68.9 C	Pressure=3.2 bar	Tank=73%	Motor=ON	Valve=42%

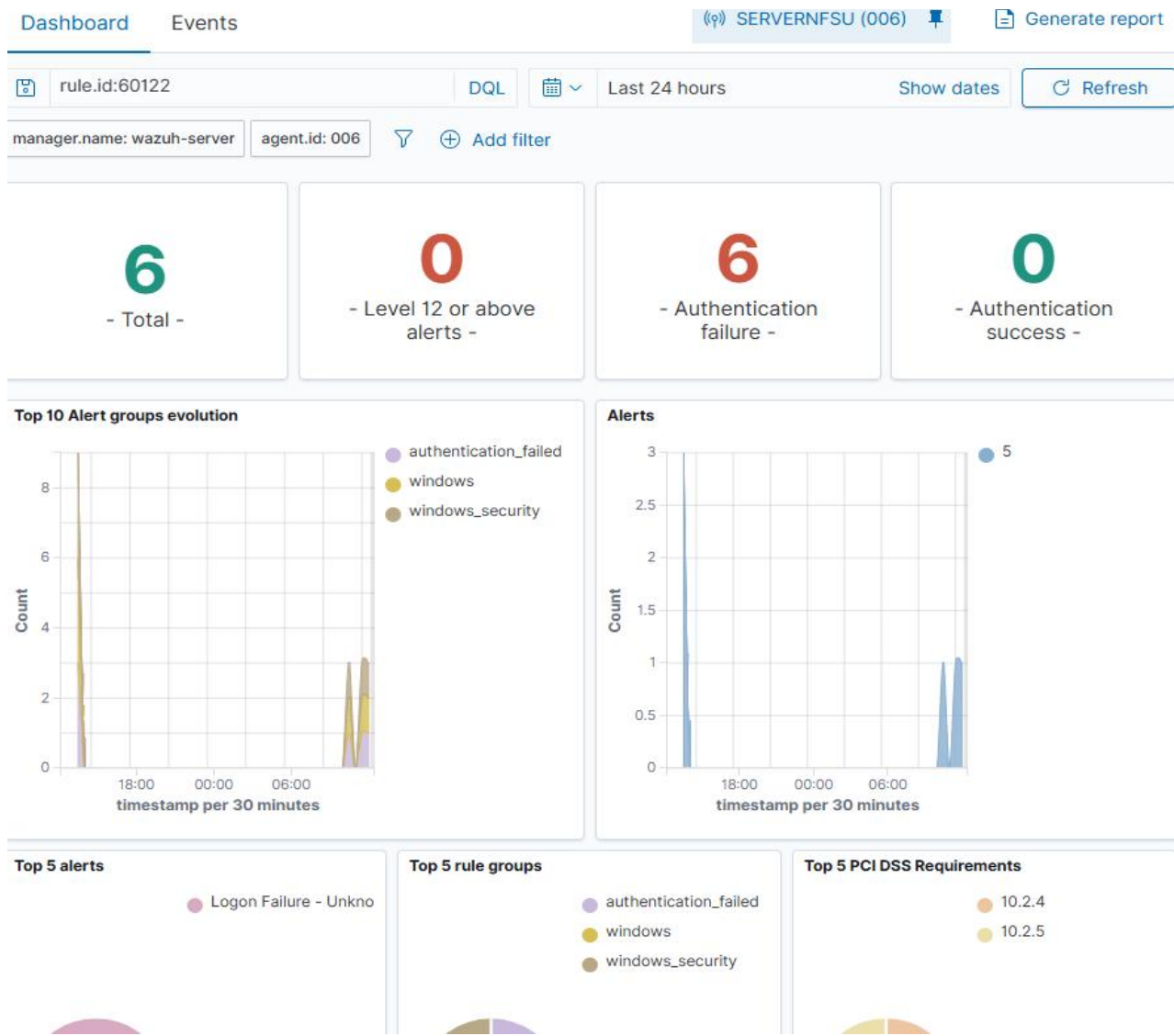
Figure 2. Kali Linux Modbus monitor receiving changing PLC process values.

4.2 Security Monitoring Experiments

The security experiments were chosen to include both common end-user activities, as well as OT-related activities. The first experiment involved a failed authentication attempt with the Windows endpoint. The second one was a benign PowerShell command to determine if script-interpreter activity is present in Wazuh. The third type of File Integrity Monitoring events that were examined were created using Kali. The fourth was network and Modbus checks against the simulated PLC with Nmap service discovery of Modbus port. An EICAR test file was also tested to check if an anti-virus related event can be seen on Wazuh. This file had been provided to EICAR entirely as a safe file to test anti-virus software; any real malicious code in this file has been removed [8]. Each test was administered separately and the results were kept confidential. The successful Nmap scan does not imply that the Wazuh detection was also successful; as they are two different events—opening a port and a SIEM alert. Similarly, if there was none, it was not regarded as a lack of Windows anti-virus protection, but rather a monitoring gap. The validated evidence is shown graphically in Figure 3 for authentication failures, and in Figures 1 and 2 for the baseline OT for comparison.

5. Results and Analysis

The Wazuh deployment demonstrated that it could actively give valuable insight into some endpoint events. The failed-authentication experiment yielded 6 alerts of authentication failure in the Wazuh dashboard. The observed detection rate for the particular test was 100% as 6 test events were observed and 6 test events were detected. We can also see PowerShell activity after a benign test command is run on Windows, indicating that pertinent endpoint information was being sent to Wazuh. On Kali Linux, File Integrity Monitoring events were fired on file creation, deletion and changes to the file checksum, proving that FIM telemetry was being collected. The OT baseline was also a success, the simulated PLC was able to be reached, Modbus TCP was able to be used, and process values were able to be read from Kali. Although Nmap is able to identify the Modbus service when PLC is running, the expected Nmap audit event was not strictly proved to be a Wazuh alert. Additionally, the Wazuh Dashboard didn't trigger any EICAR/Defender events in this period in the EICAR test. These findings will be helpful since they will provide insight into the positive aspects of the framework and areas in which further integration is necessary. Figure 1 and 2 show the simulated PLC and the related monitoring performed by Modbus while Figure 3 shows the actual Wazuh evidence for the authentication-failure experiment.



456 hits				
Sep 14, 2026 @ 16:19:56.226 - Sep 15, 2026 @ 16:19:56.226				
Export Formatted Reset view 718 available fields Columns Density 1 fields sorted Full screen				
timestamp	agent.name	rule.description	rule.level	rule.id
Sep 15, 2026 @ 16:18:16.1...	Kali	File deleted.	7	553
Sep 15, 2026 @ 16:18:16.1...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 16:18:16.0...	Kali	File deleted.	7	553
Sep 15, 2026 @ 16:18:16.0...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 12:36:33.2...	Kali	File deleted.	7	553
Sep 15, 2026 @ 12:36:33.2...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 12:36:33.2...	Kali	File deleted.	7	553
Sep 15, 2026 @ 12:36:33.1...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 12:34:01.1...	Kali	File deleted.	7	553
Sep 15, 2026 @ 12:34:01.1...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 12:33:56.1...	Kali	File deleted.	7	553
Sep 15, 2026 @ 12:33:56.1...	Kali	File deleted.	7	553
Sep 15, 2026 @ 12:33:56.1...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 12:33:56.0...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 12:32:44.1...	Kali	File deleted.	7	553
Sep 15, 2026 @ 12:32:44.0...	Kali	File added to the system.	5	554
Sep 15, 2026 @ 12:31:47.5...	Kali	File deleted.	7	553

Figure 3. Wazuh detection of repeated Windows authentication failures (Rule 60122).

Test	Observed outcome	Wazuh result
Windows failed authentication	6 failure alerts observed	Detected
PowerShell activity	Test activity visible in dashboard	Detected
Kali FIM	File add/delete/checksum events observed	Detected
Modbus baseline	Five PLC values read successfully	Verified
Nmap service discovery	TCP/502 discovered when PLC was running	Wazuh detection inconclusive

5.1 Evaluation Metrics

The project employs simple measurement scales to be able to be repeated in the future. The main metric is the "detection" status, which is considered detected if an activity generates a useful event or alert in Wazuh. If the number of events generated is known, the rate of detection can be calculated by taking the number of events detected and multiplying by 100 and the total number of events generated. The Windows authentication experiment offers the best numbers, six failures out of six, which gives 100% detection rate for this experiment. Another useful measure is detection latency, which was not systematically measured in this study. It should be measured in the future experiments by determining the event-generation time, and the time the warning is raised in Wazuh. False positives are also relevant to OT as too many alerts can make it more difficult to spot true alarms. This study is a small lab study and the results should not be reported as a statistical benchmark. Rather, they exhibit a repetitive process of experimentation that could be repeated and extended through extra trials and extra sensors.

5.3 Discussion

The experiments suggest that Wazuh is a useful part of a security-monitoring design for a simulated OT environment, particularly when important supporting systems are Windows or Linux endpoints. The successful authentication and PowerShell detections show that normal endpoint security information can be collected centrally and investigated from the Wazuh dashboard. The Kali FIM events provide another

example of how changes on a supporting system can be made visible. At the same time, the experiments show why OT monitoring cannot depend only on endpoint agents. A PLC may not be capable of running an agent, and network behavior between an engineering workstation and a controller can be security-relevant even when neither endpoint produces a conventional security alert. The Nmap experiment illustrates this distinction clearly: the reconnaissance activity could be performed and the Modbus service could be identified, but that does not by itself prove that Wazuh detected the network activity. The EICAR result also indicates that security-product telemetry needs to be deliberately integrated if it is expected to appear in the SIEM. Overall, Wazuh should be viewed as a useful monitoring layer rather than a complete standalone OT intrusion-detection system.

5.4 Reproducibility Procedure

This project has been designed to create and test a monitoring structure using Wazuh around a small and simulated OT environment. The Windows software PLC, Kali Linux, Modbus TCP communication and centralized monitoring via Wazuh were combined successfully in the laboratory. Prior to security testing, PLC communication was established as normal and Kali could read the simulated temperatures, pressure, tank, motor and valve values. Wazuh managed to detect the attempts to authenticate to windows, the PowerShell activity, and got the file integrity events from the Kali endpoint. The outcomes demonstrate that Wazuh can add more visibility to a number of critical security events related to a simulated OT process. In the same time, the Nmap reconnaissance test was not detected by Wazuh and the EICAR test did not trigger a Wazuh alert. The overall conclusion is thus balanced. While Wazuh serves as a valuable and helpful tool for centralized endpoint monitoring in a simulated OT environment, it is best used alongside network-level and protocol-aware monitoring tools for more complete OT visibility.

5.5 Figures and Evidence

The first step that a researcher would need to take to replicate this work is to create an isolated host-only network and set stable addresses for the simulated PLC, Kali Linux and Wazuh Manager. The software PLC should be turned on and monitored for good dynamic process output. Network connectivity should follow on from Kali followed by a Modbus TCP read test. It is important to verify if the Wazuh agents are active before creating any security events. A failed-authentication test can then be done on Windows and verified in the Wazuh dashboard. A harmless PowerShell command can be used for the second endpoint experiment. The current Wazuh configuration for file-integrity checking can be used to detect kali file-integrity checking activity. This simulated modbus service can be checked by using the nmap tool but it can only be said to be detected in Wazuh if its modbus telemetry data is actually detected in Wazuh. For antivirus-integration testing you do not have to use real malware; you can use the EICAR test. All experiments should record screenshots, the time, a rule ID, the severity level, and the outcome observed in the experiment.

5.6 Results Summary

Experiment	Status	Main evidence
Failed Windows authentication	Successfully detected	6 authentication-failure alerts
PowerShell activity	Successfully detected	PowerShell test activity visible in Wazuh
Kali File Integrity Monitoring	Successfully detected	File added/deleted/checksum

		events
Modbus TCP baseline	Successfully verified	Successful PLC connection and register reads
Nmap OT reconnaissance	Performed; inconclusive	TCP/502 discovered; Wazuh alert not conclusively verified

5.7 Limitations

The project employs simple measurement scales to be able to be repeated in the future. The main metric is the "detection" status, which is considered detected if an activity generates a useful event or alert in Wazuh. If there are known numbers of events generated, the rate of detection can be determined by multiplying the number of events detected by 100 and the total number of events generated. The Windows authentication experiment has the best numbers with six failures out of six, which means that the detection rate for this experiment is 100%. Another useful measure is detection latency, but this was not systematically measured in this study. It should be measured in the future experiments by determining the event-generation time, and the time the warning is raised in Wazuh. False positives also have significance in OT as false alarms can make it more difficult to detect alarms. This study is a small lab study and the results should not be reported as a statistical benchmark. Instead, they show a recursive process of experimentation which can be repeated and extended with additional trials and additional sensors.

5.8 Future Work

There are a number of limitations to the project that should be considered when interpreting the results. The biggest drawback is the use of a software simulation of the PLC. It offers useful Modbus TCP behavior and changing of process values, but doesn't replicate all aspects of a physical PLC, industrial network, engineering workstation or real process. Additionally, the network has relatively few systems, making it impossible to directly extrapolate the results to a large industrial facility. A second drawback is that the Nmap auditd experiment wasn't successfully proven to be a Wazuh detection. The audit subsystem was successfully restored but tested with a simple executable watch, but the last Nmap event was not verified in Wazuh. The EICAR test also didn't trigger the Wazuh alert. The number of repetitions was not sufficient for detecting detection latency or making statistical conclusions regarding false-positive rates. Lastly, there were no dedicated passive OT network sensor or protocol-aware IDS. The restrictions listed must be openly stated not only to describe what the research actually shows, but also to indicate the most fruitful avenues for enhancing the laboratory.

6. Conclusion

Monitoring should be added to the next phase of the project that would be able to directly observe OT network behavior, not primarily endpoint logs. The passive network sensor could be monitoring the Modbus TCP communications traffic without software being installed on the simulated PLC. It would allow for the analysis of unusual Modbus requests, unexpected write operations, repeated connection attempts, and reconnaissance patterns. The Windows endpoint could also be configured to forward more detailed Microsoft Defender and PowerShell operational telemetry to be correlated with Wazuh alerts by security-product events. The Nmap experiment can be repeated once the persistent audit configuration has been fully validated and the event generated by Nmap is correctly mapped in Wazuh. A good experiment would be to

send a controlled Modbus write message to the simulator and then automatically return to the process value to its normal state. Many repetitions of each experiment would allow for more accurate calculation of average detection latency, detection rate and false-positive behavior. Lastly, experiments could be mapped to MITRE ATT&CK for ICS and NIST OT security guidance to create a stronger link between what was observed in real tests and existing cybersecurity frameworks.

References

- [1] K. Stouffer et al., “Guide to Operational Technology (OT) Security,” NIST Special Publication 800-82 Rev. 3, National Institute of Standards and Technology, 2023. <https://csrc.nist.gov/pubs/sp/800/82/r3/final>
- [2] Wazuh, “Architecture,” Wazuh Documentation. <https://documentation.wazuh.com/current/getting-started/architecture.html>
- [3] Wazuh, “Wazuh agent,” Wazuh Documentation. <https://documentation.wazuh.com/current/getting-started/components/wazuh-agent.html>
- [4] Wazuh, “File Integrity Monitoring: How it works,” Wazuh Documentation. <https://documentation.wazuh.com/current/user-manual/capabilities/file-integrity/how-it-works.html>
- [5] Wazuh, “Configuring log collection for different operating systems,” Wazuh Documentation. <https://documentation.wazuh.com/current/user-manual/capabilities/log-data-collection/configuration.html>
- [6] MITRE ATT&CK, “Command and Scripting Interpreter: PowerShell, T1059.001.” <https://attack.mitre.org/techniques/T1059/001/>
- [7] MITRE ATT&CK, “Valid Accounts, T1078.” <https://attack.mitre.org/techniques/T1078/>
- [8] EICAR, “Anti-Malware Testfile,” European Institute for Computer Antivirus Research. <https://www.eicar.org/download-anti-malware-testfile/>