

## Article

# Agentless Approach for Security Information and Event Management in Industrial IoT

Huma Zahid <sup>1</sup>, Sadaf Hina <sup>2,\*</sup>, Muhammad Faisal Hayat <sup>1</sup> and Ghalib A. Shah <sup>3</sup><sup>1</sup> Department of Computer Engineering, University of Engineering and Technology, Lahore 54890, Pakistan<sup>2</sup> Department of Computer Science, School of Science, Engineering and Environment, University of Salford, Manchester M5 4WT, UK<sup>3</sup> Al-Khwarizmi Institute of Computer Science (KICS), University of Engineering and Technology, Lahore 54890, Pakistan

\* Correspondence: s.hina@salford.ac.uk

**Abstract:** The Internet of Things (IoT) provides ease of real-time communication in homes, industries, health care, and many other dependable and interconnected sectors. However, in recent years, smart infrastructure, including cyber-physical industries, has witnessed a severe disruption of operation due to privilege escalation, exploitation of misconfigurations, firmware hijacking, malicious node injection, botnets, and other malware infiltrations. The proposed agentless module for Wazuh security information and event management (SIEM) solution contributes to securing small- to large-scale IoT networks of industry 4.0. An agentless module is implemented by vigilantly examining the IoT device traffic without installing any agent or software on the endpoints. In the proposed research scheme, a module sniffs the network traffic of IoT devices captured from the gateway and passes it to a machine learning model for initial detection and prediction. The output of the ML model is embedded in the JSON log format and passed through the Wazuh agent to the Wazuh server where a decoder is added that decodes the network traffic logs. For event monitoring in Wazuh, industrial protocols are also thoroughly analyzed, and the feature set is determined. These features are used to write rules which are tested on the SWaT dataset, utilizing a common industrial protocol (CIP) for communication. Custom and dynamic rules are written at the Wazuh end to generate alerts to respond to any anomaly detected by the machine learning (ML) model or in the protocols used. Finally, in case of any event or an attack is detected, the alerts are fired on the Wazuh dashboard. This agentless SIEM solution has practical implications for the security of the industrial control systems of industry 4.0.

**Keywords:** industrial IoT; machine learning; cybersecurity; SIEM solution; agentless; SWaT

**Citation:** Zahid, H.; Hina, S.; Hayat, M.F.; Shah, G.A. Agentless Approach for Security Information and Event Management in Industrial IoT.

*Electronics* **2023**, *12*, 1831. <https://doi.org/10.3390/electronics12081831>

Academic Editor: Jose Santamaria

Received: 13 December 2022

Revised: 23 March 2023

Accepted: 29 March 2023

Published: 12 April 2023



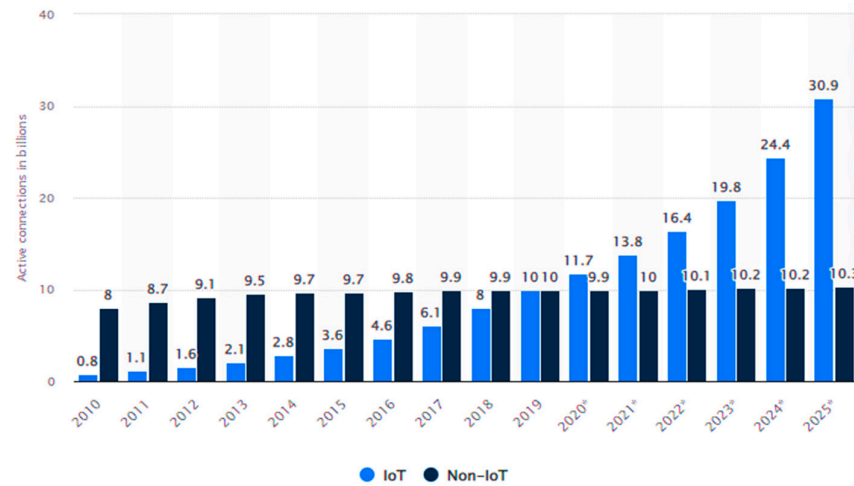
**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

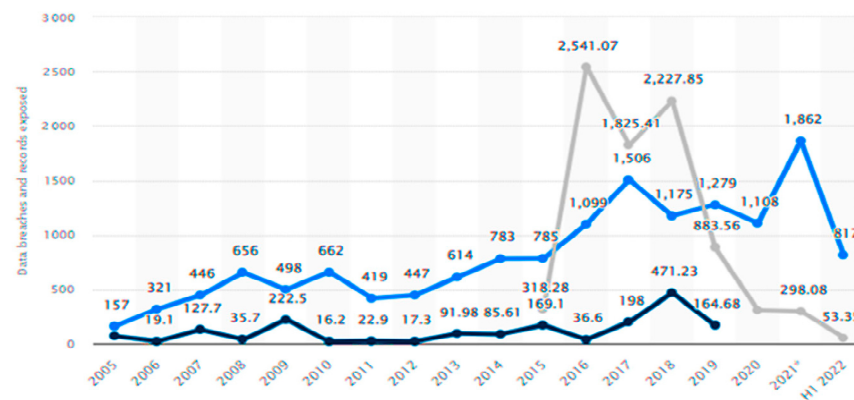
In recent years, the enormous growth of IoT devices has been witnessed, where billions of internet-connected devices are used around the world, exchanging substantial, detailed, and sensitive contextual data about the environment and users each day. Recent statistics show that the number of IoT-connected devices is estimated to increase up to 30.9 billion by 2025, as shown in Figure 1 years\* forecasts, and the damages related to cybercrime are projected to hit more than \$6 trillion annually by 2025 [1].

As IoT devices perform various exchange activities, there is always a concern about privacy, security, and an impact on the confidentiality, integrity and availability (CIA) of the communicated information. Different components of the connected world create different challenges and problems for the security of cyber-physical systems. Over the years, the equation of cyber security has diversified from militaries to individuals (Figure 2), from universities to hospitals and from the aviation industry to manufacturing. Over time, a huge number of IoT attacks, such as ransomware, phishing, denial of service, privilege escalation, and configuration manipulation, have been reported to cause disastrous financial,

legislative, and reputational exposures [2]. The attack paradigm is shifting from general system attacks to IoT settings. From January to June 2021, the Kaspersky data breach report stated around 1.51 billion IoT attacks [3]. To avoid the exploitation of vulnerabilities, IoT devices should be secured using dedicated security tools [4].



**Figure 1.** Stats of Active Connections in Billions from 2010 to 2025 on Statista [1]. Reprinted/adapted with permission from Ref. [1]. Copyright 2022, Statista. \* shows forecasts.



**Figure 2.** Data Breaches in Past Years on Statista [1]. Reprinted/adapted with permission from Ref. [1]. Copyright 2022, Statista. \* shows forecasts.

The industrial internet of things (IIoT) is a network of intelligent devices that are linked together to create an IoT ecosystem that exchanges, stores, and analyzes the data. A typical industrial IoT ecosystem includes infrastructure for public and/or private data communications, ranging from sensors, actuators, and data stores to complex industrial robots. More specifically, IIoT refers to all connected instruments and devices that, when combined with industrial applications such as production and energy management, create a complex network of services that enable automation at a higher level. Connected sensors and actuators enhance business intelligence by allowing businesses to identify inefficiencies and malfunctions in advance, saving time and money. However, the constant expansion of the connections and usage of industry-standard communication protocols makes it imperative to safeguard vital industrial systems from cyber security risks [5]. In addition to having a continual connection to the internet and industrial networks, the industrial systems that regulate the production process and functioning of smart factories also have access to the data and information of their affiliated business organizations. Industrial control systems (ICS) is a common term for such an arrangement [6].

Industrial protocols are the real-time communication protocols that are used to interact and communicate in a specific industrial scenario, among the control devices used in the IIoT environment. The protocols such as common industrial protocol (CIP), Modbus, distributed network protocol 3 (DNP3), message queuing telemetry transport (MQTT) are some of the industrial protocols frequently used in industrial control systems. These are the application layer protocols and have specific function codes and fields that contain the information that is needed to be exchanged between the field IIoT devices. These protocols are often secured by transport layer security (TLS) protocols that share secret keys and certificates for secure communications. Some of the protocols are object-based; each device in the network is an object and the objects request and respond according to the functions of the industrial control systems. The industrial protocols have specific function codes that are predefined for the recognition of specific tasks, i.e., read and write requests and responses. Anomaly or attack detection of the networks that are using these types of protocols can be performed by writing the rules against the function codes and port numbers and using the combinations of the fields of the protocols.

Adversaries intrude on the privacy of organizations, homes, or industries by exploiting the vulnerabilities in the IoT devices incorporated for automation and human ease. A typical example of espionage in industrial IoT includes compromising the control center through any susceptible sensor or programmable logic controller (PLC). Similarly, smart home systems can be compromised through a single susceptible device surrendering the entire system [7]. The attacks can be performed on devices, communication networks, or on software applications that are used to control IoT devices. Malware can be injected into the devices compromising firmware that erroneously updates the controls and other parts such as memory and connection points [8]. IIoT security, with a range of potential threats, attack surfaces, and vectors, is a challenging task. However, it can be managed by skillfully securing the layered architecture of the IIoT network. Each layer has its vulnerabilities and limitations that must be considered to assure its security by shielding it from various distinct forms of attacks [9].

The crucial need to protect private and public data stored on distinct IT infrastructures, utility infrastructures, cloud data centers, personal machines, PCs, laptops, tablets, smartphones, and IoT devices escalates here. More data means more digital footprint and a higher risk of cyber espionage. The enormous growth of IoT devices has attracted hackers to perform diverse attacks on IoT devices as they generally lack built-in security features to counter the threats. The attackers take advantage of the continuous connectivity of IoT devices to the internet and deploy distinctive attacks including denial of service, malware infiltration, man-in-the-middle, and firmware hijacking, to name a few. Common approaches for the detection and prediction of attacks in such networks are machine and deep learning models. The machine learning models work by getting trained on the historical network data and predicting the incoming traffic based on the previous data patterns. Both supervised and unsupervised ML algorithms such as decision trees (DT), random forest (RF), support vector machines (SVM) and logistic regressions (LR), K-nearest neighbor (KNN), neural networks, and K-mean clustering have been used in security prediction models. Attacks on IoT communication, such as denial of service, SYN flooding, and man-in-the-middle can be effectively detected using machine learning models. The right algorithm for the best detection of the attacks on a specific type of network can be decided by applying the model and calculating the performance measure such as accuracy, precision, recall, and the F1-score of the model. These machine learning-based computational models are deployed on the networks through firewalls, intrusion detection and prevention systems, and other dedicated security tools.

As industries and enterprises are heavily moving towards sensor-based automated devices and rigorously producing huge amounts of data through vulnerable communication protocols, a security information and event management (SIEM) solution with the latest IoT data and network protocols and standards is necessary. Wazuh is an open-source SIEM solution, and its architecture is mainly based on the agents that run on the monitored

hosts and send security information to a centralized SIEM server. By integrating separate functions into a single agent and platform architecture, the SIEM server ensures endpoint security, threat intelligence, detailed security operations, and even cloud security. This research focused on the limitation of the computational ability of IoT/IIoT devices to run the agents on them, especially the ones having operating system dependencies. The researchers signified that a holistic approach was needed to protect the IoT/IIoT devices without compromising their computational power and installing any agent or running any script on them to collect the sensitive communication data securely. Knowing the fact that IoT devices continuously exchange data through the gateway, the agentless approach is carried out by sniffing the traffic from the gateway and passing it on to the machine learning model for initial detection of anomalous traffic, and the generated ML model prediction is converted into JSON logs, which are transmitted to the Wazuh SIEM for further rule-based security monitoring. In the proposed scheme, Wazuh uses the log forwarder installed on the system, on the gateway, which is responsible for passing the generated logs to the server for IIoT protocol analysis, decoding, and dynamic rule writing. The main contributions of this research are:

- i. Attack detection at the gateway in industrial IoT devices using ML models.
- ii. Incorporating the predicted output of ML models within the Wazuh SIEM.
- iii. Dynamic rule writing for alert generation based on predictions.
- iv. Industrial protocol analysis and feature extraction for custom rule writing.
- v. Testing and validation of rules for event monitoring on the SWaT dataset.

In this research study, enhanced security is implemented using machine learning predictions, dynamic rule writing, and industrial SIEM solution for industrial IoT devices. The SIEM solution, which is a powerful tool used for the security and monitoring of remote devices, is integrated with the security framework of this research. The machine learning model is implemented at the gateway level for detection and log generation. Dynamic rules on ML prediction and the rules based on the protocols used in the industrial cyber-physical system are written on the SIEM side for further threat intelligence and event monitoring. The proposed agentless scheme works without installing any agent or running any script on the endpoints, i.e., IoT/IIoT devices.

The rest of the paper is arranged as follows: Section 2 contains the literature review and comparison of this research with the previous studies, while Section 3 defines the experimental design components. In Section 4, the methodology is explained, and the results are discussed in Section 5. Finally, the conclusion and future work are presented in Section 6.

## 2. Literature Review

As IoT/IIoT is a network of household appliances in smart homes and industrial devices such as PLCs, sensors, actuators, and human machine interfaces (HMIs) in an industrial environment; a lot of sensitive data are sent and received from one device to another. This routed data are critical to ongoing operations and requires security to avoid risks of exploitation and malicious attempts of an adversary [10]. In a recent research work [11], Splunk SIEM was implemented to improve the security of an organization. The Splunk SIEM tool uses universal forwarding agents (UFA) installed on the devices to be monitored and forwards the data to the PC having Splunk Enterprise for further processing, i.e., indexing and searching for the events. In the same research, the authors presented four rules for real-time monitoring of the event logs and scheduled reporting. The alerts generated by implemented rules were forwarded by email notification.

In [12], researchers used the hypertext transfer protocol (HTTP) and transmission control protocol (TCP) detection models with efficient machine learning models (decision trees) for mobile malware app detection. The network traffic to be monitored was collected using the tcpdump tool, which was then mirrored to the server where it was monitored and managed thoroughly, hence not affecting the devices' performance due to the network monitoring. Features selected for network traffic monitoring were TCP flow features, i.e.,

uploading and downloading bytes; total uploading and downloading packet numbers in session; and HTTP header features, i.e., requested resources' internet host and port number, the request method, request uniform resource identifier (URI), and the user agent. Researchers used the Drebin project's malware apps and benign traffic from the app market using an app crawler. The proposed method achieved higher accuracy as compared to Drebin and other malware scanners.

In [13], the authors introduced a fog network distributed over the smart city which detects and alerts unusual activities and IoT cyberattacks. Data from the IoT devices was collected through the IoT sensors in the IoT layer and computed at the fog layer, and then an alert was generated to notify the cloud security systems. The fog layer strategy is useful for reducing the latency between IoT sensors and the cloud. At the fog layer, the machine learning algorithm, random forest, was trained on the UNSW-NB15 dataset [14] with the selected features such as source and destination IP, destination port, protocol, and duration. In another research [15], researchers presented the SIEM-based IoT-botnet attack detection and mitigation framework. Traffic logs from the IoT devices were collected through the gateway by installing the Splunk SIEM agent on it. This forwarding agent sent the traffic logs after parsing and indexing them to the Splunk Enterprise server. These forwarded logs were analyzed, and if the distributed denial of service (DDoS) attack was identified, the Splunk SIEM server built the connection with the gateway using SSH and automatically added the rules to the firewall IPTABLES to stop the malicious traffic and alert the network administrator about the attack.

In [16], researchers used the agentless technique by using simple network management protocol (SNMP) push and pull requests for log collection from remote devices and storing the logs in the Prometheus database. For log analysis and visualization, the Grafana dashboard was used and in case of anomaly, and the Prometheus default alert generator generated an alert. In [17], researchers used the publicly available dataset CIC-IDS2017, which included clean and DDoS attack traffic in pcap file format. Using CICFlowMeter [18], they extracted features from the pcap-like timestamp, destination, source Ips, and other relevant features to identify the attacks. CSV files, having mentioned attributes, were directly fed to the machine learning model at open-source elastic SIEM. The simple threshold rules were configured for alarming network traffic logs that indicated the DDoS attack. For example, DDoS alarms could be activated by adding the rule that if the number of certain IP connections increases from three times the average of the last hour per minute.

In [19], researchers proposed a system that used a federated, self-learning approach for anomaly detection in IoT devices at a security gateway. For device type identification, authors used the existing approach AuDI [20], which identifies the device type in the local network. For anomaly detection, federated-learning-based models were developed based on specific device types and were trained on locally generated data from the security gateways. Later, these models were aggregated to the global models by IoT security service. The proposed system, when evaluated in a real environment, came out with a higher attack detection rate and no false alarms.

In [21], A SIEM solution is designed and implemented for the security of the smart grid. The proposed SIEM consists of some components for infrastructure monitoring, traffic capturing, machine learning/ deep learning (ML/DL)-based intrusion detection, and smart-grid application-layer monitoring of the visual analytics-based anomaly detection. The researchers used the features of Modbus, DNP3, building automation and control networks (BACnet), MQTT, and other protocols for various attacks such as DoS, SQL injection, and intrusion detection.

Industrial protocols play a significant role in the communication security of smart industrial systems. In a research study analyzing industrial protocols [22], researchers examined DNP3 protocol packets using Wireshark. The findings suggested that there was a control code inside the application layer that showed the relay trip. The rules were written in Snort to generate the alert when there was a trip command from an unknown source.

Normal and anomalous traffic detection has widely been performed with the use of machine and deep learning algorithms. In [23], the authors used deep learning algorithms for the detection of attacks in modern datasets such as UNSW-NB-15 [14] and CICDDoS2019. The authors claimed to use the hybrid deep learning approach for attack detection. First, the autoencoder was used for the important feature selection without human intervention, and then a multilayer perceptron network was utilized for attack prediction. The model resulted in 98% accuracy in DDoS attack detection.

Industrial SIEM solutions are considered the standard for security implementation. However, the relevant literature reported limited research on agentless approaches for industrial IoT security solutions such as Wazuh. Critical analysis of the literature also emphasizes the gap in industrial protocols' feature analysis and their use for rule writing. Utilization of IoT/IIoT communication data to detect and predict attacks can assist in developing the holistic scheme for industrial cyber-physical systems security. To fill the gaps in smart industrial cyber-physical systems security literature, this research work, as shown in Table 1, proposed a security scheme that collects the IIoT network data using the agentless approach from the gateway, converts the raw traffic files to CSV format, and passes the data to a trained machine learning model for attack detection. The predictions of the ML model along with the data packet information are embedded into the JSON log format. The JSON logs are forwarded through the agent of Wazuh installed on the same device having the traffic sniffing scripts and ML models at the gateway level. The logs are received at the Wazuh server end where the decoders are added to extract the features that are further used in rules writing for attack detection and event monitoring. The rules are written against Intrusion detection, DDoS, and man-in-the-middle (MITM) attacks. For event monitoring of industrial IoT devices, protocols based on the specific types of IIoT processes are studied, and important features are extracted to determine the commands the devices are sending or receiving to or from the other devices. Rules are written that match the specific function codes and service types and generate alerts of the event occurring in the industrial control systems. Alerts that are generated in case of any anomaly and the events occurring in the systems are displayed on the Wazuh dashboard. The alerts are stored with the timestamp and the summary can also be accessed and downloaded from the dashboard.

**Table 1.** Comparison of Relevant Approaches for IoT Security Using SIEM.

Papers	Agentless	Technique	SIEM Solution	Network Data/ System Data	Protocol Analyzed	SWaT Case Study
[11]	X	Installing agents on monitored devices	Splunk SIEM	System data	X	X
[15]	✓	Installing the agent on the gateway	Splunk SIEM	Network Data	DNS, TCP, ICMP	X
[16]	✓	R-Pi used for pulling the data	X	Network Data	SNMP	X
[22]	X	N/A	Security Onion	Network Traffic	DNP3	X
Proposed system	✓	Agent installed on the gateway	Wazuh	Network Traffic Monitoring of IoT and IIoT	CIP TCP Modbus DNP3	✓

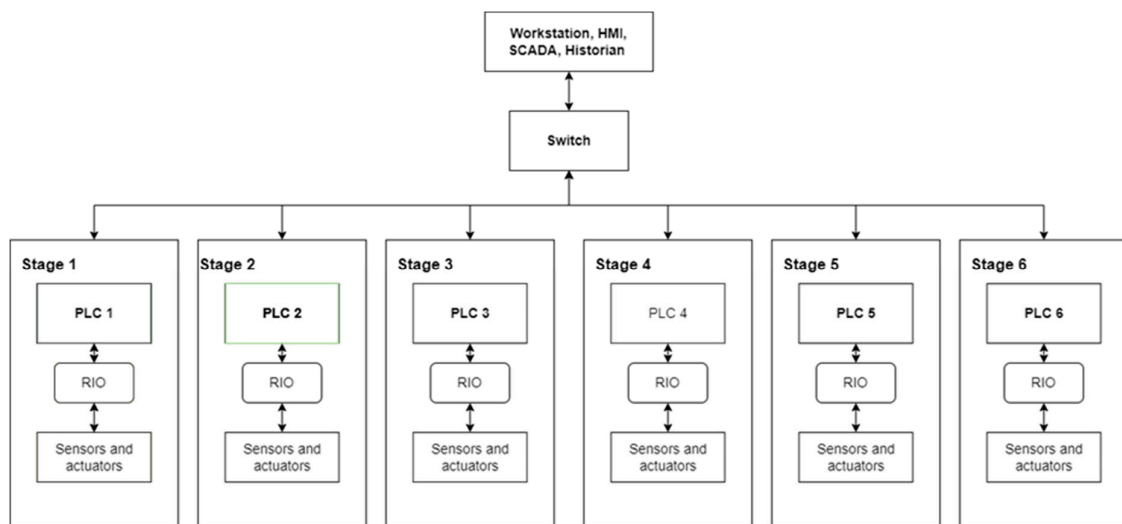
### 3. Experimental Design Components

#### 3.1. The SWaT Case Study

The water supply system is crucial infrastructure that is recently one of the most frequent targets of cyberattacks. The Singapore University of Technology's iTrust Centre for Research in Cyber Security created the secure water treatment testbed (SWaT) in 2015 for cyber security research. Later, updated datasets and details were also uploaded by the

iTrust labs to help researchers in a smart industrial security context. The last update was made on 19 July 2021. The dataset, its details, and the testbed's technical information can be accessed from iTrust labs [24].

SWaT is an operational water treatment testbed that can produce 5 US gallons of filtered water per hour. The testbed is a scale-down setup of large modern water treatment plants common in large cities, with a footprint of about 90 square meters. The control system and overall physical process of SWaT intimately reflect the legitimate setup in the field. Figure 3 shows that there are six stages of the SWaT, numbered P1 to P6. At each stage, there are dual PLCs one of which acts as the primary controller, if the primary PLC fails, the secondary PLC works as a backup of the event. In general, the testbed uses a dispersed control strategy in regular operations, with local PLCs controlling each stage of the process separately. The PLCs are connected in the layer 1 network and communicate constantly to receive the state information from other processes that the local device needs for some of the processing steps. The operator can control all the actuators in the testbed manually, instead of using the automatic distributed control mode, by using the human-machine interface (HMI) and supervisory control and data acquisition system (SCADA) systems.



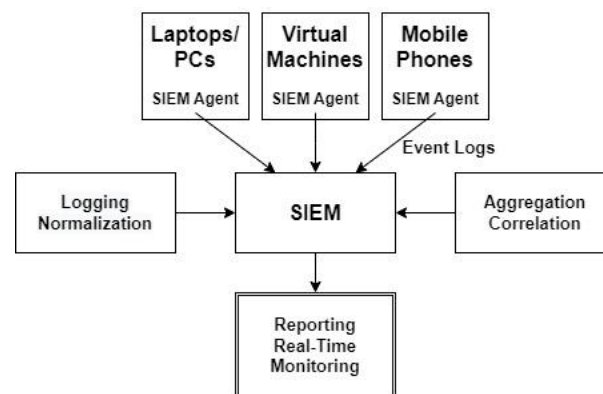
**Figure 3.** SWaT Network Architecture.

The primary PLC collects data from the sensors and manages actuators using the Remote I/O (RIO) unit during each stage of the process, such as pumps and valves. Remote I/O connected between the PLCs and the sensor/actuators for the transfer of commands and data to and from the PLCs and field devices. Water can flow into or out of a tank, for instance, by turning the pumps on or closing a valve. The PLCs monitor the status of the devices and decide when the pump should be turned on or off by updating the actuator values. To monitor the chemical traits of water flowing through all six stages, additional sensors are available.

### 3.2. SIEM Solutions

Since IoT devices are now being used in every field from homes and small-scale retail shops to large-scale industries, the vulnerabilities within them are exploited by adversaries to steal information or for ransom. Gartner in 2005 introduced SIEM (security information and event management) for endpoint monitoring. SIEM is the combination of SIM (security information management) and SEM (security event management), which are two separate systems for event storage, analysis, and reporting (SIM) and real-time collection of events (SEM) [25]. Generally, SIEM is a security tool that aids companies in identifying potential security vulnerabilities and threats before they manage to interfere with and damage business operations. For security and compliance management use

cases, it surfaces user behavior anomalies and employs artificial intelligence to automate many of the manual operations related to threat identification and incident response. It has become a mainstay in contemporary security operation centers (SOCs) [26]. Security information and event management (SIEM) tools are typically used for cyber-physical systems security. They collect the event logs from the devices and perform diverse actions to ensure the security of the connected systems. The main functions of SIEM solutions are event logging, normalization, aggregation, and event correlation. In logging, the SIEM solution stores the data collected and forwarded by the agents on the devices. These data collected from different types of devices are then converted to the common format so that the data structure operations can be performed on them. This process is called normalization. After normalization, all the redundant values from the data are eliminated (aggregation). Then, the processed logs are correlated for the detection of any suspicious behavior and unknown pattern. After performing all the functions on the collected data, SIEM solutions generate alerts in case of any abnormal activity performed on the devices. A basic SIEM architecture is shown in Figure 4.



**Figure 4.** SIEM Architecture.

SIEM is used to gather event data from monitored devices throughout the network of a company or home. IT and security teams can automatically manage their network's event log and network flow data in one centralized location thanks to the real-time collection, storage, and analysis of logs and flow data from users, applications, assets, cloud environments, and networks. To compare their internal security data with known threat signatures and profiles, several SIEM solutions also integrate with third-party threat intelligence feeds. Teams can stop or recognize novel attacks through integration with real-time threat sources. Any SIEM solution must provide event correlation which uses advanced analytics to quickly find and eliminate possible threats to enterprise security by identifying and comprehending complex data patterns. The manual operations associated with the in-depth analysis of security events are offloaded by SIEM systems, which considerably reduces the mean time to detect (MTTD) and mean time to respond (MTTR) for IT security teams.

SIEM systems can recognize all entities in the IT environment since they provide centralized management of on-premises and cloud-based infrastructure. By categorizing strange activity as it is discovered on the network, SIEM technology can monitor security incidents across all connected users, devices, and applications. Administrators can be quickly warned and take appropriate action to mitigate it using customizable, established correlation criteria before it develops into more security risks. Much research is conducted to secure IoT devices using different tools and techniques to detect and prevent anomalies within the IoT network.

### 3.3. Wazuh SIEM

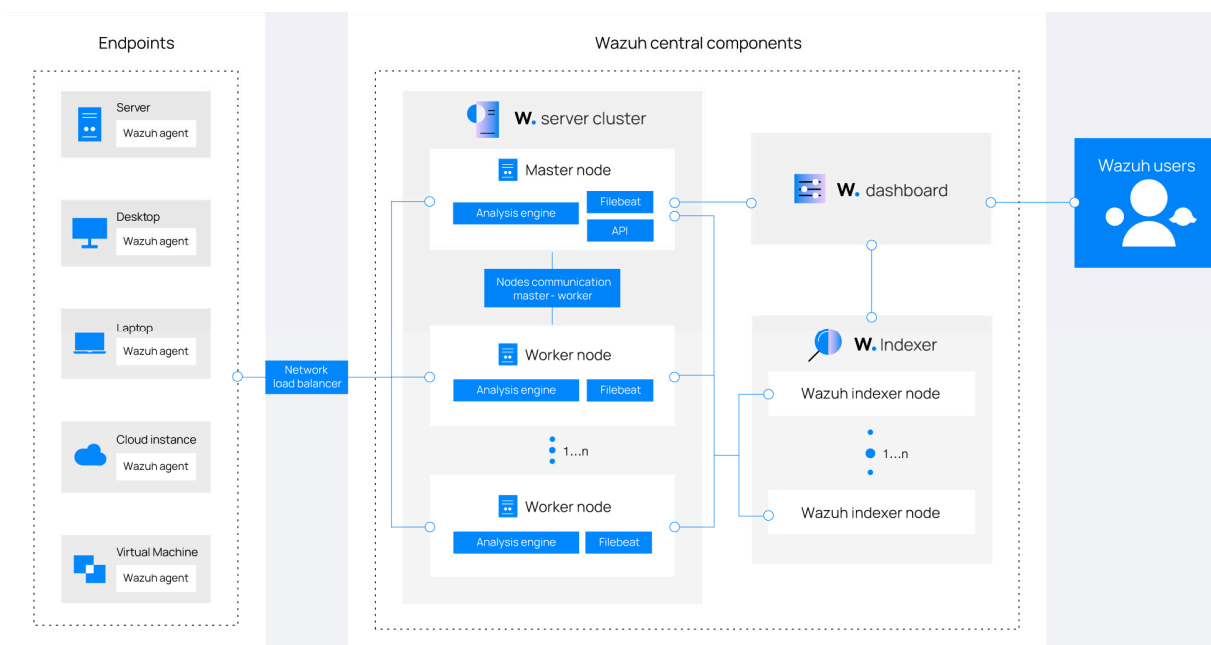
Wazuh is an open-source SIEM solution, and its architecture is mainly based on the agents that run on the monitored hosts and send security information to a centralized



SIEM server. Firewalls, switches, routers, and access points that do not require agents are supported and can actively provide log data through Syslog, SSH, or their application programming interfaces (APIs). The Wazuh indexer indexes and stores the findings from the central server's analysis and decoding of the incoming data. Wazuh indexer is a full-text search, highly scalable engine. The indexer is responsible for indexing the alerts coming from the server and storing them. The indexer provides analytics and near real-time search capabilities. In this research, a single-node indexer is used for deployment in a larger IIoT environment; the multi-node cluster can be installed for higher availability. The Wazuh indexer stores the data in a JSON document in the form of a key-value pair and then correlates the data key fields and value fields. So, in the Wazuh server, the values can be of any datatype, i.e., Boolean, integer, string, and time format. This main component of Wazuh ensures redundancy and increases query capacity by distributing the JSON documents into the shards. Shards are the containers that would be further distributed to other nodes in case of multi-node cluster deployment of Wazuh.

The server is the component of Wazuh that receives the logs from the agent. It then applies the rules written in it and in case of any anomaly; it triggers the alerts. The server also manages the remote agents and their status. Different components of the Wazuh server perform different tasks, such as the Agent enrollment service: agents are enrolled by generating the authentication keys and sharing them with the agent using the TLS/SSL connection Agent connection service. This part authenticates the agent's ID and authentication keys shared by the enrollment component, the Analysis engine: this is an important component required to make the logs of the raw pcap network traffic. This component uses the decoders to recognize the type of information that is received from the agents. The patterns extracted from the decoders are matched with the rules written in the Wazuh. All the alerts and analytics of events can be seen from the Wazuh dashboard.

Wazuh has the agents that are to be deployed on the monitored hosts. The hosts should be configured with the server using the IP and authentication keys. The paths for the Windows, Linux, and MAC operating systems are already defined in the configuration file of the agents from where they gather the system data. The agents send the data to the server on the same network using the TCP protocol on the 1514 port. Wazuh SIEM can also monitor the devices having specific operating systems or having the ability to send the data through an SSH connection with the server. Previously, efforts have been made on developing mechanisms to secure IoT/IIoT devices to overcome their power, computation, and storage limitations. Limited research and development have been performed on securing the IoT/IIoT devices using the SIEM solution. Some researchers used the SIEM solution for securing the IoT/IIoT devices but only implemented the security solution for DoS attack detection, while others either installed the SIEM agents on the devices or used SNMP pulling techniques or built some type of connection between the IoT/IIoT devices and the SIEM server for data collection. A complete Wazuh agent server architecture diagram is shown in Figure 5.



**Figure 5.** Wazuh Architecture Diagram [27] Reprinted/adapted with permission from Ref. [27]. Copyright 2023, Wazuh.

## 4. Methodology

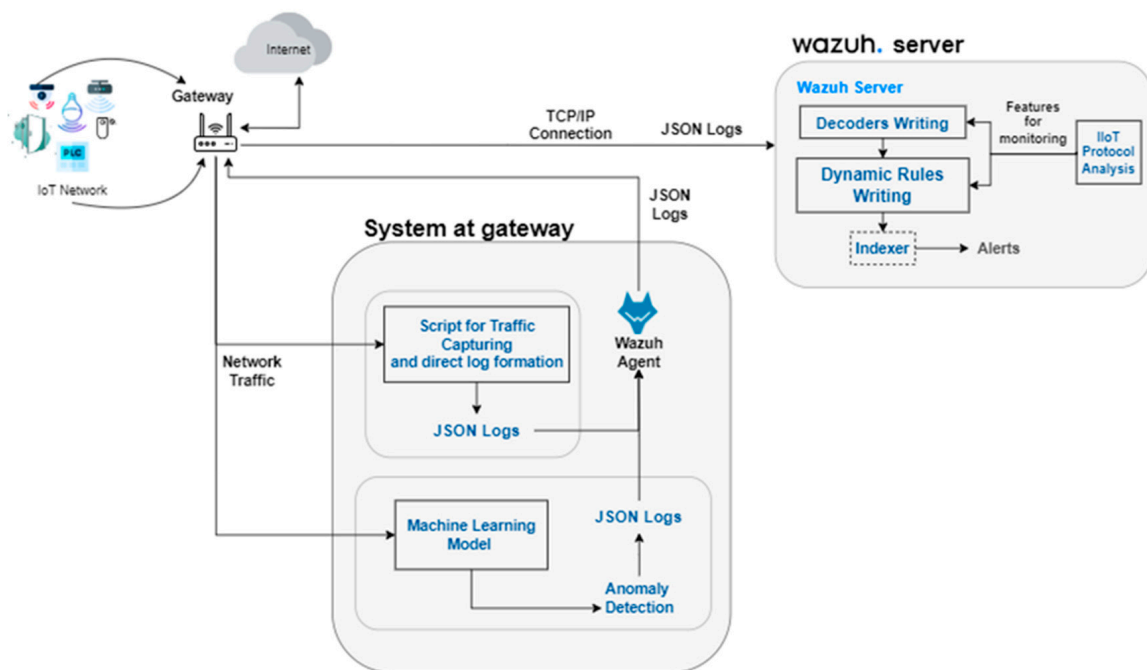
### 4.1. IIoT Dataset

The suggested technique used the agentless approach for the network traffic analysis of the IoT devices along with the Wazuh SIEM solution. The security scheme proposed the collection of IIoT traffic logs from the default gateway and directing them to the SIEM solution for correlation and monitoring without installing any software or running the script on the field IIoT devices such as sensors, actuators, and PLCs. SWaT and CICDDoS2019 datasets were used for real-time traffic monitoring and attack detection using ML models, respectively. In CICDDoS2019 dataset, different types of attacks were performed on the systems by a third party. Some of the attacks they performed on the devices were PortMap, NetBIOS, LDAP, and SYN [23]. In both publicly available datasets, the traffic was collected from the gateway.

For the implementation of this security scheme in industrial settings, the first step would be to gather the IIoT devices' network data from the gateway by collecting the IIoT network traffic through live capture using Wireshark or from the Python library Pyshark using the following script:

```
capture = pyshark.LiveCapture (interface = "en0", output_file = file)
```

In this research, real-time traffic monitoring and ML-based attack detection are the two processes carried out at the gateway level. The main goal of this research was to detect malicious activity and monitor various types of IIoT devices. The experimental testbed is shown in the proposed research scheme diagram in Figure 6.



**Figure 6.** The Proposed Security Scheme.

#### 4.2. Real-Time Traffic Monitoring

For real-time traffic monitoring, the network traffic was directly sent to the server by converting it to the JSON log and sending it to the server. The following script was used for the direct conversion of the raw IoT network packets into JSON format.

```
os.system('tshark -T json -i 4 out.json')
```

As the Wazuh agent can only read the flat log files, the JSON logs in the out.json files were converted to the inline JSON format. In this research, the logs converted from raw pcap to the JSON format had one packet in multiple lines which without using the XPath filters could not be read by the agent [27]. Using a Python script, the multiline JSON logs were converted to inline JSON logs, i.e., one log per line.

#### 4.3. Attack Detection Automation at the Gateway

The next step was to monitor the traffic using machine learning models for the detection of state-of-the-art attacks in IIoT devices. The machine learning models were trained on the CICDDOS2019 dataset which was available in CSV format. In this dataset, the researchers collected the data into pcap files from the network gateway by using CICFlowMeter, the traffic was converted into the CSV having the network traffic flow features. The pre-processing of the dataset was performed before training the models. The field's source and destination IP addresses, timestamps, and flowID were to be encoded before feeding the dataset to the model. Using the information gain technique, important features that correlated with each other were identified and the resultant (encoded and compacted features) dataset was set ready to train the models. The dataset was split as 70% for training the model and 30% for testing the models. The training data files were used to train the DT, RF, and KNN machine-learning classification models, widely used for classification problems. The dataset had different class labels, i.e., benign and the types of DDoS attacks such as UDP flooding, SYN flooding, NetBIOS, and LDAP, to name a few. The scripts for model training and testing were written in Python using the PyCharm community version. The accuracy of all the models to predict the attacks were calculated on the incorporated dataset and the best predictive model with higher performance measures was used for

the prediction of attack and benign traffic logs. The following sections are on the model's description and its results on the CICDDoS2019 dataset.

#### 4.3.1. K-Nearest Neighbor (KNN)

K-nearest neighbor is a supervised learning algorithm that works by putting the new data into the category that is most likely to the existing category data by assuming the similarity between the new and the old data. Although the KNN approach is most frequently employed for classification problems, it can also be utilized for regression problems. The algorithm is also called the lazy learner algorithm due to the reason that at the training time, it only stores a dataset and at the time of testing, it compares the new data with the existing one and categorizes it with the most similar data. The distance of the new instance, measured from the neighbors is calculated using the Euclidean distance formula given by:

$$\text{Euclidean distance between instance } A_1 \text{ and } A_2 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \quad (1)$$

#### 4.3.2. Decision Tree Model

A decision tree is a supervised algorithm mostly used for classification problems. The model is a tree-structured classifier where the dataset features are represented as the internal nodes, the branches represent the decisions made, and finally, the outcomes are represented by the leaves. In this model, to make the decision tree, the classification and regression tree (CART) algorithm is used. While testing the data, based on the trained data, the new instance is compared to the nodes and branches, and according to the decision of the model, it jumps to the other node and repeats the process, and the final decision is shown at the end of the tree, called leaves. The below algorithm shows how the decision trees are made:

1. The source set is split into the subsets based on attribute values tests.
2. The above process is repeated on the subsets recursively.
3. The recursive splitting process is called recursive partitioning.
4. Recursive partitioning process is carried out until further splitting does not yields into the performance betterment.

The instances using decision trees are classified as follows:

1. The instance is traversed through the tree from the root node to the leaf nodes.
2. The instance is tested on the root node and based on the attribute value it jumps to the specific node after testing.
3. The process is repeated on each level of node and hence an instance is classified.

#### 4.3.3. Random Forest Classification Model

As the name depicts, the model develops a forest of decision trees for class prediction. Random forest is the supervised algorithm. The dataset is first divided into subsets randomly. The Gini index of the splits is calculated to find out the impurity of the splits. The Gini index is calculated using the formula.

$$\text{Gini Index} = 1 - \sum_{i=1}^n (P_i)^2 \quad (2)$$

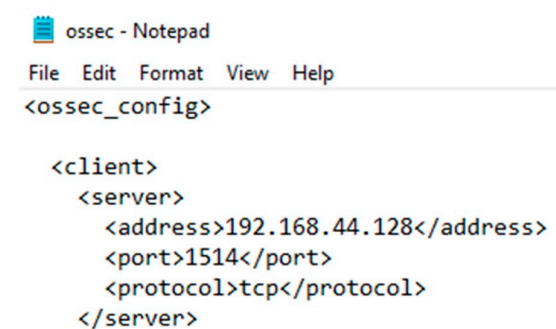
The impurity of the split can also be measured by calculating the entropy.

$$E(S) = -p_{(+)} \log p_{(+)} - p_{(-)} \log p_{(-)} \quad (3)$$

The split with the lowest Gini index or lowest entropy is used for further splitting and making the decision trees. After the row and column sampling, decision trees are made for each of the subsets. The decision trees work independently and output their prediction. The outcome of the final random forest model is what most decision trees predict.

#### 4.4. Gateway Agent Configuration with Wazuh Server

In this research, the gateway agent is configured with the server through the Wazuh server's IP setting in the agent's configuration file. The agent requests for the authentication key from the server on the server's IP and the server responds and the connection is established for monitoring. The Wazuh server receives the logs from the agent on port 1514 which is also defined in the configuration file `ossec.config` of the agent, as shown in Figure 7.



```
ossec - Notepad
File Edit Format View Help
<ossec_config>

<client>
  <server>
    <address>192.168.44.128</address>
    <port>1514</port>
    <protocol>tcp</protocol>
  </server>
```

Figure 7. Server IP configuration file.

Real-time event monitoring from the network data is performed by analyzing the application layer protocols that are mostly used in industrial control systems. Common Industrial Protocol (CIP), DNP3, and Modbus are examples of industrial protocols. Errors in the devices or the operations performed by these protocols can be detected from the industrial protocol fields that are used in the system. The analysis and description of the features that can be used in the rules for event monitoring are described in the following section.

#### 4.5. Industrial Protocols Analysis

##### 4.5.1. Common Industrial Protocol (CIP) Analysis

CIP or Common Industrial Protocol uses the producer-consumer communication model to handle general purpose network connections, network services such as file transfer, and automation operations, including analog and digital input/output devices, HMI, movement control, and position response. CIP packets have the attributes, service codes, and connections that represent data, commands, and relationships between the data and services, respectively. For a wide range of industrial automation applications, including control, safety, energy, synchronization and motion, information, and network administration, CIP comprises a comprehensive suite of messages and services. Users can connect these applications to the Internet and business-class Ethernet networks using CIP. CIP offers consumers a single communication architecture across the industrial enterprise and is supported by hundreds of vendors globally. It also offers a media-independent, expandable, and upgradeable communication architecture while protecting their current automation investments and enabling users to make use of the various benefits of open networks now.

CIP has two types of messages: explicit and implicit messages. Explicit messages contain the attributes, service codes, and paths that are used to direct the devices on what action to perform. The implicit messages are used for I/O data transportation. There are no addresses, or service codes and the device already know what to do with the data implied by the connection ID that is decided at the time of connection. Event monitoring in the industrial networks that use the CIP protocol can be monitored using the CIP attributes. The Secure Water Treatment (SWaT) setup also uses the CIP protocol. After analyzing the network traffic of SWaT, service codes, request path, and success status were recognized as the important features for event monitoring of SWaT. Important fields of CIP that were used

in the rule for threat intelligence at the Wazuh server end are `service_code`, `request_path`, and `success_status`.

#### 4.5.2. Modbus Protocol Analysis

Modbus is commonly used in remote terminal units (RTUs). The TCP listening port is 502 and is typically communicating through RS485 or RS232. The Modbus application layer consists of a protocol data unit (PDU) and before PDU its MBAP Header (Modbus application header). Modbus protocol has a transaction that uniquely identifies each request's function code that specifies the type of action to perform, the data bytes that contain the information about the start register, and several registers to read. The transaction identifier can be used to check the number of transactions made by the device in a specific time unit. Similarly, the unit identifier can be used to check how many times an RTU, or any other communication device connects with the same PLC or any other slave device. The main resource is the function code by which we can identify whether the PLC is getting an error or not, how many types of work a device performs in a day, and if the specific device is given the assigned task or not. For a read operation, numbers 01 to 04 are used, and for a write operation, 05 to 10 codes are used, which are for reading and writing the coils, discrete inputs, multiple holding registers, and input register values. Rules can be written for monitoring the network using the Modbus protocol and for diagnostic purposes based on the above information.

#### 4.5.3. DNP3 Protocol Analysis

DNP3 is a three-layer protocol following the standard of the Ethernet for plant automation (EPA) two-way communication. It is rolled up into an application layer, encapsulated within the TCP/IP or UDP layer. Sometimes other layers are also added, i.e., pseudo transport layer. DNP3 protocol is used between the master and distributed remote units called outstations. The master works as an interface between the human network manager and the monitoring systems, while the remote unit is between the master station and the physical equipment.

In DNP3 packets, a feature 'length of Datalink layer' checks the information of application data regarding each function of a device. The function code of the application layer identifies the action to perform. There is a field trip control that is set in case of tripping in the devices. The protocols such as DNP3 are specialized and function codes are already configured; the rule can look for the function codes and generate the alert according to the meaning of that function code in the system. For example, if a `Function_code==15` alert is generated with the message 'unsolicited', the alarm is disabled.

#### 4.5.4. SWaT Network Monitoring

For SWaT process monitoring, the features of the common industrial protocol are deeply analyzed, extracted, and compared in the rules for the event alert generation. The CIP protocol has the service codes in the packets that have the fixed value for read (0x4c) and write (0x4d) commands. The example packets of the read operation are stated in Figures 8 and 9. PLC 6 (192.168.1.60) requests the value of level sensor LIT101 from PLC 1 (192.168.1.10), which is equipped in the first stage of the SWaT system. PLC 1 responds back with success status to PLC 6 with the sensors' data.

No.	DNS Time	Source	Destination	Protocol	Length	Info
243	0.009290	192.168.1.60	192.168.1.10	CIP	116	'HMI_LIT101' - Servi...
290	0.010313	192.168.1.10	192.168.1.60	CIP	144	Success: 'HMI_LIT101...
294	0.010587	192.168.1.10	192.168.1.30	CIP	116	'HMI_FIT301' - Servi...
296	0.010818	192.168.1.10	192.168.1.30	CIP	124	'HMI_PLANT' - Servic...
300	0.010823	192.168.1.10	192.168.1.20	CIP	124	'HMI_PLANT' - Servic...

```

> Frame 243: 116 bytes on wire (928 bits), 116 bytes captured (928 bits)
> Ethernet II, Src: Rockwell_c7:fa:2d (00:1d:9c:c7:fa:2d), Dst: Rockwell_c8:bd:e7 (00:1d:9c:c8:bd:e7)
> Internet Protocol Version 4, Src: 192.168.1.60, Dst: 192.168.1.10
> Transmission Control Protocol, Src Port: 59072, Dst Port: 44818, Seq: 1, Ack: 1, Len: 62
> EtherNet/IP (Industrial Protocol), Session: 0x00260180, Send Unit Data, Connection ID: 0xFFE146C9
  > Common Industrial Protocol
    > Service: Unknown Service (0x4c) (Request)
      0... .... = Request/Response: Request (0x0)
      .100 1100 = Service: Unknown (0x4c)
      Request Path Size: 6 words
    > Request Path: HMI_LIT101
    
```

Figure 8. Request packet in CIP protocol.

No.	DNS Time	Source	Destination	Protocol	Length	Info
243	0.009290	192.168.1.60	192.168.1.10	CIP	116	'HMI_LIT101' - Servi...
290	0.010313	192.168.1.10	192.168.1.60	CIP	144	Success: 'HMI_LIT101...
294	0.010587	192.168.1.10	192.168.1.30	CIP	116	'HMI_FIT301' - Servi...
296	0.010818	192.168.1.10	192.168.1.30	CIP	124	'HMI_PLANT' - Servic...
300	0.010823	192.168.1.10	192.168.1.20	CIP	124	'HMI_PLANT' - Servic...

```

> Frame 290: 144 bytes on wire (1152 bits), 144 bytes captured (1152 bits)
> Ethernet II, Src: Rockwell_c8:bd:e7 (00:1d:9c:c8:bd:e7), Dst: Rockwell_c7:fa:2d (00:1d:9c:c7:fa:2d)
> Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.60
> Transmission Control Protocol, Src Port: 44818, Dst Port: 59072, Seq: 1, Ack: 63, Len: 90
> EtherNet/IP (Industrial Protocol), Session: 0x00260180, Send Unit Data, Connection ID: 0xFF9B726E
  > Common Industrial Protocol
    > Service: Unknown Service (0x4c) (Response)
      1... .... = Request/Response: Response (0x1)
      .100 1100 = Service: Unknown (0x4c)
    > Status: Success:
      [Request Path Size: 6 words]
    > [Request Path: HMI_LIT101]
    > CIP Class Generic
    
```

Figure 9. Response Packet in CIP.

The connection between all the devices, while analyzing the SWaT network traffic, is summarized in Appendix A. As discussed above, the SWaT has six stages of water purification. All six stages have their own local PLCs, sensors, and actuators. One-stage sensors and actuators send the values to the PLCs, and the PLCs communicate with each other to exchange the sensor's data and control the whole process. The common industrial protocol is used in the system for communication that has implicit and explicit messages in it. Implicit messages are used to send control I/O data where the PLCs send these messages to the sensors or actuators when they must read or write. The explicit messages are important here for event monitoring. Through explicit messages, the PLC sends the request to read the value of the specific sensor and the other PLC responds with the value of the sensor. Later, the values are passed to the program written to the PLCs and after making the decisions the control data are sent to the local devices. All the communication between the PLCs and other field devices are summarized in Appendix A. By using this information in the SWaT network, the rules are written in Wazuh for event monitoring.

#### 4.6. Decoder Writing in Wazuh

On the server side, the decoder is written and added which is applied to the logs received by the Wazuh server to extract the important features for the rules to monitor or detect the events in the IIoT network. The decoder extracts the transport layer features, as shown in Figure 10, from the logs such as ip.src and ip.dst, MAC addresses, and some TCP fields, while for testing on the SWaT, common industrial protocol fields are utilized which are further used in the rules to generate the alerts.

```

_source.layers.ip.ip.dst_host: '192.168.1.10'
_source.layers.eth.eth.src: '00:1d:9c:c8:f4:b9'
_source.layers.ip.ip.src_host: '192.168.1.50'
_source.layers.eth.eth.dst: '00:1d:9c:c8:bd:e7'
_source.layers.tcp.tcp.analysis.tcp.analysis.bytes_in_flight: '108'
_source.layers.cip.cip.service_tree.cip.sc: '0x4c'
_requestPath: 'HMI_LIT101'

```

**Figure 10.** Features Extracted from the Decoder.

As mentioned earlier, the decoders were written and added in this research for extracting the fields from the JSON logs.

The decoder decodes and extracts the ip.src and ip.dst field values from the logs which will further be used in the rules to generate the alerts (Figure 11). The explanation of keywords used in the decoders is given in the preceding sections.

```

20 - status - event status (success, failure, etc)
21 - extra_data - Any extra data
22 -->
23 <decoder name="json">
24 <prematch>^{\s*}</prematch>
25 </decoder>
26 <decoder name="json_child">
27 <parent>json</parent>
28 <regex type="pcre2">"ip.src": "([^\s]+)"</regex>
29 <order>srcip</order>
30 </decoder>
31 <decoder name="json_child">
32 <parent>json</parent>
33 <regex type="pcre2">"ip.dst": "([^\s]+)"</regex>
34 <order>dstip</order>
35 </decoder>
36 <decoder name="json_child">
37 <parent>json</parent>
38 <plugin_decoder>JSON_Decoder</plugin_decoder>
39 </decoder>
..

```

**Figure 11.** Decoder for Custom Logs.

- Keyword 'prematch'

The prematch keyword is used to match some string that is common in the logs that are intended to be decoded by this decoder. If the string is matched, then the current decoder will be used, and the other decoders will not be searched.

- Keyword 'regex'

Regex is the abbreviation for regular expression used by the decoders to find patterns or words in the rules. Only the fields that are in the parenthesis are extracted by the decoders.

- Keyword 'order'

Order label is mandatory to define with the regex keyword. The order keyword defines the field name in which the regex pattern is received.

- Keyword 'parent'

This keyword is used to link the child decoders with a parent decoder. A parent decoder may have one or more children, but the child decoder cannot be a parent to another decoder. There are many more keywords that can be used in decoders, and rules can be found from [27].

- 'pcre2'



Pcre2: Perl Compatible Regular Expression used or logs interpretation. There are many quantifiers used in pcr2; however, some that are used in our decoders are given below in Table 2.

**Table 2.** Decoder Quantifiers.

*	To match 0 or more times
+	To match 1 or more times

- Special Characters

Table 3 shows some special characters used in decoders. More characters and quantifiers with description can be studied [27]. The decoder can be tested using the option “Decoders Test” by giving an example log. The test results are shown in Figure 12.

**Table 3.** Wazuh Special Characters used in Decoders.

^	Specifies beginning of the text
&	End of the text
\s	Specifies Space
!	To negate the expression

```

**Phase 1: Completed pre-decoding.
  full event: '{"_index": "packets-2021-11-29", "_type": "doc", "_score": n

**Phase 2: Completed decoding.
  name: 'json'
  _index: 'packets-2021-11-29'
  _score: 'null'
  _source.layers.eth.eth.dst: '40:5d:82:35:14:c8'
  _source.layers.eth.eth.dst_tree.eth.addr: '40:5d:82:35:14:c8'
  _source.layers.eth.eth.dst_tree.eth.addr_resolved: 'Netgear_35:14:c8'
  _source.layers.eth.eth.dst_tree.eth.dst.ig: '0'
  _source.layers.eth.eth.dst_tree.eth.dst.lg: '0'
  _source.layers.eth.eth.dst_tree.eth.dst.oui: '4218242'
  _source.layers.eth.eth.dst_tree.eth.dst.oui_resolved: 'Netgear'
  _source.layers.eth.eth.dst_tree.eth.dst_resolved: 'Netgear_35:14:c8'
  _source.layers.eth.eth.dst_tree.eth.ig: '0'

```

**Figure 12.** Decoder Testing.

#### 4.7. Rule Writing and SWaT Event Monitoring

For this research, the industrial map was already available indicating where the security solution was to be deployed into the network. The devices' IPs or MAC were manually configured in the rules file. Rules were written for the intrusion detection and the alerts generation test was carried out by sending the requests from the unknown IP to the network whose information was not configured in the SIEM server. The DoS attack was detected by the rule in which the timeframe and frequency of packets were defined, and the alert was generated when the logs were received from the same IP that was unknown to the system within a specific timeframe.

All the information in Appendix A was gathered by analyzing the network traffic of SWaT. This information was used to write the rule to monitor the network. The service code fields, and CIP request path fields were extracted and used to generate the alerts whenever the event occurred as shown in Figure 13.

```

<rule id="100005" level="5">
  <field name="_source.layers.cip.cip.service_tree.cip.sc">^0x4d$</field>
  <description>$(requestPath) Write operation request from $(dstip)</description>
</rule>

<rule id="100006" level="5">
  <if_matched_sid>100005</if_matched_sid>
  <match>Success</match>
  <description>$(requestPath)Value written on $(dstip) </description>
</rule>

```

Figure 13. Rules to Monitor SWaT Network Communications.

The tag names (request path) were also defined and used to predict which sensor or actuator's value was requested or transmitted in the monitored packet. The alerts were generated whenever the data was requested or responded to from or to the programmable logic controller.

#### 4.8. DDoS Attack Dynamic Rule Writing

JRip algorithm is used for the dynamic rule writing for DDoS attack detection. JRip works by learning the patterns from the training dataset and specifying the threshold for each class given in the training dataset. The deep understandings of how JRip works can be gained from [28]. By feeding the CICDDoS2019 dataset into the Weka software using the JRip algorithm, the rules were written to detect different types of DDoS attacks found in the dataset. The thresholds for the fields were noted from the output of the JRip algorithm. The fields needed to compare from the traffic were extracted from the decoder and the rule was written into the WAZUH for the attack's detection to complement the machine learning model attack prediction. Enhanced security is achieved by this approach.

The thresholds can be changed periodically for performance maintenance on future attacks. A generalized DDoS attack can be detected by defining the timeframe and the packet count in a rule as shown in Figure 14. The following Figure 15 is an example of such a rule.

```

( ACK Flag Count >= 1) and (Init_Win_bytes_forward <= 5840) => Label=Syn (34984.0/0.0)
( SYN Flag Count >= 1) and ( Flow Duration <= 1) and ( Source Port <= 38996) and ( Source Port >= 3127) => Label=Syn (9.0/0.0)
( Min Packet Length >= 378) => Label=Syn (5.0/1.0)
=> Label=BENIGN (36744.0/1.0)

```

Figure 14. Dynamic Rules for DDoS Types Detection.

```

1 <group name="local,syslog,sshd,">
2 <rule id="100001" level="2">
3   <decoded_as>json</decoded_as>
4   <srcip>!192.168.137.249</srcip>
5   <description>A new IP found</description>
6 </rule>
7
8 <rule id="100002" level="3" frequency="5" timeframe="10">
9   <if_matched_sid>100001</if_matched_sid>
10  <same_srcip />
11 <description>Dos Attack Detected $(srcip)</description>
12 </rule>
13 </group>

```

Figure 15. DDoS Attack Detection Rule.

## 5. Results and Discussions

For this research, the raw network traffic was converted to JSON logs as the Wazuh SIEM accepts the logs in the specific formats, i.e., Syslog and JSON formats. For protocol-based network monitoring, SWaT network traffic was converted into the JSON format and was sent to the server. Concurrently, machine learning techniques were incorporated at the

same gateway level for attack detection in IIoT devices. Classification models KNN, DT, and RF were used for measuring the prediction accuracy on the CICDDoS2019 dataset.

*ML Models Parameters and Performance Measures*

The parameters used for all the KNN, DT, and RF models are detailed in Table 4.

**Table 4.** Parameters for ML Models.

Models	Parameters
KNN	No. of neighbors = 7
DT	Maximum depth = 10, minimum sample split = 2
RF	Number of features = 15, number of decision trees = 15

The models were trained using the parameters as shown in the Table 4, and the evaluation measures were examined. RF model showed higher performance measures as depicted in Table 5. Hence, it was used for the prediction of attacks for this research.

**Table 5.** Evaluation of ML Models.

Models	Accuracy	Precision	F1-Score	Recall
K-Nearest Neighbor	99.96%	0.79	0.83	0.88
Decision Tree	99.98%	0.94	0.93	0.94
Random Forest	99.99%	0.94	0.94	0.94

The output of the machine learning module was also written into the JSON file format with the source and destination IP addresses for the proper information on the attack source and attacked devices. The only agent at the gateway level forwarded the JSON logs to the server by making the TCP connection with the server on port 1514. The decoders, and scenario-based and dynamic rules were written for network logs in the Wazuh server and tested on the CICDDoS2019 and SWaT dataset. Wazuh applied the rules and generated the alerts as shown in Figure 16.

Time	Source	Destination	Alert	Count	Severity
Oct 11, 2022 @ 07:56:48.686	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002
Oct 11, 2022 @ 07:56:40.677	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002
Oct 11, 2022 @ 07:56:32.592	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002
Oct 11, 2022 @ 07:56:26.559	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002
Oct 11, 2022 @ 07:56:18.487	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002
Oct 11, 2022 @ 07:56:10.434	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002
Oct 11, 2022 @ 07:56:00.450	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002
Oct 11, 2022 @ 07:55:52.373	001	DESKTOP-LTCGFK8	Dos Attack Detected 192.168.137.5	3	100002

**Figure 16.** DoS Attack Alerts in Wazuh.

Other custom rules written in this research were evaluated using the SWaT dataset. An unknown IP, not configured in the server, sent the traffic and the rule written for intrusion detection was applied and fired the alerts as shown in Figure 17.

The screenshot shows the Wazuh Security Alerts interface. The header includes the Wazuh logo and navigation links for 'Modules' and 'Security events'. The main content area is titled 'Security Alerts' and contains a table with the following data:

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
> Oct 15, 2022 @ 02:37:21.585	001	DESKTOP-LTCGFK8			A new IP 192.168.1.20 found	5	100001
> Oct 15, 2022 @ 02:37:19.651	001	DESKTOP-LTCGFK8			A new IP 192.168.1.20 found	5	100001
> Oct 15, 2022 @ 02:37:19.605	001	DESKTOP-LTCGFK8			A new IP 192.168.1.20 found	5	100001

Figure 17. Intrusion Alerts.

SWaT events were monitored by using the CIP protocol fields that appeared in the logs. Complete detail of the CIP protocol fields was already discussed in the Section 4. The alerts, as shown in the Figure 18, were fired when any PLC in the SWaT system requested or responded to the sensors or actuators’ values.

The screenshot shows the Wazuh Security events interface. The header includes the Wazuh logo and navigation links for 'Modules' and 'Security events'. The main content area contains a table with the following data:

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
> Oct 15, 2022 @ 02:59:22.310	001	DESKTOP-LTCGFK8			HMI_PLANT_AUTO Write operation request from 192.168.1.60	5	100005
> Oct 15, 2022 @ 02:59:22.310	001	DESKTOP-LTCGFK8			Read request from 192.168.1.10	5	100003
> Oct 15, 2022 @ 02:59:22.184	001	DESKTOP-LTCGFK8			HMI_PLANT Write operation request from 192.168.1.40	5	100005
> Oct 15, 2022 @ 02:59:22.077	001	DESKTOP-LTCGFK8			HMI_PLANT_RESET Write operation request from 192.168.1.50	5	100005

Figure 18. SWaT Event Monitoring Rules.

The Wazuh indexer performed analytics functions on the logs, correlated them, and generated the summary graphs on the dashboard according to the level of alerts, as shown in Figure 19.

The alert summary can be downloaded from the server where all the alerts are stored. Figure 20 is the example of such summary generated at the time of model testing.

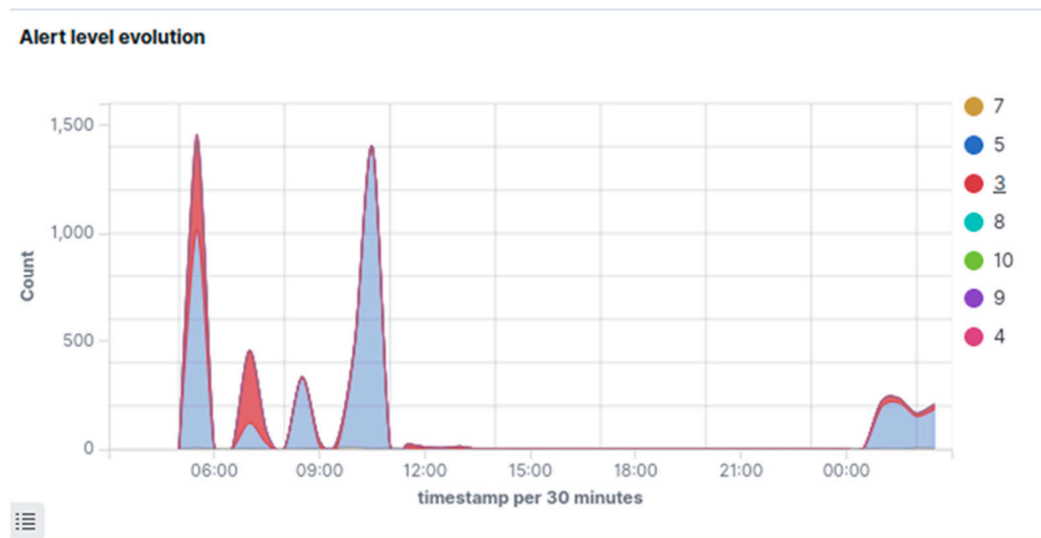


Figure 19. Graphical Representation of Alerts in Wazuh.

### Alerts summary

Rule ID	Description	Level	Count
100004	Successful read operation	5	1317
60106	Windows logon success.	3	418
100004	Read values from 192.168.1.10	5	268
751	Registry Value Entry Deleted.	5	263
752	Registry Value Entry Added to the System	5	258
100004	Read values from 192.168.1.20	5	148
100004	Read values from 192.168.1.40	5	148
100004	Read values from 192.168.1.30	5	136
750	Registry Value Integrity Checksum Changed	5	122
594	Registry Key Integrity Checksum Changed	5	119
100001	A new IP 192.168.1.20 found	5	117
61102	Windows System error event	5	113
100003	successful operation	5	108
100003	Write operation from 192.168.1.10	5	105

Figure 20. Alerts Summary.

### 6. Conclusions and Future Work

The Wazuh SIEM is used for monitoring the systems with Windows, Linux, and MAC operating systems. Recent research has clearly emphasized its importance in monitoring the IoT device network. The proposed security scheme is tested and validated using the real-time network traffic monitoring datasets, machine learning models, industrial protocol feature selection, and the Wazuh SIEM solution. The decoders were written for traffic logs to extract the features for rule writing. The combination of dynamic and custom rule writing techniques has strengthened the security design and implementation for event monitoring of the industrial networks. In this research, the rules are written for intrusion detection, DDoS attacks, man-in-the-middle attacks, and event alerts in SWAT. The enhanced network security techniques carried out in this paper, using machine learning algorithms and rule-based techniques together, will be beneficial for industry 4.0 and also for the security of IoT networks in any field. The limitation of this research is that system level events cannot be monitored as any agents is not installed on the end devices, only the network events of the devices are monitored. This research will help industrial security practitioners, IoT experts, and professionals in understanding the sensitivity of the IoT network traffic and

the protection mechanisms for better security posture. In future research, more fields can be extracted from more industrial protocols for the detection of other types of attacks and anomalous events.

**Author Contributions:** Conceptualization, S.H. and G.A.S.; Methodology, H.Z., S.H. and G.A.S.; Validation, H.Z.; Investigation, H.Z.; Resources, S.H. and M.F.H.; Data curation, H.Z.; Writing – original draft, H.Z.; Writing – review & editing, S.H.; Visualization, S.H. and G.A.S.; Supervision, M.F.H.; Project administration, G.A.S.; Funding acquisition, G.A.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Not available.

**Acknowledgments:** The research work is supported by a grant from the National Center for Cyber Security, Pakistan.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

**Table A1.** PLC1 Communications in SWaT network.

PLC1 > PLC2 (192.168.1.20)	HMI_FIT201 (Service: 0x4c)_Read tag service (Request) HMI_PLANT_AUTO (Service: 0x4d)_Write tag service (Request) HMI_PLANT_RESET (Service: 0x4d)_Write tag service (Request) HMI_PLANT (Service: 0x4d) (Request with command specific data) HMI_P2_Permissivise (Service: 0x4c)_Read tag service (Request) Unconnected send: HMI_MV201 (Service: 0x4c)_Read tag service (Request)
PLC1 > PLC3 (192.168.1.30)	HMI_FIT301 (Service: 0x4c)_Read tag service (Request) HMI_P3_Permissive (Service: 0x4c)_Read tag service (Request) HMI_PLANT (Service: 0x4d)_Write tag service (Request) HMI_PLANT_RESET (Service: 0x4d)_WRITE tag service (Request) Unconnected send: HMI_LIT301 (Service: 0x4c)_Read tag service (Request) Unconnected send: HMI_Plant_Auto (Service: 0x4d)_Write tag service (Request)
PLC1 > PLC4 (192.168.1.40)	HMI_P4_Permissive (Service: 0x4c)_Read tag service (Request) HMI_PLANT (Service: 0x4d)_Write tag service (Request) HMI_PLANT_RESET (Service: 0x4d)_Write tag service (Request) HMI_PLANT_AUTO (Service: 0x4d) Request
PLC1 > PLC5 (192.168.1.50)	HMI_P5_Permissive (Service: 0x4c)_Read tag service (Request) HMI_P5_SD_Flushing_Performed (Service: 0x4c)_Read tag service (Request) HMI_P5_STATE (Service: 0x4c)_Read tag service (Request) HMI_PLANT (Service: 0x4d)_Write tag service (Request) HMI_PLANT_AUTO (Service: 0x4d)_Write tag service (Request) HMI_PLANT_RESET (Service: 0x4d)_Write tag service (Request) HMI_SHUTDOWN_FLUSHING (Service: 0x4d)_Write tag service (Request)
PLC1 > PLC6 (192.168.1.60)	HMI_P6_PERMISSIVE (Service: 0x4c)_Read tag service (Request) HMI_PLANT (Service: 0x4d)_Write tag service (Request) HMI_PLANT_AUTO (Service: 0x4d)_Write tag service (Request) HMI_PLANT_RESET (Service: 0x4d)_Write tag service (Request) HMI_LIT101 (Service: 0x4c)_Read tag service (Response with data) Connection Manager: Forward Open (Service Code: 0x54)_AND, OR (Response)
PLC1 > Historian Server (192.168.1.200)	Class (0xac) -Get Attribute all Class (0xb2) -Service (0x4c) read tag service Identity -Get Attribute all Multiple Service packets: Get Attribute all, Get Attribute list, Get Attribute Single
PLC1 > Actuator or RIO 239.192.2.63	Implicit control data messages

**Table A2.** PLC2 Communications in SWaT.

PLC2 > PLC1(192.168.1.10)	HMI_FIT201 (Service: 0x4c)_Read tag service (Response with data) HMI_PLANT_AUTO (Service: 0x4d)_Write tag service (Response with success status) HMI_PLANT_RESET (Service: 0x4d)_Write tag service (Response with data write success status) HMI_PLANT (Service: 0x4d) (Response with success status) HMI_P2_Permissisive (Service: 0x4c)_Read tag service (Response with requested data) (in response to Unconnected send: HMI_MV201 in PLC1 request) (Service: 0x4c)_Read tag service (Response with data)
PLC2 > PLC3(192.168.1.30)	HMI_P_NAOCL_UF_DUTY (Service: 0x4c)_Read service (Response with data) P2_P2078_AutoINP (Service: 0x4d)_Write service (Response with data write success status) HMI_FIT301 (Service: 0x4c)_Read Service (Request to read data) HMI_LIT301 (Service: 0x4c)_Read Service (Request to read data) HMI_FIT301 (Service: 0x4c)_Read Service (Request to read data) HMI_FIT201 (Service: 0x4c)_Read Service (Response with data) HMI_MV201 (Service: 0x4c)_Read Service (Response with data) Connection Manager_Forward Open (Service: 0x54) (Request)
PLC2 > PLC4 (192.168.1.40)	HMI_AIT402 (Service: 0x4c)_Read Service (Request to read data) Connection Manager_Forward Open (Request)
PLC2 > PLC5(192.168.1.50)	HMI_AIT501, AIT502, AIT503, AIT504 (Service: 0x4c)_Read Service (Request to read data) Connection Manager_Forward Open (Service: 0x54) (Request)
PLC2 > PLC6(192.168.1.60)	HMI_AIT202 (Service: 0x4c)_Read Service (Response with success status)
PLC2 > 239.192.3.127	Sensor Actuator or RIO (Implicit messages) CIP I/O
PLC2 > Historian Server(192.168.1.200)	Connection Manager: Forward Close Connection Manager: Forward Open CPS_ALID_P2_S01 Response with path segment error

**Table A3.** PLC3 Communications in SWaT.

PLC3 > PLC1(192.168.1.10)	HMI_PLANT_AUTO (Service: 0x4d) Unconnected send Response with success status HMI_PLANT (Service: 0x4d)_Read service Response HMI_PLANT_Reset (Service: 0x4d) Response HMI_P3_PREMISSIVE (Service: 0x4c) Response HMI_FIT301 (Service: 0x4c)_Read Service Response HMI_LIT301 (Service: 0x4c)_Read Service Response HMI_FIT301 (Service: 0x4c)_Read Service Response
PLC3 > PLC2(192.168.1.20)	HMI_P_NAOCL_UF_DUTY (Service: 0x4c)_Read service (Request) P2_P2078_AutoINP (Service: 0x4d)_Write service (Request) HMI_FIT301 (Service: 0x4c)_Read Service (Response with data) HMI_LIT301 (Service: 0x4c)_Read Service (Response with data) HMI_MV301 (Service: 0x4c)_Read Service (Response with data) Unconnected send: HMI_FIT201 (Service: 0x4c)_Read Service (Request) Unconnected send: HMI_MV201 (Service: 0x4c)_Read Service (Request) Connection Manager_Forward Open (Service: 0x54) (Response)
PLC3 > PLC4(192.168.1.40)	Unconnected send: HMI_FIT401 (Service: 0x4c)_Read Service (Request) HMI_FIT301 (Service: 0x4c)_Read Service (Response with data) HMI_MV302 (Service: 0x4c)_Read Service (Response with data) HMI_P301 (Service: 0x4c)_Read Service (Response with data) HMI_LIT401 (Service: 0x4c) (Request) Connection Manager_forward Open (Service: 0x54) (Request)
PLC3 > PLC6(192.168.1.60)	HMI_P602 (Service: 0x4c) (Request) P6_P602_AutoINP (Service: 0x4d) (Request) Connection Manager: Forward Open Request
PLC3 > 239.192.4.191	Sensor, actuator, or RIO (Implicit messages) CIP I/O messages

**Table A4.** PLC4 Communications in SWaT.

PLC4 > PLC1(192.168.1.10)	HMI_P4_Permissive (Service: 0x4c)_Read tag service (Response) HMI_PLANT (Service: 0x4d)_Write tag service (Response) HMI_PLANT_RESET (Service: 0x4d)_Write tag service (Response) HMI_PLANT_AUTO (Service: 0x4d)_Write tag service (Response)
PLC4 > PLC2(192.168.1.20)	HMI_AIT402 (Service: 0x4c)_Read Service (Response) Connection Mnager_Forward Open (Response)
PLC4>PLC3(192.168.1.30)	Unconnected send: HMI_FIT401 (Service: 0x4c)_Read Service (Response) HMI_FIT301 (Service: 0x4c)_Read Service (Request) HMI_MV302 (Service: 0x4c)_Read Service (Request) HMI_P301 (Service: 0x4c)_Read Service (Request) HMI_LIT401(Service: 0x4c) (Response) Connection Manager_forward Open (Service: 0x54) (Response)
PLC4> PLC5(192.168.1.50)	HMI_P_RO_FEED_DUTY (Service: 0x4c)_Read Service (Response) HMI_MV501 (Service: 0x4c)_Read Service (Request) HMI_MV502 (Service: 0x4c)_Read Service (Request) HMI_MV503 (Service: 0x4c)_Read Service (Request) HMI_MV504 (Service: 0x4c)_Read Service (Request) HMI_RO_HPP_SD HMI_MV501 (Service: 0x4c)_Read Service (Response) HMI_UV401 (Service: 0x4c)_Read Service (Response) HMI_FIT401 (Service: 0x4c)_Read Service (Response)
PLC4 > 239.192.5.255	Sensor, Actuator or RIO Implicit messages cip I/O

**Table A5.** PLC5 Communications in SWaT.

PLC5 > PLC1(192.168.1.10)	HMI_P5_Permissive (Service: 0x4c)_Read tag service (response) HMI_P5_SD_Flushing_Performed (Service: 0x4c)_Read tag service (response) HMI_P5_STATE (Service: 0x4c)_Read tag service (response) HMI_PLANT (Service: 0x4d)_Write tag service (response) HMI_PLANT_AUTO(Service: 0x4d)_Write tag service (response) HMI_PLANT_RESET (Service: 0x4d)_Write tag service (response) HMI_SHUTDOWN_FLUSHING (Service: 0x4d)_Write tag service (response)
PLC5 > PLC2(192.168.1.20)	HMI_AIT501, AIT502, AIT503, AIT504 (Service: 0x4c)_Read Service (Response with data) Connection Manager_Forward Open (Service: 0x54) (Response)
PLC5 > PLC4(192.168.1.40)	HMI_P_RO_FEED_DUTY (Service: 0x4c)_Read Service (Request) HMI_MV501 (Service: 0x4c)_Read Service (Response) HMI_MV502 (Service: 0x4c)_Read Service (Response) HMI_MV503 (Service: 0x4c)_Read Service (Response) HMI_MV504 (Service: 0x4c)_Read Service (Response) HMI_RO_HPP_SD HMI_MV501 (Service: 0x4c)_Read Service (Request) HMI_UV401 (Service: 0x4c)_Read Service (Request) HMI_FIT401 (Service: 0x4c)_Read Service (Request)
PLC5 > 239.192.7.60	Sensor, Actuator, RIO Implicit messages Control data CIP I/O messages

**Table A6.** PLC6 Communications in SWaT.

PLC6> PLC1(192.168.1.10)	HMI_P6_PERMISSIVE (Service: 0x4c)_Read tag service HMI_PLANT (Service: 0x4d)_Write tag service Response HMI_PLANT_AUTO (Service: 0x4d)_Write tag service Response HMI_PLANT_RESET(Service: 0x4d)_Write tag service Response HMI_LIT101 (Service: 0x4c)_Read tag service Request Connection Manager: Forward Open (Service Code: 0x54)_AND, OR Request
--------------------------	--



Table A6. Cont.

PLC6 > PLC2(192.168.1.20)	HMI_AIT202 (Service: 0x4c)_Read Service(Request) Implicit (control data) messages
PLC6 > PLC3(192.168.1.30)	HMI_P602 (Service: 0x4c) Read tag Service (Response with data) P6_P602_AutoINP (Service: 0x4d) (Response with success statement) Connection Manager: Forward Open_Response
PLC6 > 239.192.8.127	Implicit messages control data CIP I/O
PLC6 > Historian Server (192.168.1.200)	Class (0xb2) Read request service 0x 4c Class (0xb2) Read response with success status. 0x4c

## References

- Vailshery, L.S. Internet of Things (IoT) and Non-IoT Active Device Connections Worldwide from 2010 to 2025 (In Billions). Available online: <https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/#:~:text=IoT%20and%20non%2DIoT%20connections%20worldwide%202010%2D2025&text=The%20total%20installed%20base%20of,that%20are%20expected%20in%202021> (accessed on 9 July 2022).
- Sengupta, J.; Ruj, S.; Bit, S.D. A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT. *J. Netw. Comput. Appl.* **2020**, *149*, 102481. [CrossRef]
- Cyrus, C. IoT Cyberattacks Escalate in 2021, According to Kaspersky. Available online: <https://urgentcomm.com/2021/09/20/iot-cyberattacks-escalate-in-2021-according-to-kaspersky/> (accessed on 9 July 2021).
- Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network intrusion detection for IoT security based on learning techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [CrossRef]
- Municio, E.; Latre, S.; Marquez-Barja, J.M. Extending network programmability to the things overlay using distributed industrial IoT protocols. *IEEE Trans. Ind. Inform.* **2020**, *17*, 251–259. [CrossRef]
- Stouffer, K.; Falco, J.; Scarfone, K. Guide to industrial control systems (ICS) security. *NIST Spec. Publ.* **2011**, *800*, 16.
- Deogirikar, J.; Vidhate, A. Security attacks in IoT: A survey. In Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 10–11 February 2017; pp. 32–37.
- Gurunath, R.; Agarwal, M.; Nandi, A.; Samanta, D. An overview: Security issue in IoT network. In Proceedings of the 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), Palladam, India, 30–31 August 2018; pp. 104–107.
- Vishwakarma, R.; Jain, A.K. A survey of DDoS attacking techniques and defence mechanisms in the IoT network. *Telecommun. Syst.* **2020**, *73*, 3–25. [CrossRef]
- Sultan, A.; Mushtaq, M.A.; Abubakar, M. IOT security issues via blockchain: A review paper. In Proceedings of the 2019 International Conference on Blockchain Technology (ICBCT 2019), Honolulu, HI, USA, 15–18 March 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 60–65.
- Hristov, M.; Nenova, M.; Iliev, G.; Avresky, D. Integration of Splunk Enterprise SIEM for DDoS Attack Detection in IoT. In Proceedings of the 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA), Boston, MA, USA, 23–26 November 2021; pp. 1–5.
- Kouicem, D.E.; Bouabdallah, A.; Lakhlef, H. Internet of things security: A top-down survey. *Comput. Netw.* **2018**, *141*, 199–221. [CrossRef]
- Alrashdi, I.; Alqazzaz, A.; Aloufi, E.; Alharthi, R.; Zohdy, M.; Ming, H. Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 0305–0310.
- Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.
- Al-Duwairi, B.; Al-Kahla, W.; AlRefai, M.A.; Abedalqader, Y.; Rawash, A.; Fahmawi, R. SIEM-based detection and mitigation of IoT-botnet DDoS attacks. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 2182. [CrossRef]
- Brattstrom, M.; Morreale, P. Scalable agentless cloud network monitoring. In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 171–176.
- Çakmakçı, S.D.; Hutschenreuter, H.; Maeder, C.; Kemmerich, T. A framework for intelligent DDoS attack detection and response using SIEM and ontology. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
- Index of /IOTDataset/CIC\_IOT\_Dataset2022/CICIOT. Available online: [http://205.174.165.80/IOTDataset/CIC\\_IOT\\_Dataset2022/CICIOT/](http://205.174.165.80/IOTDataset/CIC_IOT_Dataset2022/CICIOT/) (accessed on 2 January 2023).
- Nguyen, T.D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A.-R. D<sup>2</sup>IoT: A federated self-learning anomaly detection system for IoT. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 756–767.

20. Marchal, S.; Miettinen, M.; Nguyen, T.D.; Sadeghi, A.-R.; Asokan, N. Audi: Toward autonomous iot device-type identification using periodic communication. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1402–1412. [[CrossRef](#)]
21. Radoglou-Grammatikis, P.; Sarigiannidis, P.; Iturbe, E.; Rios, E.; Martinez, S.; Sarigiannidis, A.; Eftathopoulos, G.; Spyridis, Y.; Sesis, A.; Vakakis, N. Spear siem: A security information and event management system for the smart grid. *Comput. Netw.* **2021**, *193*, 108008. [[CrossRef](#)]
22. Perez, S. Practical SIEM Tools for SCADA Environment. Master's Thesis, Iowa State University, Ames, IA, USA, 2018.
23. Priya Devi, A.; Johnson Singh, K. A machine learning approach to intrusion detection system using UNSW-NB-15 and CICD-DoS2019 datasets. In *Smart Computing Techniques and Applications. Smart Innovation, Systems and Technologies*; Springer: Singapore, 2021; Volume 1, pp. 195–205.
24. Itrust-Labs\_Datasets. Available online: [https://itrust.sutd.edu.sg/itrust-labs\\_datasets/](https://itrust.sutd.edu.sg/itrust-labs_datasets/) (accessed on 19 October 2022).
25. IBM. What is Security Information and Event Management (SIEM)? Available online: <https://www.ibm.com/topics/siem> (accessed on 8 October 2022).
26. Diaz Lopez, D.; Blanco Uribe, M.; Santiago Cely, C.; Vega Torres, A.; Moreno Guataquira, N.; Morón Castro, S.; Nespoli, P.; Gómez Mármol, F. Shielding IoT against cyber-attacks: An event-based approach using SIEM. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 3029638. [[CrossRef](#)]
27. Wazuh Agent. Available online: <https://documentation.wazuh.com/current/getting-started/components/wazuh-agent.html?highlight=Xpath> (accessed on 8 October 2022).
28. Class JRip. Available online: <https://weka.sourceforge.io/doc.dev/weka/classifiers/rules/JRip.html> (accessed on 2 January 2023).

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.