Journal of Mathematical Sciences and Modelling, 7(2) (2024) 60-74
Research Article

Journal of Mathematical Sciences and Modelling

Journal Homepage: www.dergipark.gov.tr/jmsm
ISSN: 2636-8692
DOI: https://doi.org/10.33187/jmsm.1484997

An Incident Management System Design to Protect Critical Infrastructures from Cyber Attacks

Uğur Gürtürk[1] **and Zeynep Gürkaş Aydın**[1*]

[1]*Department of Computer Engineering, Faculty of Engineering, Istanbul University-Cerrahpaşa, Istanbul, Türkiye*
[2]*Department of Computer Engineering, Faculty of Engineering, Istanbul University-Cerrahpaşa, Istanbul, Türkiye*
[*]*Corresponding author*

Article Info

Keywords: *Critical infrastructures, Cyber security, Information security, Log analysis*
2010 AMS: *68M25, 68M10, 68T01, 68T05*
Received: *16 May 2024*
Accepted: *4 June 2024*
Available online: *13 June 2024*

Abstract

In recent years, there has been a noticeable trend toward targeted threats to information security, where companies are now leveraging vulnerabilities and risks associated with widely used services in order to generate financial gain. Additionally, they implement numerous precautions and consistently carry out their tasks. One item that requires precautionary measures is the network devices utilized. Network devices in computer networks possess the capability to log events. These logs enable the identification of security events on the network and facilitate the implementation of precautionary measures. Various security measures can be implemented to handle such data. One of these measures is Security Information and Event Management (SIEM). It is a system that gathers and analyzes data from networks and security devices. SIEM is a technique employed to consolidate critical information within a cohesive structure. It allows for the correlation of events from different security devices, thereby improving the monitoring capabilities of cybersecurity operations centers. This study extensively covers the critical infrastructure-SIEM relationship, current studies, critical infrastructure, cyber security policies, and SIEM. Our system design was developed using the UNSW_NB15 dataset, a widely recognized dataset in cybersecurity due to its comprehensive and realistic representation of cyber threats. This dataset consists of data obtained from network traffic, various attack activities, and real-life modern normal scenarios, making it particularly relevant to our study. With the studies, a total of 10 different categories were analyzed, with the category consisting of nine types of attacks, namely Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms and Normal activities. The study is divided into two as the basic structure. The first step was carried out on Google Collaboratory, and then some experimental studies were carried out in Weka. Classifications were made using several methods, including Logistic Regression (LR), Extra Trees (XT), Support Vector Machines (SVM), Random Forest (RF), and Decision Trees (DT). These methods were chosen for their proven effectiveness in similar studies. In the application developed with Google Colabratory, we achieved 98.62% in Random Forest, 99.10% in Decision Trees, 98.87% in Logistic Regression, 95.13% success in Extra Trees and 99.12% success in Support Vector Machines. As a result of the studies and experiments carried out in Weka, we achieved 92.05% in Random Forest, 100% in Decision Trees, 100% in k-Nearest Neighbours, 100% in J48, 99.19% in Naive-Bayes and 99.35% in BayesNet achievements.

***Email addresses and ORCID numbers*: 23ugurgurturk23@gmail.com, 0000-0002-0695-0926 (U. Gürtürk), zeynepg@iuc.edu.tr, 0000-0002-4125-0589 (Z. Gürkaş Aydın)**

Cite as "U. Gürtürk, Z. Gürkaş Aydın, *An incident management system design to protect critical infrastructures from cyber attacks*, J. Math. Sci. Model., 7(2) (2024), 60-74"

1. Introduction

Security Information and Event Management (SIEM) refers to collecting and analyzing data from network and security devices, allowing for the correlation of events from various security devices. SIEM is a method that aggregates all essential information to enhance monitoring capabilities, commonly utilized by many Cyber Security Operation Centers today. It collects security data from network devices, servers, domain controllers, and more, storing and normalizing it to detect trends and threats and investigating user alerts by applying analytical methods. SIEM is a data collector, monitor, and reporting system that provides reporting capabilities to a security incident response team, matching specific rules to identify security issues and alerting them. Its primary focus is on ensuring information security, with its core objective being preserving the principles of Confidentiality, Integrity, and Availability. Missing to follow them may cause rejecting of the manuscript without further processing. Confidentiality protects information from unauthorized third parties due to its value, ranging from bank accounts, identity numbers, and credit card numbers to state secrets. Protecting this valuable data is the fundamental reason for information security, with encryption being the foremost solution when discussing confidentiality. Ensuring that data is accessible only to authorized personnel or third parties is crucial. Implementing file permissions and utilizing security protocols such as SSL/TLS (Secure Sockets Layer/Transport Layer Security) for communication over the internet are common solutions in today's world to restrict access to sensitive data and ensure confidentiality. Integrity, however, refers to protecting information from being altered. Ensuring data integrity involves maintaining data accuracy, which is crucial. Common methods to ensure integrity include hashing data using timestamps and comparing the hash value of the original message, among others. Availability refers to authorized individuals being able to access information when needed. It holds no value if the right people cannot access the information when necessary. Distributed Denial of Service (DDoS) attacks are prevalent attacks that target availability by bypassing the accessibility rule. SIEM software gathers log and event data produced by an organization's applications, security devices, and primary computer systems, consolidating them on a centralized platform. These software solutions also strive to identify interventions using the aforementioned fundamental principles. The data collected from antivirus events, firewall logs, and other sources is classified into categories such as malicious software activities and failed and successful login attempts. When a Security Information and Event Management (SIEM) system identifies a potential security threat while monitoring network activity, it produces an alert. It assigns a threat level according to predetermined rules. Figure 1.1. presents examples of security method evaluations.

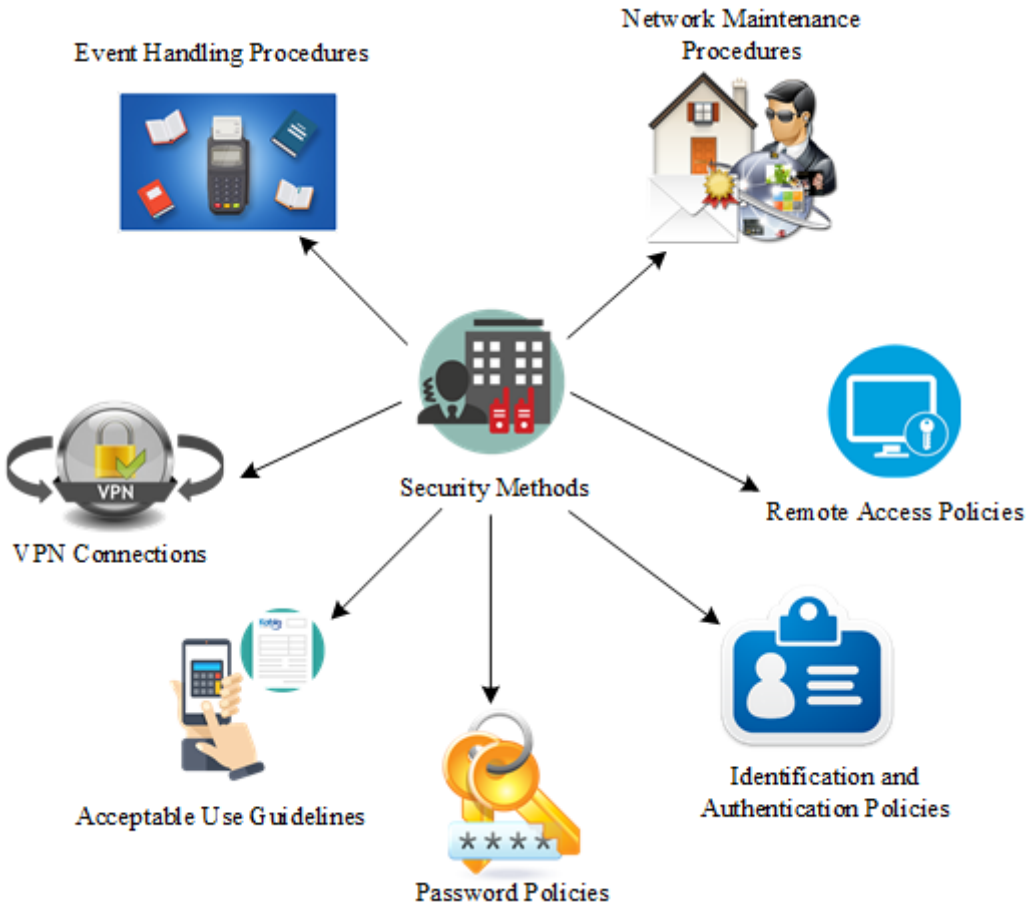

Figure 1.1: Security Method Evaluation

When examining SIEM tools, various types of vulnerabilities are noticeable. For instance, someone attempting to log in to an account 10 times in 10 minutes may not be seen as a problem, but an access request of 100 times in 10 minutes could be flagged as an attack attempt. SIEM detects such threats and generates security alerts. Its specialized dashboards and event management system enhance research efficiency and reduce time spent on false positives. When brought together and integrated, SIEM possesses a range of capabilities that provide comprehensive protection for organizations. Additionally, consolidating them into a single dashboard enables easier and more efficient management. The software used in this field allows security teams to gain insights into attacker tactics, techniques, procedures, and

known indicators of compromise through threat rules derived from information about security vulnerabilities. This can include User and Entity Behavior Analytics (UEBA), which monitors behaviors and activities to identify abnormal behaviors that could indicate a threat, lateral movement, and compromised security accounts. This is similar to the security analytics component that identifies anomalies in data to gather threat intelligence for threats that have not been encountered before. The managed rules component allows organizations to promptly respond to the most recent attacker techniques by utilizing analysts' nearly real-time updates. SIEM software generates alerts for an organization's security teams to promptly respond to threats, vulnerabilities, attacks, or suspicious behavior that it identifies. Certain software versions incorporate workflow and case management features to expedite investigations by automatically generating step-by-step investigation instructions in conjunction with searches and required actions. SIEM alerts can be customized to user requirements through customization in log management. Log management is a complex element of SIEM that encompasses three primary domains. Data collection involves consolidating an enormous amount of data from multiple applications and databases into a single location. SIEM collects event data from multiple sources throughout an organization's entire network. Data logs and flow data from users, applications, assets, cloud environments, and networks are gathered, stored, and examined instantly, enabling IT and security teams to centrally administer their network's event logs and network flow data automatically in a single central location. SIEM solutions can also incorporate third-party threat intelligence feeds to match internal security data with established threat signatures and profiles. Integrating with real-time threat feeds allows teams to identify or prevent new attack signatures. Data normalization: SIEM enables the comparison, correlation, and analysis of all different data types. It allows for classifying abnormal behaviors detected in the network across all connected users, devices, and applications and monitoring security incidents. Data analysis/security event correlation: This refers to identifying possible indicators of a data breach, threat, attack, or security vulnerability. Event correlation is a critical component of any SIEM solution. Using advanced analytics to identify and understand complex data models, event correlation provides insights to detect and mitigate potential threats to business security rapidly. Managers can be instantly alerted using customizable predefined correlation rules and take appropriate measures to mitigate them before they escalate into more significant security issues. SIEM solutions greatly enhance the Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) for IT security teams by eliminating the need for manual processes to analyze security events thoroughly. With its automated data collection and analysis capabilities, SIEM is a crucial tool for organizations that must gather and verify compliance data across their entire business infrastructure, especially those subject to different compliance formats. Given the fast pace at which the cyber security environment changes, organizations must depend on solutions that can identify and react to familiar and unfamiliar security risks. Organizations commonly select SIEM products that must adhere to various compliance formats because of their ability to collect and analyze data automatically. This enables them to gather and verify compliance data throughout the entire business infrastructure. Due to the swift rate of change in the cyber security landscape, organizations must depend on solutions capable of identifying and addressing familiar and unfamiliar security risks. Currently, organizations must choose between utilizing SIEM products or creating their own software [1–7].

2. Related Work

Computer networks are increasingly vulnerable to various types of cyber attacks due to security vulnerabilities. Therefore, cybersecurity aims to make networks as secure as possible by providing defense systems to detect suspicious activity. However, despite constantly updating their databases to detect cyber threats, traditional security systems (STS), such as firewalls, cannot fully ensure security. Therefore, the new guidelines for STS aim to utilize Machine Learning (ML) models to design reliable systems with higher detection rates and lower false alarm rates. Based on this approach, Moualla et al. (2021) proposed an approach to improve the performance of ML-based network intrusion detection systems using the UNSW-NB15 dataset. Considering studies using the UNSW-NB15 dataset and its importance in network security, they defined a new network STS. The proposed system is a dynamically scalable multi-class ML-based network STS. In the study, the Synthetic Minority Over-sampling Technique (SMOTE) was used to address the class imbalance problem in the dataset. Then, the Extremely Randomized Trees Classifier with the Gini Impurity criterion was used to select important features for each class in the dataset. An Extreme Learning Machines (ELM) model pre-trained as a binary classifier for each attack was used to detect attacks separately using the "One-versus-All" method. They demonstrated that the proposed system showed much better performance in terms of accuracy, false alarm rate, Receiver Operating Characteristic (ROC) curve, and Precision-Recall Curves (PRC) [8,9].

In the study by Zoghi et al. (2021), they addressed the class imbalance and class overlap problems that must be considered before using the UNSW-NB15 dataset. As part of the preparation for visual analysis, they applied preprocessing steps such as removing irrelevant features, normalizing features, and scaling features. After data preprocessing, they visualized the dataset using various visualization techniques to reveal and demonstrate class imbalance and class overlap issues, projecting the dataset onto 2D and 3D views using Principal Component Analysis (PCA). They then analyzed it using distribution plots, t-SNE, and K-means cluster distance maps. They argued that applying effective approaches to reduce the negative effects of any statistical or ML model on classification performance based on data is essential [10].

Aleesa et al. (2021) aimed to test the models of the UNSW-NB15 dataset by merging the entire dataset into a single file instead of separately testing them for each file, allowing models to be tested once. They then used attack types in the dataset as a new class, aiming to develop a multi-classified labeled dataset. With the dataset they developed, they investigated the performance of deep learning in both Binary and Multi-Class categories. They found that the proposed deep learning models achieved 99.59% accuracy in multi-class classification and 99.26% in binary classification [11]. Kocher et al. (2021) employed the UNSW-NB15 dataset to train classifiers, including KNN, Stochastic Gradient Descent (SGD), RF, LR, and NB, for the purpose of classification. A comparative analysis was conducted on the classifier performance, accuracy, Mean Squared Error (MSE), precision, recall, F1-Score, True Positive Rate (TPR), False Positive Rate (FPR), and feature selection technique. The UNSW-NB15 dataset was subjected to the Chi-Square filter-based feature selection technique to eliminate irrelevant and unnecessary features [12].

Mahalakshmi et al. (2021) demonstrated that the binary-encoded dataset showed maximum performance when using the Convolutional Neural Networks (CNN) deep learning method on the UNSW-NB15 dataset [13]. Iqbal et al. (2021) aimed for a more efficient machine learning approach to detect botnets in IoT networks using the PyCaret machine learning library and analyzing its overall performance [14]. Sharma et al. (2021) evaluated whether attack detection with machine learning methods, applying certain operations on the dataset using RF, XT, AdaBoost, and XGBoost methods, was accurate [15]. In the study by Sarhan et al. (2022), they transformed the UNSW-NB15, BoT-IoT, ToN-IoT, and CSE-CICIDS2018 datasets into new variants with the proposed NetFlow-based feature sets based on the XT classifier. They then compared the classification performance of NetFlow-based feature sets with registered feature sets provided with the original data

sets [16].

Pacheco et al. (2021) evaluated the security vulnerability against popular deep learning attack methods such as Multi-Layer Perceptron (MLP), DT, RF, and SVM on UNSW-NB15 and Bot-IoT datasets. They assessed the reliability of various machine learning classifiers [17]. Kılınçer et al. (2021) conducted a study in which they extensively reviewed literature studies that utilized commonly used datasets such as CSE-CIC IDS-2018, UNSW-NB15, ISCX-2012, NSL-KDD, and CIDDS-001 for the development of Security Testing Suites (STS). Additionally, they performed max-min normalization on these datasets and classified them using classical machine learning approaches such as SVM, KNN, and DT algorithms [18]. Kushwah et al. (2021) introduced a DDoS attack detection system based on Self-Adaptive Evolutionary Extreme Learning Machine (SaE-ELM) [19]. Roy et al. (2021) proposed a technique using SVM and NB algorithms, claiming to solve the classification problem of attack detection systems [20].

Ahsan et al. (2021) utilized CNN, LSTM, Bidirectional LSTM (Bi-LSTM), Gated Recurrent Units (GRU), and RF methods. They extracted accuracy rates and then reduced features using their proposed algorithm by filtering out insignificant variables, thus providing more accurate predictions [21]. Pooja et al. (2021) implemented an identity model based on deep learning methods and Bi-LSTM. The designed system was tested using KDDCUP-99 and UNSW-NB15 datasets, with the Bi-LSTM model yielding results with 99% accuracy for both datasets. Additionally, experiments were repeated by altering activation functions used in the network [22].

Thirimanne et al. (2021) aimed to identify the best machine learning algorithm for intrusion detection trained on NSL-KDD and UNSW-NB15 datasets and conducted a comparative analysis among six machine learning algorithms classified as supervised, semi-supervised, and unsupervised learning. The study revealed that supervised and semi-supervised machine learning algorithms outperformed unsupervised machine learning algorithms for both datasets. SVM and DNN perform better for NSL-KDD and UNSW-NB15, respectively [23]. Rani et al. (2022) proposed a deep neural network for addressing class imbalance. The network data underwent preprocessing through data transformation followed by min-max normalization using the Artificial Neural Networks (NN) method for analysis [24].

Okay et al. (2021) propose a methodology to reduce deficiencies in existing STSs for WLANs, aiming to create a more effective system that can dynamically detect unknown and complex attack variants. Two main contributions of the proposed methodology are claimed. The first contribution is using the Feature Selection Approach (FSAP) to reduce the number of features used, thus increasing the speed of attack detection. The second contribution is a hybrid attack detection approach using Signature and Anomaly-Based Attack Detection Technique (SABADT), which can detect attacks quickly and accurately. The proposed methodology was applied to KDD'99 and UNSW-NB15 datasets. The results were compared with existing machine learning techniques. The detection model was created using KDD'99 and UNSW-NB15 training datasets and tested on KDD'99 and UNSW-NB15 test datasets. The obtained accuracy rates of 99.65% and 99.17% were considered quite high compared to leading methods in the literature. Additionally, common tools were used to obtain a mixture of normal activities and existing attack behaviors to test new attacks. Different types of attacks were captured using Wireshark, and some of these captured attacks were used only in the testing phase. In this test scenario, attacks were detected with a 99.69% accuracy rate [25].

Sekhar et al. (2021) proposed a new Attack Detection Technique using Fruitfy Optimization with Deep Autoencoder. Firstly, missing values in the dataset were replaced using the Fuzzy C-Means Rough Parameter (FCMRP) algorithm, which handles uncertainty in datasets by leveraging fuzzy and rough clusters while preserving important information. Next, robust features were extracted from Autoencoder with multiple hidden layers. Finally, the obtained features were fed into a Backpropagation Neural Network (BPN) for classifying attacks. Additionally, neurons in the hidden layers of Deep Autoencoder were optimized using the population-based Fruitfy Optimization algorithm. Experiments were conducted on NSL KDD and UNSW NB15 datasets. The computational results of the proposed attack detection system using a BPN-based deep autoencoder were compared with NB, SVM, Radial Basis Function Networks (RBFN), BPN, and Autoencoder with Softmax [26].

With the rapid development of the Internet, cyber-attack methods have become more complex, causing increasingly significant damage worldwide. Therefore, in Yang's (2021) study, given the growing importance of timely detection of malicious behavior on the Internet as a significant security issue, a deep learning-based STS is proposed, implementing bidirectional LSTM architecture and using the UNSW-NB15 dataset for training and testing. Experimental tests in the study showed that the STS effectively detects known or unknown malicious behaviors in the current network environment [27].

Han et al. (2022) introduced an Intrusion Detection Hyperparameter Control System (IDHCS) that manages and trains a k-means clustering module as a reinforcement learning model using a DNN feature extractor and Proximal Policy Optimization (PPO). The IDHCS system utilizes a DNN feature extractor to extract the most significant features in the network environment. It then identifies unauthorized entries by employing k-means clustering. The reinforcement learning model, which combines PPO and iterative learning, is designed to automatically enhance performance in the network environment of IDHCS. System performance was assessed through experiments using the CICIDS2017 and UNSW-NB15 datasets. Achieving an F1 score of 0.96552 in CICIDS2017 and a score of 0.94268 in UNSW-NB15 demonstrates the level of performance attained. A series of experiments were conducted by merging the two datasets to establish a more comprehensive and intricate testing environment. The diversity of attack types in the experiment increased due to merging the datasets. Achieving an F1 score of 0.93567 in the combined dataset demonstrates a significant performance improvement of 97% to 99% compared to the CICIDS2017 and UNSW-NB15 datasets. The results indicate that the proposed Intrusion Detection and Host-based Control System (IDHCS) automates learning to detect new types of attacks and enhances the performance of Security Threat Scenarios (STS). This is achieved by utilizing unauthorized entry detection features independent of network environment changes and through continuous learning [28].

Al-Gethami et al. (2021) aimed to demonstrate that ML-based STSs' classification accuracy can be influenced by certain factors using DT, RF, SVM, ANN, and NB algorithms. The factors considered in the study were the method of using the dataset for training and testing, removal of outliers and extreme value instances, addition of mislabeled instances, and the use of ensemble learning techniques. The study showed various effects of these factors on classification accuracy; in some cases, the impact of noise in the data on the accuracy of the RF algorithm was demonstrated to have negative effects on classification accuracy. However, the negative effects of these factors were shown to result in significant improvement when applied to the UNSW-NB15 dataset with classification methods such as DT, RF, SVM, ANN, and NB [29].

Meliboev et al. (2022) proposed a DL method to implement an effective and adaptive STS using CNN, LSTM, Recurrent Neural Network (RNN), and Gated Recurrent Units (GRU) methods [30]. El-Sayed et al. (2021) proposed a suggestion for attack detection in intelligent transportation systems using only 20 features from the dataset. SVM, ANN, and NB algorithms were used as methods, and it was found that the proposed method increased accuracy [31]. Kim et al. (2021) introduced a model that utilizes LSTM, a machine learning technique, for

predicting the timing of attacks in the prediction domain of four different types of STS attacks. The UNSW-NB15 dataset was employed for this purpose. During training, the LSTM inputs for each attack comprised 80% and 90% of the examples. The epoch values were incremented from 1 to 26. The model accurately predicted the number of attack points for Back Door, DoS, Exploit, and Generic attacks. The output values produced by the model were examined to verify the precise timing of the attack. During the analysis of Dos, Backdoor, and Generic attacks, it was discovered that the expected attack occurrence was T[i]+2 time slots, in contrast to the actual attack timing of T[i]. Nevertheless, in the context of an Exploit attack, researchers observed a series of overlapping attacks and encountered challenges in accurately interpreting the numerical values generated by the LSTM model. Consequently, it became arduous to ascertain whether an attack had occurred. [32].

Hossain et al. (2021) evaluated the performance of four popular classifiers, DT, SVM, RF, and NB, using the Pandas and SKlearn libraries with Python language on the UNSW-NB15 dataset. They used the UNSW-NB15 dataset with 43 features comprehensively. The experimental results showed improved RF, DT, and NB accuracy compared to previously reported results by Apache Spark and its MLlib [33]. In Dutt et al. (2021), the authors extensively analyzed and examined two commonly used datasets in intrusion detection, namely KDD'99 and UNSW-NB15 datasets. Data preprocessing was performed for both datasets, considering missing, redundant, and noisy data, using the Weka data mining tool. A new dataset using a week's worth of network traffic was also introduced. This dataset was prepared considering resource consumption-based features, as these features played a significant role in detecting unauthorized entries. It was also emphasized that preprocessing is crucial in any predictive scenario applicable to intrusion detection systems. Data preprocessing for KDD'99 and UNSW-NB15 datasets was demonstrated using the Weka data mining tool [34].

Kim et al. (2021) introduced a deep learning model that utilizes LSTM and GRU on the UNSW-NB15 dataset to forecast the occurrence of attacks in a dataset for intrusion detection systems. The application of finite state machines was utilized to convert floating-point values into corresponding binary values, enhancing the model's accuracy. Consequently, they discovered that the precision of GRU and LSTM, measured by weighted F1 scores, was roughly 13% and 18% greater, respectively [35]. Intrusion detection systems (IDSs) are critical for protecting ICT infrastructures (STSs). Deep learning and machine learning are widely used to process high-dimensional, complex data to provide robust solutions for new attack types and complexity control. IDSs using unsupervised machine learning techniques detect and capture attack types such as known, unknown, and zero-day attacks. Sing et al. (2021) designed a structure using the concept of One-Class SVM (OCSVM) and active learning to detect threats without prior knowledge loss. The performance of this structure was tested using the CIC-IDS2017 dataset and compared with the UNSW-NB15 and KDD cup 99 datasets, demonstrating superior performance [36].

Silva et al. (2022) highlighted the lack of accurate evaluation, comparison, and distribution due to the scarcity of well-structured datasets in machine learning mechanisms to detect unauthorized entries on the network. They proposed a statistical analysis of the features in the four most commonly used datasets. It was concluded that the analyzed datasets should not be used as a comparison to create new anomaly-based mechanisms for intrusion detection systems. Instead, the correlation between features was analyzed to control outliers or data imbalance. DT, LR, RF, and XGBoost methods were applied in the analysis phase [37]. In the study by Priya et al. (2021), performance analysis and feature analysis of STS datasets were presented by comparing the Sigmoid classifier with datasets such as KDDCUP99, DARPA 1999, TWENTE 2008, UNIBS 2009, ISCX2012, NGIDS-DS 2016, UNSWNB15, and CICIDS2017 [38].

The study conducted by Man et al. (2021) introduced a network intrusion detection model based on machine learning. Initially, the study performed preprocessing on the dataset and then transformed it into images. More complex convolutional neural networks incorporating residual blocks were developed to capture important features effectively. Additionally, focal loss was employed to handle the class imbalance problem in the training set and improve the detection of minor attacks in the test set. The model employed batch normalization and global average pooling techniques to mitigate and enhance the accuracy of false detections. The experiments demonstrated that the suggested model has the potential to enhance the accuracy of attack detection in comparison to current models [39]. Ashiku et al. (2021) propose using deep learning architectures to develop an adaptable and flexible network STS for detecting and classifying network attacks. The study addresses network intrusion detection systems using a network traffic dataset that includes common cybersecurity vulnerabilities. The combined system merged with a semi-dynamic hyperparameter adjustment approach, showed significant improvements in multi-class models compared to similar deep learning-based network STS. The proposed approach achieved an overall accuracy of 95.4% and 95.6% for pre-segmented and user-defined multi-class classification, respectively [40].

Hooshmand et al. (2022) propose a model using one-dimensional CNN architecture. The approach initially separates network traffic data into Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and other protocol categories. Each category is then addressed independently. Feature selection using the Chi-square technique was performed before training the model, followed by oversampling using the synthetic minority oversampling technique (SMOTE) to address the class imbalance. The proposed method achieved weighted average F-scores of 0.85, 0.97, 0.86, and 0.78 for TCP, UDP, OTHER, and ALL categories, respectively [41]. The study by Kamarudin et al. (2021) aims to improve the accuracy of detecting DoS attacks using ML by identifying the most important features in the dataset and analyzing abnormal network activities more accurately. Experimental results show the accuracy of the proposed system in detecting DoS attacks using the RF method, with accuracy rates of over 98.8% compared to traditional approaches for each dataset [42].

Magan-Carrión et al. (2021) presented a new methodology for ML-based Network STS called Reliable-NIDS (R-NIDS) and proposed a new dataset named UNK21. They collected the system from the three well-known attack datasets (UGR'16, USNW-NB15, and NLS-KDD), each from their own network environment, with different features and classes, using an existing data collection approach. They showed increased attack detection accuracy with this dataset [43]. The study by Sharma et al. (2021) aimed to increase attack detection accuracy on the UNSW-NB15 dataset using RF, XT, AdaBoost, and XGBoost methods. Comparative analysis of all classifiers used standard evaluation parameters, and graphs were utilized to examine outlier values in each feature. The obtained graphs provided insights into each feature's minimum, maximum, and median values for each class label. Graphical analysis of the data presented frequency distributions for each feature and explained the correlation between features. Dot plots revealed that the mean value of each feature was around zero for attack labels and varied significantly for normal class labels. The Random Forest classifier achieved the highest accuracy of 86.9%, while AdaBoost had the lowest accuracy despite minimal differences in performance among all classifiers, yielding nearly similar results [44].

In the study by Chew et al. (2021), ten important machine classifiers (ZeroR, Random Tree, REPtree, Decision Stump Adaboost, Bayesnet, NB, RF, SMO, and J48) were used to evaluate three selected NIDS datasets. The analysis revealed that the UNSWNB15 dataset is more suitable for attack detection in low-footprint scenarios, the CIDDS-001 dataset is suitable for detecting reconnaissance techniques, and GureKDDCup contains most features similar to the previous comparison KDDCup'99 and can be used as an alternative dataset [45]. Acharya

et al. (2021) discussed the effectiveness of various machine learning algorithms, including RF, J48, NB, BayesNet, Bagging, AdaBoost, and SVM, using Weka on network log traffic datasets from KDD99, UNSW-NB15, and CIC-IDS2017. They examined the impact of changing the output class counts of network attack datasets on sensitivity, TPR, FPR, Area Under Curve (AUC) of the ROC Curve, and misclassified percentage. The study showed that reducing the target class count increased the performance of machine learning classifiers and adding a highly correlated feature to the output class improved classifier performance [46]. In the study by Dlamini et al. (2021), they introduced a data generator model named DGM to improve anomaly detection accuracy in the anomaly detection domain. They conducted experiments on NSL-KDD and UNSW-NB15 datasets to demonstrate the effectiveness of their approach. They compared their method with the existing statistical approach SMOTE and found that DGM performed better [47].
Pavlov et al. (2021) addressed the requirements for creating datasets for use in unauthorized entry detection systems on networks and analyzed modern datasets. They created a requirement list for datasets used in test methods for identifying attacker groups, determined weights for requirements, and established a usability rating for modern datasets. They also proposed an alternative data source to meet requirements inadequately addressed by current datasets [48]. The study by Güler et al. (2021) utilized supervised learning classification-based algorithms, namely RNN, LSTM, and GRU, to compare their effectiveness in detecting network attacks using the UNSW-NB15 dataset. The study's primary objective was to evaluate the efficacy of deep learning algorithms and identify the most optimal model for detecting and classifying attacks. The models achieved accuracy values of 98%. The false positive rate (FPR) values for the RNN, LSTM, and GRU models were determined to be 0.014, 0.011, and 0.011, respectively [49].

3. Dataset and Methods

Data analysis can be defined as the process of transforming raw data into useful information. It leverages big databases to extract insights with predictive analytics, enabling companies, organizations, or individuals to focus on the most critical information in data warehouses. Data analysis possesses significant potential as an emerging technology that can forecast future trends and behaviors, thereby enabling businesses to make well-informed decisions. Data analysis employs advanced mathematical algorithms to segment data, evaluate the likelihood of future events, and facilitate effective decision-making and action. Existing software and hardware platforms can quickly incorporate data analysis techniques to improve the usefulness of current information sources and seamlessly integrate with new products and systems. The primary characteristics of data analysis are as follows:

- Automated discovery of patterns
- Prediction of potential outcomes
- Generation of practical insights
- Emphasis on big datasets and databases
- Proficiency in addressing complex inquiries that cannot be resolved using basic query and reporting methods.

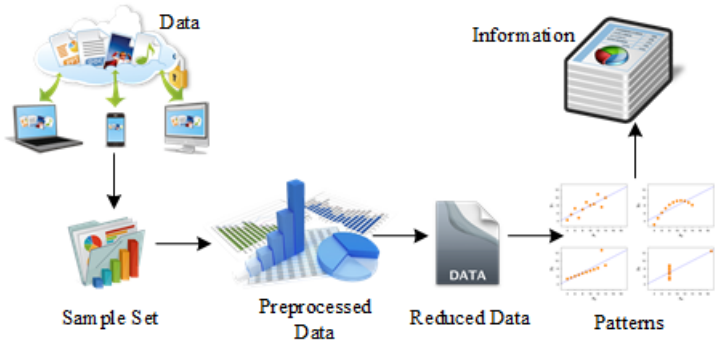

Figure 3.1: Steps of Data Analysis Process [50]

In Figure 3.1, the steps of the data analysis process are provided. Among the given steps, the first and most crucial step of data analysis is selecting data to be analyzed and making decisions to be derived from it. It is one of the most time-consuming steps in the data analysis process steps. In this step, the data generated in the system should be well-selected, and the analysis should be carried out meticulously to ensure the accuracy of the decision. Another important step for successful data analysis implementation is preprocessing, where data is prepared for later use. The success achieved at this stage significantly influences the success of the outcome. The proper and efficient execution of the preprocessing stage will lead to clear and definitive results.
In the data reduction phase, the sample dataset obtained from collected data undergoes a specific preprocessing stage to obtain useful and real information. In this phase, although the data has gone through a certain preprocessing, the data reduction process that will not be used in subsequent steps is carried out to bring it into the necessary format for later use. For the complete implementation of data analysis and the method, data mining methods are applied to the reduced data according to the purpose of the study. One or more known data analysis techniques can be applied to the reduced data at this stage. Moreover, different data analysis methods can be combined to ensure more accurate and clear information. After applying data analysis techniques to the obtained data, interpretations can be made of the results. The correctness of the interpretations can be determined based on the results of other data analysis techniques applied to the same data. Therefore, it should be determined which method among the applied methods reached a more accurate result. The success obtained from the applied methods and the interpreted result are compared with other studies in the literature to ensure that the best result is achieved and the results are validated [50, 51].
Data preprocessing is considered a crucial step in the process of data mining and data analysis, where raw data is taken and transformed into a format that can be understood and analyzed by computers and machine learning methods. This stage is among the most important steps in

the analysis. Performing a healthy and appropriate preprocessing will significantly increase the success rate. In this stage, unstructured data in text and image formats should be cleaned and formatted before analysis. Processed data is even more important than the most powerful algorithms; machine learning models trained on poor data can be detrimental to the analysis attempted and yield meaningless results. When data is properly subjected to preprocessing and cleaned, much easier and more accurate results can be obtained.

In this study, the UNSW NB15 Dataset, consisting of data obtained from network traffic and various attack activities, has been analyzed alongside real-life modern normal scenarios. The dataset includes both normal and abnormal network activities. The total number of samples in the dataset is 2,540,044, structured as a subset containing 257,673 samples. The training set comprises 175,341 samples, while the test set contains 82,332 samples. The objective of the study was to perform an analysis of nine types of attacks, namely ”Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms,” along with Normal activities, and to obtain the most reliable classification methodology through various classification algorithms based on this analysis. There are 49 features in the samples of this dataset, categorized into time, flow, content, and additional features, making a total of separate groups of features.

In the study, the initial procedure involved importing the necessary libraries. The dataset was generated to address missing values, among others. Data analysis and visualization techniques were employed to gain a deeper comprehension of the characteristics by analyzing and detecting patterns in the data. Several studies were conducted to investigate the data distribution and the correlation between different features and to detect anomalies in the dataset. The dataset is an upgraded contemporary substitute for the traditional KDD dataset, which has become obsolete and contains numerous anomalies. The UNSW-NB15 dataset exhibits a higher degree of class balance than other datasets used for unauthorized entry detection. Despite being smaller and less redundant than other datasets, the UNSW-NB15 dataset is still adequate for training a model with high accuracy. Data preprocessing is the initial stage in the development of machine learning models. The data collected by the system frequently exhibits inconsistencies; preprocessing is conducted to ready the data for visualization and analysis. Data cleaning analyses were performed during the initial study, and no missing values were detected in the dataset. Every aspect was thoroughly analyzed, and normalization was deemed necessary due to the skewed data before visualization. At first, the ”id” attribute was eliminated from both datasets because of the distinct numbering in this characteristic that hindered the detection of duplicate records. Subsequently, by utilizing the Pandas library’s inherent functions, replicated data was detected and eliminated from both the training and test datasets. In addition, the dataset’s attributes that contained attack types were eliminated due to their lack of relevance for binary classification problems.

The encoded variables and extracted features were examined; initially, relationships among features were deduced since they only increased the dimensionality of the dataset. Before proceeding to the data analysis step, outliers were removed from the dataset to understand better how features were distributed. The PyCaret library creates a sequential arrangement consisting of all necessary function blocks or modules that can simplify the model training process. This includes the data preprocessing stage as well. Functions related to data preprocessing and preparation are important components of the library. As a result, the PyCaret library can automatically handle these functions. As a developer, the task is to call the setup function and configure the necessary functions. To deal with missing values in the dataset, PyCaret defaults to using the mean value of the feature in the case of numerical features. If necessary, a median value can also be manually selected, but in this study, it was observed that the median value provided more suitable results than the standard value.

Normalization is performed using the ”z-score” by default in PyCaret, and the yeo-johnson method is used for transformation. Data visualization techniques have been used to better analyze and identify data patterns to understand the features. Based on predictions, decisions have been made for subsequent stages regarding which optimal features to include in the dataset during training or which relevant features can be removed to improve the model’s accuracy. The Apriori algorithm created association rules for the given dataset, and sorting was done based on confidence and lift values. After sorting, each row has been duplicated to include antecedents and consequents. Following this process, sorting was done again to create a list. Frequent item columns were identified for the entire dataset, and the Apriori algorithm was again used to find subsets of frequently used items. The rule was created from subsets, and each frequent column was added for adjustment. A repeated sequence was created to find all possible columns in the dataset. The One Hot Encoding technique was applied to each nominal field in the dataset. The reason for employing this approach is that machine learning models are incapable of cannot handling our dataset, as they can only process numerical values. Therefore, it is necessary to preprocess categorical features for machine learning models. One Hot Encoding is a frequently employed technique for preprocessing categorical attributes in machine learning models. This encoding method generates a distinct binary attribute for every potential category and assigns a value of ”1” to the attribute corresponding to each instance’s original category. Converting categorical data into numerical form is a crucial step in feature engineering during the training process of learning techniques. Encoding is typically utilized to represent data in integer form. The integer-coded variable is eliminated, and a distinct binary variable is introduced for every unique integer value. A column containing encoded categorical data with labels is obtained and split into multiple columns. The numbers are substituted with random ”1”s and ”0”s according to the values assigned to each column. Although this method may benefit certain situations with a clear order among category values, it may encounter difficulties and result in poor performance when applied to input data that lacks any inherent order. Although the ”Label Encoding” method performs a similar process, it is recommended to use this method when categories are ordered and uniform. Therefore, the ”One Hot Encoding” encoding method was used, as it showed lower performance in the previous classification. Subsequently, all important parameters and objects were saved to disk to apply the same process to test data. All methods were applied to the test dataset as well, and it was considered that the preprocessing stage was completed. The final step of data preprocessing is feature scaling, which transforms all numerical values of the dataset to a standard scale. The feature scaling process was performed using the MinMaxScaler and StandardScaler functions in Scikit-Learn.

3.1. Classification

3.1.1. Classification with logistic regression

After all the preprocessing steps mentioned above, training the data primarily started with LR classification. LR is a supervised learning algorithm that predicts a dependent categorical target variable. It is a method commonly used for binary classification problems (problems with two class values). Hyperparameters such as ”alpha” and ”penalty” were set for LR. The feature scaling method for each ML algorithm is selected based on performance metrics after a comparative analysis. LR is advantageous for large datasets that need to be categorized; therefore, the LR method was initially used for classification. This method utilized the Sigmoid function, a mathematical function that maps predicted values to probabilities. The Sigmoid function maps a real value to another value in the range of ”0 and 1,” as regression values

must be between 0 and 1, creating a curve like an ”S” shape. The S-shaped curve can be called the Sigmoid or logistic function. A threshold value defining the probabilities of 0 or 1 was used in LR. Values above the threshold are biased towards 1, while values below the threshold are biased towards 0.
The dataset had been properly prepared for these operations and then trained using the training set. The Logistic Regression class from the Sklearn library was imported to provide training or fit the model to the training set. Once the class was imported, a classifier object was instantiated and utilized to train the model using logistic regression. The model was effectively trained on the training set, resulting in accurate predictions on the test set data. Subsequently, a confusion matrix (CM) was generated to evaluate the precision of the classification. This was created using the confusion matrix function from the Sklearn library. Once the function was imported, it was invoked using a fresh variable. The function requires two parameters: the real values and the predicted values generated by the classifier. The Matplotlib library was utilized to represent the outcome graphically. The LR training set result was effectively visualized to distinguish between attack and non-attack data in this classification task.

3.1.2. Classification with extra trees classifier

XT can be evaluated similarly to RF in terms of creating multiple trees and separating nodes using random subsets of features, but there are two important differences. Firstly, it does not bootstrap observations; nodes are split into random splits rather than best splits. Initially, multiple trees are created by default, without replacement nodes, and all observations are divided into random splits based on random splits between selected features in a random subset. Parameters such as max depth, min_samples_split, and min_samples_leaf are adjusted for classification. Subsequently, inputs are provided for cross-validation. After determining the number of trees to be constructed, the classification model is built, and classification is performed.

3.1.3. Classification with support vector machines

SVM is a supervised machine learning algorithm that can be used for classification and regression problems, although it is typically used for classification. When given data with 2 or more labeled data classes, it acts as a discriminative classifier defined formally by an optimal hyperplane that separates all classes. Subsequently, new examples mapped to the same space can be categorized based on which side of the gap they fall into. In SVM, support vectors are the closest data points to the hyperplane, which are points that would change the position of the hyperplane dividing the dataset. Therefore, they can be considered critical elements of a dataset. This geometry indicates that the hyperplane is a flat subset of one dimension less than the surrounding space. For instance, the hyperplane of an ndimensional space is a flat subset of dimension n-1, inherently dividing the area into two half-spaces. To train our classifier, we used a function called ”Hinge Loss,” known as ”Margin Loss.” Hinge loss is specifically used for ”maximum margin” classification, particularly for SVMs. The regularization balances between maximizing the margin and minimizing the loss. The goal is to find the decision surface that is maximally distant from any data point. The objective is to minimize and optimize the loss to learn the weights. The objective function is derived to obtain gradients, using the total rule to differentiate each term separately. The aim here is to update the weight vector using gradients of both terms if there’s a misclassified example; otherwise, the weight vector is updated only with the regularization’s gradient if the classification is correct. The learning rate determines the length of steps the algorithm takes down the gradient on the error curve. It’s important to note that the algorithm should not overshoot the optimal point, the learning rate shouldn’t be too high, and convergence shouldn’t take too long. The regularization controls the balance between achieving low training error and having the ability to generalize your classifier to unseen data with low test error. A regularization parameter of ”1/epochs” was chosen, so this parameter decreases as the number of epochs increases. Additionally, the height of the regularization is considered to be the error.

3.1.4. Classification with random forest classifier

RF can be characterized as a classifier built on decision trees. Each tree in RF makes a class prediction, and the class with the most similarity becomes the model’s prediction. In the study, estimators called predictors were initially set and assigned to an array. Then, cross-validation was performed, and their visualization was ensured. Next, the maximum depth setting and minimum sample split were performed. The purity of the subset is calculated when ”Gini” is randomly selected, quantifying the probability amount of misclassified specific features. It can be called pure if all elements are solely connected to a single class. The Gini Index criterion includes entropy as a factor, resulting in values between 0 and 1. A value of 0 indicates perfect classification purity, where all elements belong to a single class. The value ”1” denotes the stochastic allocation of elements across various categories. A Gini Index value of 0.5 indicates a balanced distribution of elements across different classes. Features with the lowest Gini Index values were prioritized when generating the decision tree. As a result, several parameters were modified with appropriate values for the classifier. The performance is thought to rely less on ”n_estimators,” ”max_depth,” and other parameters. We selected the optimal parameters for our model and performed the classification accordingly.

3.1.5. Classification with decision tree classifier

Decision trees (DT) are non-parametric supervised learning techniques that can be used for both classification and regression tasks. Decision Trees (DT) aims to construct a model that accurately predicts the value of a target variable. This is achieved by extracting simple decision rules from the features of the data. The root node in classification serves as the starting point for the decision tree. It represents the entire dataset divided into two or more homogeneous clusters. On the other hand, leaf nodes are the final output nodes, and once a leaf node is reached, the tree cannot be further divided. The process involves splitting, where the decision node/root node is divided into sub-nodes based on given conditions. A new subtree is created with the split of the tree. Pruning is performed to remove unwanted branches from the tree. The deeper the tree, the more complex the decision rules, and the more suitable the model. The tree construction began with the root node, creating a tree that includes the entire dataset. Using feature selection, the best feature in the dataset was identified, and a decision tree node containing the best attributes’ possible values was created by splitting into subsets containing the best attributes. New decision trees were iteratively created using the subsets of the generated dataset; then parameters were set for ”max_depth,” ”min_samples_split,” and ”min_samples_leaf” in classification. Subsequently, inputs were provided for cross-validation. After designing the number of trees to be

built, a classification model was created, and classification was performed. Following these processes, the performance rate was compared with studies in the literature and found to be lower. Therefore, the idea of conducting correlation analysis was considered to improve the performance. Correlation analysis is a statistical method used to evaluate the strength of the relationship between two variables that are measured on a continuous scale. This analysis method is employed to ascertain the potential correlations between variables. Correlation analysis is essential in the study because it accounts for the potential influence of unmeasured variables on the research outcomes, which is often misunderstood. A correlation between two variables indicates that any systematic change in one variable will also result in a systematic change in the other; thus, the variables can change in tandem. The magnitude of the correlation, whether positive or negative, is contingent upon the measured numerical values. The positive correlation is observed when there is a simultaneous increase in both variables; one variable's high values correlate with the other's high values. When one variable decreases and the other increases, a negative correlation exists; that is, the high values of one variable are correlated with the low values of the other. Columns in the dataset were analyzed, and columns with high correlation were removed. Initially, a value of 0.97 was considered a limit in the created correlation matrix, and columns with relationships below this value were not included in the analysis stage. The columns with the most significant relationship among them are provided in Table 3.1.

Table 3.1: Columns with the highest correlation in the dataset

COLUMN NAMES					
'spkts'	'dpkts'	'dpkts'	'sbytes'	'dbytes'	'swin'
'sloss'	'dbytes'	'dloss'	'sloss'	dloss'	'dwin'
'ct_srv_src'	'ct_dst_src_ltm'	'is_ftp_login'	'ct_srv_dst'	'ct_srv_dst'	'ct_ftp_cmd'

The data types of the fields are given in Table 3.1, and all other fields were extracted in the study. This aimed to avoid type errors when analyzing different data types simultaneously. The fields in the dataset are divided into "Nominal" and "Numeric." Nominal columns, which are non-numeric columns, are seen as proto, service, state, and attack_cat. In the preprocessing steps, symbolic features such as protocol, service, flags, etc., were processed, and all symbolic and non-numeric features that do not actively contribute to attack detection and do not require processing were removed. To improve performance, nominal attributes were converted to numeric attributes. Subsequently, an advanced forward-selected wrapper selection method was used to improve performance and reduce features. The class values of the UNSW B15 dataset were transformed from numeric to nominal form following each of these stages. The Weka data mining tool offered a numeric-to-nominal filter to convert numeric values to nominal ones. In practice, the attack class is appended as the last attribute to the dataset. On the contrary, attribute number 45 of the UNSW NB15 dataset comprises class values expressed as numeric values. In Weka, the objective of each experimental group is to investigate one particular aspect of the performance of machine learning-based intrusion detection in the presence of noise. In the first experimental group, a baseline is established to assess the potential effects of noise on machine learning-based intrusion detection systems (IDSs) and evaluate the efficacy of noise-filtering methods. Moreover, it is employed to determine the effectiveness of ensemble learning algorithms. Compared to the outcomes of subsequent stages, this stage's results are evaluated. An algorithm will filter the datasets in the second experimental set for interquartile noise. By applying this filter to the datasets, outliers, and extreme values can be identified. A comparison will be made between the outcomes of this phase and the initial target. Following that, datasets containing unauthorized entries were injected with noise in varying proportions. The noisy data was subsequently contrasted with the baseline acquired during the initial phase of noise introduction. Noise filtering was executed through the exclusion of noisy instances and the determination of varying levels of noise. This will facilitate the analysis of the impact of noise on machine learning algorithms when executed on datasets containing unauthorized entries devoid of outliers and extreme value instances. Furthermore, before noise filtration, noise injection was implemented. Noise is introduced into this experiment by manipulating a specific proportion of the training set's labels. Outliers and extreme values are eliminated to filter out noise. Finally, the effect of ensemble learning techniques applied to unauthorized entry datasets on the precision of ML algorithms was investigated. The general flow diagram of the proposed system design is presented in Figure 3.2.

In this study, initially, analyses for data cleansing were conducted, examining the description of each feature, and normalization was ensured before visualization due to the skewness of the data. Duplicate records were removed, and subsequently, attributes containing attack types in the dataset were eliminated as they were unnecessary for binary classification problems. To deal with missing values in the dataset, PyCaret defaults to using the mean value of the feature, particularly for numerical attributes. It has been observed that using the median value yields more suitable results compared to the standard value. Therefore, normalization and transformation processes have been conducted. To observe relationships in the dataset, data was visualized, association rules were generated using the Apriori algorithm, and sorting was performed based on confidence and lift. The conceptual framework is depicted in the figure. In the structure shown in the figure, the first area encompasses normal and abnormal network activities prepared by the IXIA PerfectStorm tool at the University of New South Wales in Australia. After a series of small steps, each nominal field in the dataset was encoded using the "One Hot Encoding" method. Later, in order to apply the same process to the test data, all significant parameters and objects were saved to disk. Feature scaling was performed. Subsequently, classification was carried out using LR, XT, SVM, RF, and DT methods. Then, analyses and experiments were conducted again using the KNN, BayesNet, NB, DT, and RF algorithms with both preprocessed data and the entire dataset combined using a technique called "Cross Validation."

Noise Injection: Noise can manifest differently, including incorrectly labeled data or inaccurately classified examples. To obtain more precise outcomes, it is advisable to eliminate any noisy data, as utilizing clean data assists in preventing potential problems such as overfitting. The "AddNoise filter" in Weka introduces noise into data sets. The default value for noise injection in the AddNoise filter is 10

The study is fundamentally divided into two parts. The first step was conducted on Google Colab, followed by various experimental works on Weka. Applications in Weka are generally divided into two main parts. The first part involves working on the entire dataset, while the second part entails taking pre-processed data from Colab and subjecting it to analysis again using the abovementioned steps.

Figure 3.3 shows a general block diagram showing the overall steps and algorithmic overview of two parts of the experimental studies.

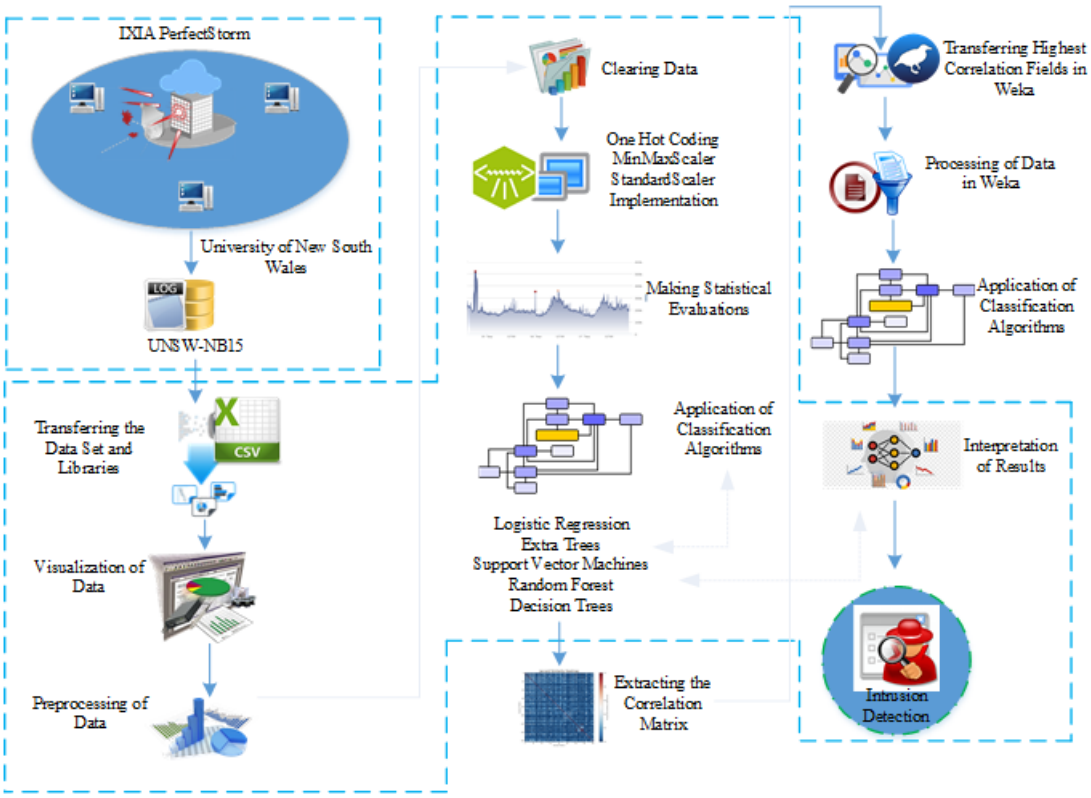

Figure 3.2: General flow diagram of the proposed system design

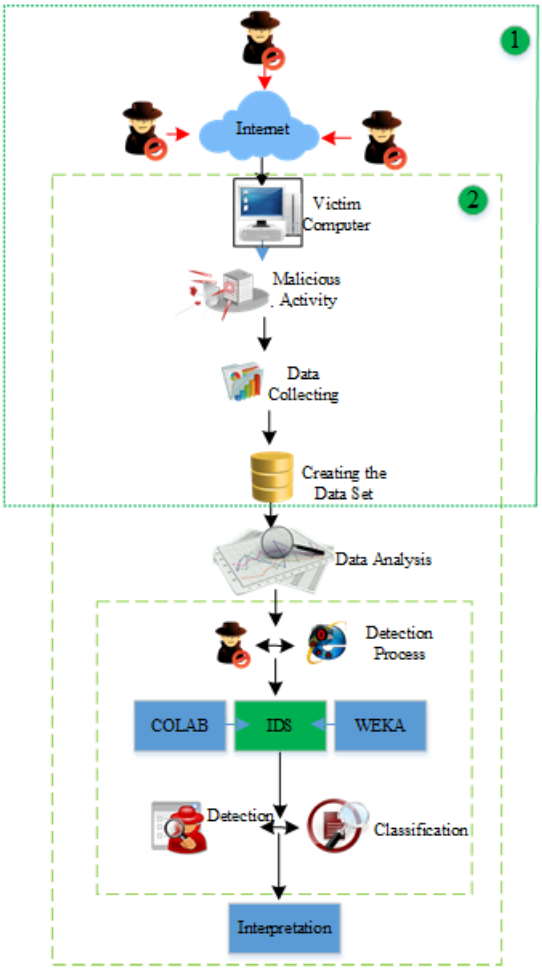

Figure 3.3: Block diagram of experimental studies on WEKA and Colab Platforms

The main steps of experimental studies conducted in Weka are as follows:

1. Loading the training data set,
2. Converting label values from numeric to nominal,
3. Training the ML algorithm on the training data set,
4. Running the ML algorithm on the test data set,
5. Reverting to step 3 using other ML algorithms,
6. Removing noise by manipulating the labels of training examples,
7. Training the ML algorithm on the noisy training data set,
8. Running the ML algorithm on the test data set,
9. Repeating step 3 until each ML algorithm is trained and tested with different noise levels,
10. Loading the unauthorized entry data set for the training section,
11. Converting label values from numeric to nominal,
12. Performing noise filtering by removing outliers and extreme values,
13. Training the ML algorithm on the noisy training data set,
14. Running the ML algorithm on the test data set,
15. Initiating ensemble learning methods (bagging-boosting),
16. Using a base classifier as an ML algorithm,
17. Training the ensemble of classifiers on the data set,
18. Testing the ensemble of classifiers on the test data set,
19. Repeating the steps until each ML algorithm is used with both ensemble learning methods.

The first stage involves conducting studies on the entire dataset, while the second stage entails retrieving preprocessed data with Colab and subjecting it to the aforementioned steps for reanalysis. Initially, analyses were conducted for data cleansing, with each feature's description examined. Due to data skewness, normalization was ensured before visualization. Duplicate records were removed, and subsequently, attributes containing attack types in the dataset were eliminated as they were unnecessary for binary classification problems. To handle missing values in the dataset, PyCaret defaults to using the mean value of the feature, particularly for numerical attributes. It was observed that the median value provided more suitable results than the standard value. Normalization and transformation processes were then carried out.

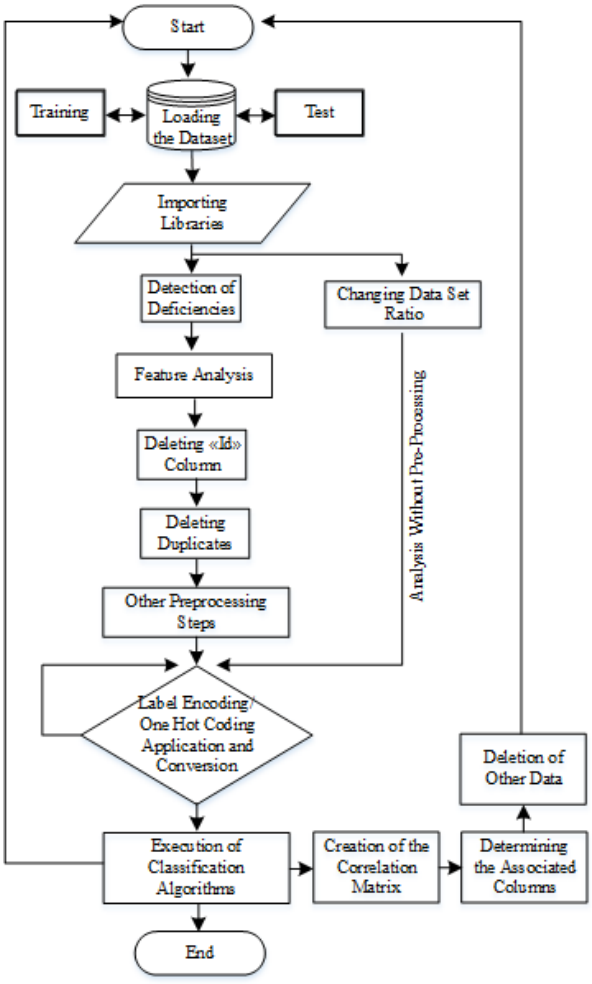

Figure 3.4: Flowchart of the experimental studies on the Colab platform

Data was visualized to observe relationships in the dataset. The Apriori algorithm generated association rules, and sorting was performed based on confidence and lift. After a series of steps, each nominal field in the dataset was encoded using the ”One Hot Encoding” method. Subsequently, all significant parameters and objects were saved to disk to apply the same process to test data. Feature scaling was conducted, followed by classification. Further analyses and experiments were conducted by combining both preprocessed data and the entire dataset using a technique called ”Cross Validation” with various algorithms. This resembles a security analytics component that detects anomalies in data to obtain intelligence for previously unseen threats. The managed rules component enables organizations to respond to the latest attacker techniques almost in real-time with nearly real-time updates for analysts. Figure 3.4 represents the flow of experimental studies on the Colab platform.

4. Results

Log files are the most important data source for ensuring network control. Therefore, the UNSW NB15 Dataset was used, which consists of data obtained from network traffic and various attack activities, in addition to real-life modern scenarios in log files. This dataset contains both normal and abnormal network activities, with 2,540,044 samples. This study used a subset of the dataset containing 257,673 samples, divided into training and test datasets. The training set comprises 175,341 samples, while the test set contains 82,332 samples, encompassing attacks such as Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms, along with normal activities. The environment used to run the application was the free Google Colab version, which utilizes a Tesla K80 GPU.

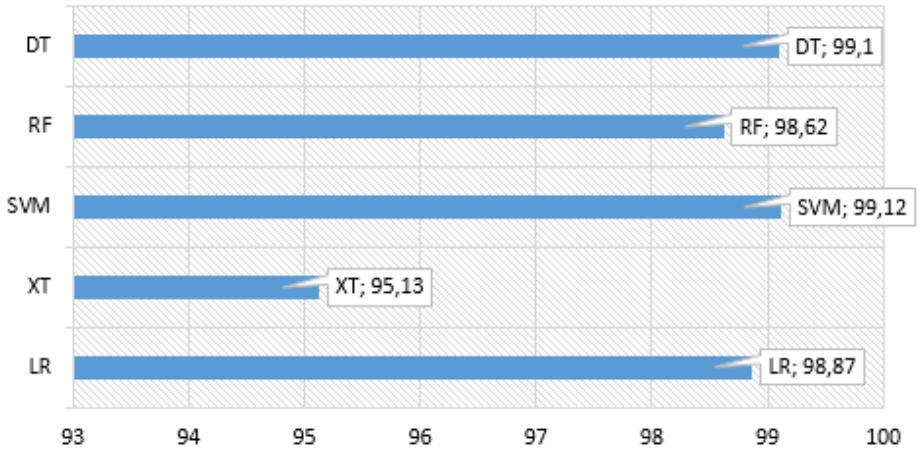

Figure 4.1: Colab Application Classification Results

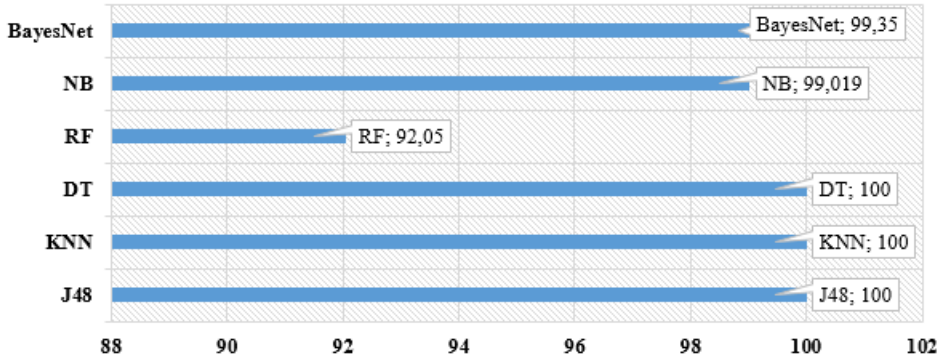

Figure 4.2: Weka Analysis Classification Results

Figure 4.1 presents the results obtained from the application conducted on Colab. Figure 4.2 displays the results of the experiments and applications conducted on Weka. As seen in Table 4.1, the collective test results of the applications show that RF yielded the lowest performance rate at 92.05% in operations performed with Weka. The highest performance was achieved by analyzing DT, J45, and KNN methods in Weka. The highest performance in the Colab application was achieved with the DT method, while the lowest performance was obtained with the implementation of the XT method. Compared to others, SVM in the Colab application produced a higher accuracy score. The PyCaret library facilitated this research to be conducted swiftly and with less effort when it became available. PyCaret’s tuning function also contributed to the study.

Table 4.1: Overall test results for applications

Colab Application Results		Weka Application Results	
Algorithm	**Accuracy(%)**	**Algorithm**	**Accuracy(%)**
RF	98,62	***RF***	92,05
DT	99,10	***DT***	100
LR	98,87	***KNN***	100
XT	95,13	***NB***	99,19
SVM	99,12	***BayesNet***	99,35

The attacks in the UNSW NB15 dataset are highly imbalanced, with some having a probability close to zero. The imbalanced distribution of data has a negative effect on the accuracy of classifying the least frequent attack types, namely Analysis, Backdoor, Shellcode, and Worms, which collectively account for a mere 2.08% of the dataset. Given that these attack classes make up over 68% of the dataset, it is possible to achieve a notably high level of performance in Exploits and Generic classes. The proposed model has demonstrated superior performance to all other Weka models, except for RF. Overall performance experiences a substantial decline when the machine learning detection and classification stages are integrated, and a semi-dynamic hyperparameter tuning strategy demonstrated substantial enhancements in multi-class models with pertinent characteristics and prevalent cybersecurity vulnerabilities. The suggested deep learning classification framework, except for the dynamic hyperparameter tuning strategy, demonstrated substantial enhancements in multi-class models compared to similar deep learning-based network intrusion detection systems (IDSs). Several experiments were conducted to determine the best-performing ML algorithm for each dataset. Additionally, it was found that the performance metrics of the training set results, including accuracy, precision, recall, F1-score, ROC, and CM test results, were higher than the baseline and in the same ranking order. Overall, the models correctly predicted 'normal' data while exhibiting misclassifications when predicting unauthorized entries. An association rule-based technique was applied for feature selection, using the point model selection for each feature. This reduced processing time for determining frequent values and identified the top-ranked features by removing irrelevant or noisy ones, resulting in the input for the machine learning model. The associated cost with false negatives should be very low in this scenario.

Upon examining these results, it is evident that data preprocessing is the first and most crucial step in machine learning model development. Data collected by systems is often inconsistent, requiring preprocessing to prepare the data for visualization and analysis. Each feature was examined, and due to the skewness of the data, normalization was necessary before visualization. Initially, the 'id' attribute was removed from both datasets due to unique numbering in this feature, preventing the identification of duplicate records. This highlights the importance of such cleaning processes. Subsequently, iterative operations identified and removed significant data from the training and test datasets. The ratios in the data set were modified in both Colab and Weka, with the training and test datasets swapped, resulting in better classification results observed in the test dataset.

Encoded variables and extracted features were examined, and correlations among features were identified to reduce the dataset's dimensionality. Before proceeding to data analysis, outliers were removed from the dataset to understand better how the features are distributed. Due to the continuous increase in cyber threats and attacks, optimizing Intrusion Detection System (IDS) capabilities has become critical. ML-based IDSs emerge as novel paradigms that could be utilized to counteract misuse and anomaly attacks. However, the classification accuracy of ML-based IDSs depends on several factors. Identifying these factors aids in developing better ML-based IDSs. The results obtained from the experiments of this study clearly demonstrate that the classification accuracy of ML-based IDSs can be influenced by specific factors related to IDSs. These factors include using the dataset for training and testing, removing outliers and extreme value samples, the addition of mislabeled examples, proper preprocessing, an abundance of experimental studies, implementation of appropriate methods, and utilization of community learning techniques. These factors have various impacts on classification accuracy. In some cases, such as the positive effect of noise on accuracy, these factors often have a negative impact on classification accuracy. However, some of these factors have significantly improved the performance of certain classification algorithms when applied to the UNSW-NB15 dataset. In some cases, these factors have not affected the accuracy of ML-based validation.

5. Conclusion

The increasing importance of information and information systems security with rapidly advancing technology has elevated the necessity of taking measures in system security to the highest level. While tools used in information security systems are diverse and powerful, they may not always succeed in identifying new security vulnerabilities arising from emerging technologies. Hence, a completely secure system can never be guaranteed. Anticipating how attackers might exploit vulnerabilities before malicious attempts and ensuring security accordingly will maximize the effectiveness of the measures taken.

Critical infrastructure systems and evolving technology form the foundation of modern societies and are crucial for maintaining national welfare. These sectors, which generally encompass agriculture, water, electricity grids, transportation, communication, and various public and private sector structures, are essential for daily life. Being interconnected and interdependent complex systems, the malfunction of just one part can lead to the failure of other dependent systems, affecting the entire infrastructure. Furthermore, they play significant roles in state security, the economy, and citizens' well-being. Given these reasons, ensuring the security of critical infrastructure plays a vital role in cybersecurity, and there should be a focus on achieving complete security for such infrastructure in Turkey. Issues related to critical infrastructure protection should not be overlooked in the shadow of cybersecurity efforts, and all necessary work and collaborations must be undertaken. Cybersecurity should be considered as a comprehensive set of measures against cyber attacks. Furthermore, institutions, organizations, and users should develop tools, policies, and practices to protect their assets, and regulatory frameworks should be established for legal documents, electronic media, events, training, and security technologies through national and international cooperation. The growth of critical infrastructure, technological advancements, and cyberspace necessitates education and training. Awareness programs, seminars, conferences, and the inclusion of cybersecurity-related courses in curricula are essential. In recent years, threats to information security have become increasingly targeted towards actively used services, turning threats into a profit model for companies. Exploitation and compromise techniques have become more complex, and analyzing daily access logs and threat monitoring is crucial in information security. SIEM

(Security Information and Event Management) is utilized by many cybersecurity operations centers to enhance monitoring capabilities by correlating events from various security devices into a consolidated framework. SIEM collects security data from network devices, servers, domain controllers, etc., stores and normalizes this data to detect trends and threats, and applies analytical methods to these data. It reports security events to an incident response team, matches specific rule sets to alert and identify security issues, and serves as a data collector, monitor, and reporting system. SIEM primarily focuses on ensuring the three main concepts of information security. Considering the rapid changes in the cybersecurity landscape, organizations need solutions that can detect and respond to known and unknown security threats, which may involve using SIEM products or developing in-house custom software. In this study, first and foremost, the definition and details of critical infrastructure are presented; the aim is to create awareness of the need to take measures to protect these structures. Additionally, the relationship between critical infrastructure and SIEM, which can be considered a whole, and the concept of software engineering as necessary for technological developments are discussed; information about software engineering at national and international levels is provided. Recent studies involving critical infrastructure, cyber security policies, and SIEM are researched, and comprehensive academic and administrative works are included. In the practical part of the study, an application is developed using real-life modern scenarios and data obtained from network traffic and various attack activities from the UNSW NB15 Dataset. The developed application analyzes 10 categories, including "Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms," as nine types of attacks and Normal activities. The study is divided into two main parts. The first step is conducted on Colab, followed by experimental studies on Weka. The Weka applications are divided into two parts: initially working with the entire dataset and then analyzing the preprocessed data with Colab using the aforementioned steps. The results of all these experimental studies and applications are presented in Section 4. Initially, analyses were conducted for data cleansing, descriptions of each feature were examined, and normalization was ensured before visualization due to skewed data. Duplicate records were removed, and attributes containing attack types in the dataset were removed as they were unnecessary for binary classification problems. PyCaret defaults to using the mean value of a feature in case of missing values for numerical features. The median value was observed to provide better results than the standard deviation. Normalization and transformation processes were performed. Data were visualized to observe relationships, association rules were created using the apriori algorithm, and ranking was done based on confidence and lift. Each was encoded using the "One Hot Encoding" method for nominal fields in the dataset. Subsequently, all significant parameters and objects were saved to disk to apply the same process to test data. Feature scaling was performed, followed by classification using LR, XT, SVM, RF, and DT methods. Later, analyses and experiments were conducted again with KNN, BayesNet, NB, DT, and RF algorithms using preprocessed data, and the entire dataset was combined through "Cross-validation." In the application developed with Colab, RF achieved 98.62%, DT 99.10%, LR 98.87%, XT 95.13%, and SVM 99.12% accuracy. Subsequently, in Weka experiments, RF achieved 92.05%, DT 100%, KNN 100%, J48 100%, NB 99.19%, and BayesNet 99.35% accuracy. Based on all these steps and studies, governments, institutions, organizations, and users must develop tools, policies, and practices to protect their assets. Legal documents, electronic media documents, events, training, and security technologies under national/international cooperation regulations are crucial. Advancements in information technologies have emphasized logical security alongside the physical security of systems due to their impact on infrastructures and their interrelationships and dependencies. This is because critical infrastructure IT (Information Technology) systems are often controlled and monitored through computer systems. Since computer systems use the TCP/IP protocol suite, they are exposed to security risks whether connected to the internet or not. In future work, the study conducted on Colab will be developed into a system capable of real-time data analysis and attack detection. A system with intensive reporting and graphing capabilities would benefit operations personnel. Additionally, storing datasets ourselves with a system to be established would be useful for more realistic analysis and development. The planned work also includes real-time packet analysis and suspicious packet detection in the system, along with analyzing and classifying logs and different types of log files based on the SIEM logic using methods beyond classification algorithms.

Article Information

Acknowledgements: The authors would like to express their sincere thanks to the editor and the anonymous reviewers for their helpful comments and suggestions.

Author's contributions: All authors contributed equally to the writing of this paper. All authors read and approved the final manuscript.

Conflict of interest disclosure: No potential conflict of interest was declared by the authors.

Supporting/Supporting organizations: No grants were received from any public, private or non-profit organizations for this research.

Ethical approval and participant consent: It is declared that during the preparation process of this study, scientific and ethical principles were followed and all the studies benefited from are stated in the bibliography.

Plagiarism statement: This article was scanned by the plagiarism program.

References

[1] Y. Alaca, Yapay bağışıklık sistemleri ile bilgi güvenliği ve olay yönetimi geliştirilmesi, M. Sc. Thesis, Karabük University, 2018.
[2] E. Yüksel, Experimenting, threat detection and SIEM integration with custom created honeypots, M.Sc. Thesis, Ankara Yıldırım Beyazıt University, 2019.
[3] S. İşgüzar, Siber aylaklık davranışlarının bir kamu kurumu özelinde incelenmesi: log analizine dayalı bir çalışma, M. Sc. Thesis, Fırat University, 2020.
[4] F. Akgiş, Anomali tespiti için log analizi, M. Sc. Thesis, İstanbul University-Cerrahpaşa, 2021.
[5] R. Daş, M. Z. Gündüz, *Analysis of cyber-attacks in IoT-based critical infrastructures*, Int. J. Inf. Sec. Sci., **8**(4) (2020), 122-133.
[6] D. Gökçeoğlu, Güvenlik bilgileri ve olay yönetimi (SIEM)/Log korelasyon kurallarının yazılması, Ph. D. Thesis, Fırat University, 2021.
[7] H. N. Yerlikaya, Log analysis of a large scale network by using Elastic Stack, M. Sc. Thesis, Bahçeşehir University, 2020.
[8] S. Yenal, N. Akdemir, *Uluslararası ilişkilerde yeni bir kuvvet çarpani: siber savaşlar üzerine bir vaka analizi*, Cankiri Karatekin Univ. J. Inst. Soc. Sci., **11**(1) (2020), 414-450.

[9] S. Moualla, K. Khorzom, A. Jafar, *Improving the performance of machine learning-based network intrusion detection systems on the UNSW-NB15 dataset*, Comput. Intell. Neurosci., **1** (2021), 5557577.

[10] Z. Zoghi, G. Serpen, G., *UNSW-NB15 computer security dataset: Analysis through visualization*, Secur. Priv., **7**(1) (2024), e331.

[11] A. M. Aleesa, Y. Mohammed, A. A. Mohammed, N. Sahar, *Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques*, J. Eng. Sci. Technol, **16**(1) (2021), 711-727.

[12] G. Kocher, G. Kumar, 2021, *Analysis of machine learning algorithms with feature selection for intrusion detection using UNSWNB15 dataset*, Int. J. Netw. Secur. Appl., **13**(1) (2021).

[13] G. Mahalakshmi, E. Uma, M. Aroosiya, M. Vinitha, *Intrusion detection system using convolutional neural network on UNSW NB15 dataset*, Advances in Parallel Computing Technologies and Applications, 2021.

[14] Abdullah, F. B. Iqbal, S. Biswas, R. Urba, Performance analysis of intrusion detection systems using the PyCaret machine learning library on the UNSW-NB15 dataset, B. Sc. Thesis, Brac University, 2021.

[15] N. Sharma, N. S. Yadav, S. Sharma, 2021, *Classification of UNSW-NB15 dataset using exploratory data analysis using ensemble learning*, EAI Endorsed Trans. Ind. Netw. Intell. Syst., **8**(29) (2021), e4-e4.

[16] M. Sarhan, S. Layeghy, M. Portmann, *Towards a standard feature set for network intrusion detection system datasets,* Mob. Netw. Appl., **27**(1) (2022), 357-370.

[17] Y. Pacheco, W. Sun, *Adversarial machine learning: a comparative study on contemporary intrusion detection datasets*, Proceedings of the 7th International Conference on Information Systems Security and Privacy (ICISSP 2021), (2021), 160-171.

[18] I. F. Kilincer, F. Ertam, A. Sengur, *Machine learning methods for cyber security intrusion detection: datasets and comparative study*, Comput. Netw., **188** (2021), 107840.

[19] G. S. Kushwah, V. Ranga, *Optimized extreme learning machine for detecting DDoS attacks in cloud computing,* Computers & Security, **105** (2021), 102260.

[20] S. Roy, A. Mandal, D. Dey, *Intelligent intrusion detection system using supervised learning*, AIJR Proceedings, (2021), 25-34.

[21] M. Ahsan, R. Gomes, M. Chowdhury, K. E. Nygard, *Enhancing machine learning prediction in cybersecurity using dynamic feature selector*, J. Cybersecur. Priv., **1**(1) (2021), 199-218.

[22] T. S. Pooja, P. Shrinivasacharya, *Evaluating neural networks using bi-directional LSTM for network IDS (intrusion detection systems) in cyber security*, Global Trans. Proc., **2**(2) (2021), 448-454.

[23] S. Thirimanne, L. Jayawardana, P. Liyanaarachchi, L. Yasakethu, *Comparative algorithm analysis for machine learning based intrusion detection system*, 10th International Conference on Information and Automation for Sustainability (ICIAFS), (2021), 191-196.

[24] M. Rani, *Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications*, Multimed. Tools Appl., **81**(6)(2022), 8499-8518.

[25] M. Ozkan-Okay, Ö. Aslan, R. Eryigit, R. Samet, *SABADT: hybrid intrusion detection approach for cyber attacks identification in WLAN*, IEEE Access, **9** (2021), 157639-157653.

[26] R. Sekhar, K. Sasirekha, P. S. Raja, K. Thangavel, *A novel GPU based intrusion detection system using deep autoencoder with Fruitfly optimization*, SN Appl. Sci., **3**(6)(2021), 1-16.

[27] S. U. Yang, 2021, *Research on network malicious behavior analysis based on deep learning*, IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), (2021), 2609-2612.

[28] H. Han, H. Kim, Y. Kim, 2022, *An efficient hyperparameter control method for a network intrusion detection system based on proximal policy optimization*, Symmetry, **14**(1) (2022), 161.

[29] K. M. Al-Gethami, M. T. Al-Akhras, M. Alawairdhi, 2021, *Empirical evaluation of noise influence on supervised machine learning algorithms using intrusion detection datasets*, Secur. Commun. , **2021**(1)(2021), 8836057.

[30] A. Meliboev, J. Alikhanov, W. Kim, *Performance evaluation of deep learning based network intrusion detection system across multiple balanced and imbalanced datasets*, Electronics, **11**(4) (2022), 515.

[31] O. A. El-Sayed, S. K. Fawzy, S. H. Tolba, R. S. Salem, Y. S. Hassan, A. M. Ahmed, A. Khattab, *Deep learning framework for accurate network intrusion detection in ITSs*, 2021 International Conference on Microelectronics (ICM), (2021), 212-215.

[32] S. Kim, L. Chen, J. Kim, *Intrusion prediction using long short-term memory deep learning with UNSW-NB15*, 2021 IEEE/ACIS 6th International Conference on Big Data, Cloud Computing and Data Science (BCD), (2021), 53-59.

[33] Z. Hossain, M. M. R. Sourov, M. Khan, P. Rahman, *Network intrusion detection using machine learning approaches*, Fifth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), (2021), 438-442.

[34] I. Dutt, *Pre-processing of KDD'99 & UNSW-NB network intrusion datasets*, Turk. J. Comput. Math. Educ., **12**(11) (2021), 1762-1776.

[35] S. Kim, L. Chen, J. Kim, *Intrusion Prediction using LSTM and GRU with UNSW-NB15*, 2021 Computing, Communications and IoT Applications (ComComAp), (2021), 101-106.

[36] R. Singh, G. Srivastav, G., *Novel framework for anomaly detection using machine learning technique on CIC-IDS2017 dataset*, 2021 International Conference on Technological Advancements and Innovations (ICTAI), (2021), 632-636.

[37] J. V. V. Silva, N. R. de Oliveira, D. S. Medeiros, M. A. Lopez, D. M. Mattos, *A statistical analysis of intrinsic bias of network security datasets for training machine learning mechanisms*, Ann. Telecommun., **77**(7) (2022), 555-571.

[38] S. Priya, 2021, *Performance analysis comparison on various cyber-attack dataset by relatıng a deep belief network model on an intrusion detectıon system (IDS)*, Inf. Technol. Ind., **9**(3) (2021), 608-613.

[39] J. Man, G. Sun, *A residual learning-based network intrusion detection system*, Secur. Commun. Netw., **2021**(1) (2021), 5593435.

[40] L. Ashiku, C. Dagli, *Network intrusion detection system using deep learning*, Procedia Computer Science, **185** (2021), 239-247.

[41] M. K. Hooshmand, D. Hosahalli, *Network anomaly detection using deep learning techniques*, CAAI Trans. Intell. Technol., **7**(2) (2022), 228-243.

[42] I. E. Kamarudin, M. F. Ab Razak, A. Firdaus, M. I. Jaya, Y. T. Dun, *Performance Analysis on Denial of Service attack using UNSW-NB15 Dataset*, 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), (2021), 423-426.

[43] R. Magán-Carrión, D. Urda, I. Díaz-Cano, B. Dorronsoro, *Improving the reliability of network intrusion detection systems through sataset integration*, IEEE Trans. Emerg. Top. Comput., **10**(4) (2022), 1717-1732.

[44] N. Sharma, S. Yadav, *Ensemble learning based classification of UNSW-NB15 dataset using exploratory data analysis,* 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), (2021), 1-7.

[45] Y. J. Chew, N. Lee, S. Y. Ooi, K. S. Wong, Y. H. Pang, *Benchmarking full version of GureKDDCup, UNSW-NB15, and CIDDS-001 NIDS datasets using rolling-origin resampling*, Inf. Secur. J. Global Perspect., **31**(5) (2022), 544-565.

[46] T. Acharya, I. Khatri, A. Annamalai, M. F. Chouikha, *Efficacy of machine learning-based classifiers for binary and multi-class network intrusion detection*, 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), (2021), 402-407.

[47] G. Dlamini, M. Fahim, *DGM: a data generative model to improve minority class presence in anomaly detection domain*, Neural Comput. Appl., **33**(20) (2021), 13635-13646.

[48] A. Pavlov, N. Voloshina, *Dataset selection for attacker group identification methods*, 30th Conference of Open Innovations Association FRUCT, (2021), 171-176.

[49] H. Güler, Ö. Alpay, *Intrusion detection and classification based on deep learning*, 2021 International Conference on Information Security and Cryptology (ISCTURKIYE), (2021), 40-44.

[50] U. Gürtürk, M. Baykara, M. Karabatak, *Identifying the visitors with data mining methods from web log files*, Int. J. Emerg. Technol. Eng. Res., **5**(3) (2017), 243-249.

[51] U. Gürtürk, Türkiye'nin siber güvenlik politikalarının yazılım mühendisliği açısından değerlendirilmesi ve kritik altyapıların siber saldırılardan korunmasına yönelik olay yönetim sistemi tasarımı, M.Sc. Thesis, İstanbul University-Cerrahpaşa, 2022.