Impact Factor (SJIF): 4.714 E-ISSN: 2583-1615

International Journal of Research in Academic World

Received: 20/January/2023 IJRAW: 2023; 2(2):122-129 Accepted: 23/February/2023

Intrusion Detection using Log Datasets Based on Severity in Machine Learning

[*1]Greshma P Sebastian and [2]Gopika Chingan

[*1]Assistant Professor, Department of Computer Science and Engineering, SCMS School of Engineering and Technology, Ernakulam, Kerala, India.

[2]MCA Student, Department of Computer Science and Engineering, SCMS School of Engineering and Technology, Ernakulam, Kerala, India.

Abstract

Cyber-attacks have become an ongoing challenge for businesses to protect their information from. A solution to this problem is the use of a network security operating system, which monitors logs for unusual traffic and can prevent security incidents. However, these systems generate massive amounts of logs, which require an automated method to identify potential cyber-attacks. One such method is a log-based intrusion detection system, which predicts the presence or absence of attacks by analyzing selected functions from the collected logs. The effectiveness of various machine learning algorithms is tested to determine the best one for intrusion detection.

Key features are identified from logs collected from various sources, and the best machine learning algorithm is selected based on the results of the testing. The goal is to have an effective system that can detect potential attacks in real-time, classify the events and minimize false alarms to improve the overall security operations. In summary, a log-based intrusion detection system is a powerful tool for businesses to safeguard their information from cyber-attacks by using machine learning algorithms to analyze the logs. By testing various algorithms, the best one is selected to ensure the most effective and efficient detection of intrusions.

Keywords: Intrusion detection, machine learning, logs, security

1. Introduction

An intrusion is when a hacker or cracker attempts to gain unauthorized access to a system or network [4]. It is a security event that can lead to a security incident if the intruder is successful. Intrusion detection is the process of identifying these attempts and preventing them from succeeding. One way to improve intrusion detection is by using machine learning models. These models can be trained to recognize unusual behavior and security anomalies, allowing the computer system to improve its ability to detect intrusions.

Machine learning, a subset of Artificial Intelligence, enables computer systems to improve their performance through the use of historical data as input to predict new outcomes, without the need for explicit programming. ML has become one of the most popular AI technologies among firms, institutions, and individuals in the automation market, due to recent advancements in data access and processing capabilities which enables practitioners to achieve valuable results in various domains.

The aim of the project is to reunite log data from servers and then evaluate if there are any security threats. It also helps to take an immediate step for eradicating risk. The risk of attack in the network is classified into 4 classes based on severity. Severity is used to determine the impact and harm caused by security incidents. This classification considers factors such as the type of attack, the possibility of data loss or operational disruption and the potential for financial or reputational damage. Low, medium, high, and critical are the levels of severity commonly used in intrusion detection [1].

Incidents that are considered benign and do not cause major harm to the system or network are classified as low severity. Incidents that have the possibility of causing limited damage, such as operational disruption or data loss, are classified as medium severity. High severity incidents have a high potential for causing significant harm, such as financial loss or compromise of sensitive information. Critical severity incidents are those that have the potential to cause catastrophic damage, such as network compromise or widespread data loss

1.2. Objective

The aim is to create a machine learning model that predict the risk of intrusion on a network from logs datasets and classify security incidents based on the information captured in log files. This includes identifying different types of attacks, such as DoS or data breaches, and determining the severity of these incidents based on their potential impact and harm. Machine learning algorithms can be used to analyze log data and

***Corresponding Author:** Greshma P Sebastian

identify anomalies and patterns that may indicate a security breach. They can also classify different types of attacks and evaluate their severity. By using machine learning, the process of intrusion detection becomes more accurate and efficient, reducing the need for manual analysis and classification. The ultimate objective of an intrusion detection system is to quickly identify intrusions, classify events, and minimize false alarms for improved security operations.

1.3. Scope

Intrusion detection using machine learning encompasses a variety of security incidents and attack types, which can be adapted to the specific needs of an organization. These include network-based attacks, host-based attacks, insider threats, and advanced persistent threats (APTs).

Machine learning algorithms are trained to identify and categorize security incidents, and they can also determine the severity of an attack based on pre-defined criteria. The capabilities of the intrusion detection system can be broadened by adding more data sources and refining the algorithms to increase accuracy. The system can also be combined with other security tools and systems, such as firewalls, intrusion prevention systems, and security information and event management (SIEM) systems [3].

1.4. Organization of Report

The report is organized into five chapters. The first chapter of the report deals with the general background, objective and scope of the project. The second chapter contains various literature reviews related to the project. The detailed architecture and working of the proposed system is discussed in the third chapter. The fourth chapter contains the experimental results and discussions. The conclusions are summarized in the final chapter. The future scope of the given project is also added in the last chapter. Finally the references are given in the last pages.

2. Literature Survey

Doaa Hassan [1] have proposed a presents a new method using data mining to predict the danger level of an attack by analyzing alerts from a NIDS log. This method uses the type of suspicious traffic to automatically determine the attack severity, saving time and effort using the machine learning classifiers such as Random Forest, Naïve Bayes, and Support Vector Machine etc. Experiment results show that the approach is highly effective, with over 99.2% accuracy in identifying attack severity.

Revathi. S and Malathi. A [2] present an in-depth analysis of the NSL-KDD dataset for intrusion detection using various machine learning techniques. The study employs methods such as decision tree, random forest, k-NN, and SVM to detect intrusions in the NSL-KDD dataset, which is widely recognized as a benchmark in intrusion detection research. The paper evaluates the performance of these machine learning methods on the dataset and provides insights into their effectiveness for intrusion detection, serving as valuable information for future research in this field.

X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu [3] introduced a machine learning-based intrusion detection model that employs an adaptive ensemble technique. The method combines multiple classifiers and adjusts the weight of each classifier based on its performance. The goal of the adaptive ensemble strategy is to enhance the overall performance of the intrusion detection system by mitigating the impact of underperforming classifiers. The study evaluates the performance of the proposed method using the KDD Cup 99 dataset and shows that it outperforms single classifier methods. The results demonstrate that the adaptive ensemble model significantly improves the performance of the intrusion detection system with an overall accuracy of 99.4% and a detection rate of 99.2%. Additionally, the research reveals that the adaptive ensemble model can maintain high performance even when the number of features is reduced by a factor of 10.

The research article [6] offers an overview of the use of machine learning algorithms in intrusion detection systems (IDSs). The authors of the study conducted a thorough examination of various machine learning techniques applied to intrusion detection, including popular methods such as decision trees, artificial neural networks, and support vector machines. The study evaluates the strengths and weaknesses of each technique and its suitability for different intrusion detection problems. The conclusion of the paper highlights that while machine learning methods have proven effective in detecting security threats, the performance of these techniques greatly relies on the quality and diversity of the data used for training the models.

3. Proposed System

The proposed system for intrusion detection using log datasets based on severity in machine learning is a comprehensive approach that employs multiple steps to effectively detect and classify intrusions. Initially, the log data undergoes preprocessing to identify relevant features and transform them into a format that is compatible with machine learning models. This step is crucial as it makes sure that the data is in the correct format and contains the necessary information for the models to make accurate predictions.

Once the data is preprocessed, it is classified into different severity levels. This classification could be based on various factors such as the type of intrusion, the level of damage caused, and the level of risk to the organization. This step helps to prioritize the incidents and take necessary action accordingly. The next step is to train machine learning models using the classified log data. This step is essential as it allows the models to learn the patterns of intrusion activities for each severity level. Different supervised techniques such as Random Forest, Decision Tree, SVM etc.

The trained model is then deployed to continuously monitor and analyze incoming log data. It classifies the incoming data into different severity levels, which can then be used to trigger appropriate actions.

In summary, the proposed system for intrusion detection using log datasets based on severity in machine learning is a comprehensive approach that involves preprocessing, classification, training, and deployment of machine learning models to effectively detect and classify intrusions. It also takes into account the severity level of the intrusion to trigger appropriate actions.

3.1. Architecture

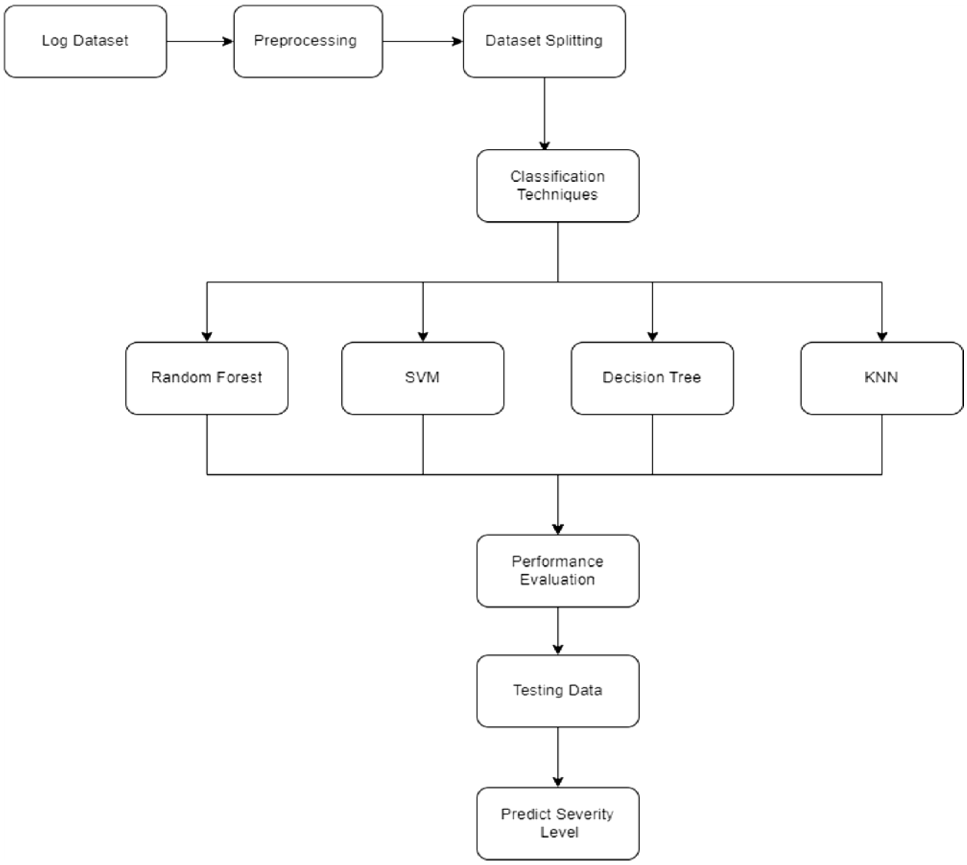


Fig 1: High level Architecture

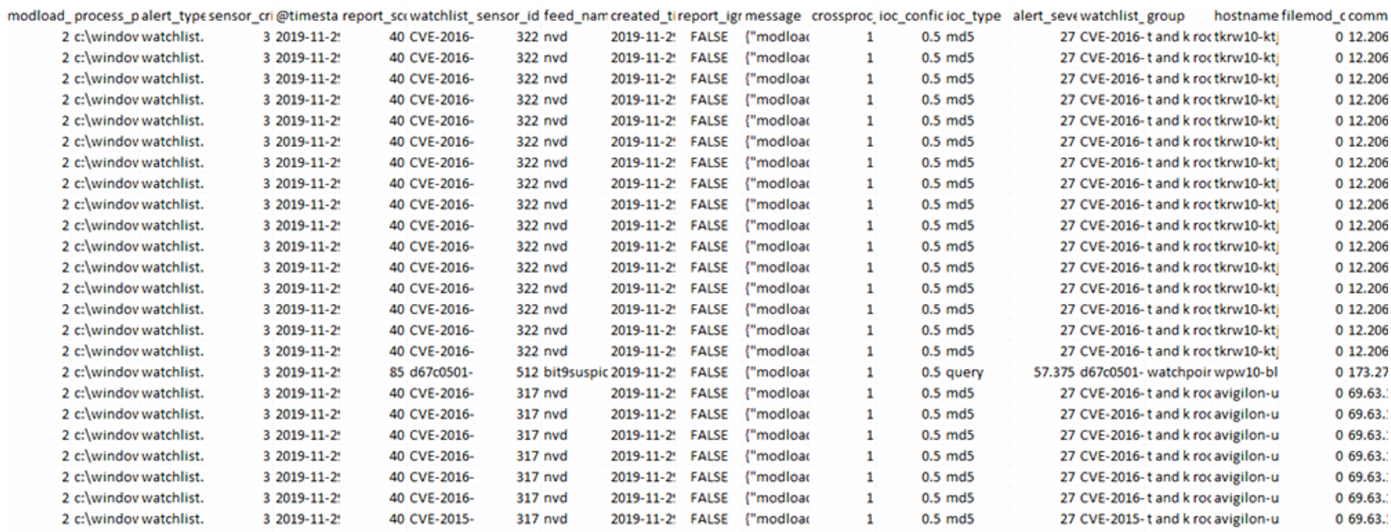
3.2. Implementation of System

The implementation of this project utilized Python 3.6.4 and important libraries such as scikit-learn, pandas, and matplotlib. The dataset was obtained from Ciber Digital Consultants, containing 4 distinct intrusion classes and was divided into train and test datasets. The train dataset was used for training while the test dataset was utilized to evaluate the model's performance. We sourced the intrusion detection dataset from the log dataset and applied Machine Learning algorithms such as decision tree, SVM, random forest, and KNN.

3.3. Dataset

The log data from Ciber Digital Consultants is stored in a json file and includes details of all events within their systems and networks. The data, consisting of 2500 points, will be converted from json to csv(comma separated file) format.

This dataset contain 47 features which include mod load count, process path, alert type, sensor criticality, report score, alert type etc.



modload_	process_p	alert_type	sensor_cri	@timesta	report_sc	watchlist_	sensor_id	feed_nam	created_ti	report_igr	message	crossproc_	ioc_confic	ioc_type	alert_seve	watchlist_	group	hostname	filemod_c	comm
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	322	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	tkrw10-ktj	0	12.206
2	c:\window	watchlist.	3	2019-11-2!	85	d67c0501-	512	bit9suspic	2019-11-2!	FALSE	{"modloac	1	0.5	query	57.375	d67c0501-	watchpoir	wpw10-bl	0	173.27
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	317	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	avigilon-u	0	69.63.:
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	317	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	avigilon-u	0	69.63.:
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	317	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	avigilon-u	0	69.63.:
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	317	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	avigilon-u	0	69.63.:
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	317	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	avigilon-u	0	69.63.:
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2016-	317	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2016-	t and k roc	avigilon-u	0	69.63.:
2	c:\window	watchlist.	3	2019-11-2!	40	CVE-2015-	317	nvd	2019-11-2!	FALSE	{"modloac	1	0.5	md5	27	CVE-2015-	t and k roc	avigilon-u	0	69.63.:

Fig 2: Log Dataset

From the dataset 7 important features are selected from related papers [1, 4, 7]. This include

- **Process Path:** A record of the steps or stages an event or process goes through, which provides insight into the flow of events and can be used for problem diagnosis and performance monitoring.
- **Alert Type:** A classification of log events into specific categories (e.g. error, warning, informational, critical) based on event severity for easier analysis, troubleshooting and monitoring.
- **Timestamp:** Indicate the occurrence of an event, providing a reference for event analysis and investigation of cause-and-effect relationships between events.
- **Watch List ID:** is a unique identifier used to group log events of interest for real-time monitoring and notifications, helping organizations identify security threats, track critical processes and investigate performance issues efficiently.
- **Tags:** Metadata labels that categorize and describe log events, providing additional context and allowing for easier organization, search, and analysis of log data.
- **Report Score:** Is a numerical value indicating the importance or severity of a log event, used to prioritize and categorize events for quick identification and response.
- **Alert Severity:** A classification of log events based on level of importance or criticality, used to quickly identify and respond to important events.

3.4. Data Preprocessing

Data preprocessing is a crucial step in the intrusion detection from a log dataset using severity in machine learning

3.4.1. Data Type Conversion

The data which is in the form of object data type is converted into integer data type using label encoder. The label encoder can be used to convert categorical values in the log data into numerical codes, enabling the machine learning algorithm to identify patterns and anomalies that may indicate an intrusion. The use of label encoder helps to reduce the dimensionality of the log data and make the data more manageable for machine learning algorithms. Additionally, it can help to improve the accuracy and performance of the machine learning model, as well as prevent over fitting and improve generalization.

3.4.2. Balancing

The dataset obtained was imbalanced. So data is balanced by oversampling the dataset. Balancing a dataset using oversampling involves increasing the number of instances in the minority class to make it more proportional to the majority class. This can be achieved by either duplicating existing instances of the minority class or generating new synthetic samples.

The steps to balance a dataset using oversampling are:

- **Identify the Minority Class:** Determine which class has fewer instances in the dataset.
- **Duplicate Instances:** Choose instances of the minority class and repeat them to increase their representation in the dataset.
- **Synthetic Sample Creation:** Utilize methods such as SMOTE (Synthetic Minority Oversampling Technique) to produce synthetic samples for the minority class.
- **Merge Original and Balanced Datasets:** Combine the original dataset with the newly generated or duplicated instances of the minority class.

3.4.3. Data Splitting

In Machine Learning, evaluating a model's accuracy is commonly done through splitting the dataset into two parts, Train and Test. The Train set is used to train the model and the Test set is used to assess its performance after training is completed. This approach of dividing the dataset is widely accepted. The objective of the train/test split is to have an impartial evaluation of the model's performance on data it hasn't seen before. A common split ratio is 80% training and 20% testing, ensuring that the test set is significant enough to provide a trustworthy evaluation, but not so large as to over-represent the data.

3.5. Machine Learning Algorithm Selection

The selection of the appropriate Machine Learning algorithm for intrusion detection in log datasets can be made from a range of options such as Random Forest, Support Vector Machines (SVM), K-Nearest Neighbors (KNN) and Decision Tree. The choice is dependent on the characteristics of the dataset and the computational resources at hand. The prediction results are evaluated based on their accuracy and error values.

3.5.1. Random Forest Classifier

The Random Forest algorithm is a well-known method in machine learning that can be applied to detect intrusions in log data. This algorithm combines multiple decision trees, and each tree predicts the target variable. In this scenario, the log data is utilized for training the model, and the severity of each intrusion is the target variable to be predicted. As it can manage large datasets and complex feature relationships, Random Forest is a suitable option for intrusion detection. However, it is crucial to preprocess the log data carefully and choose relevant features wisely to prevent over fitting and enhance the accuracy of the model. The algorithm starts by partitioning the training dataset into random subsets, and each decision tree is trained on a different subset. To build each tree, a random set of features is selected and used to split the data into branches, until the leaves contain pure or nearly pure groups of data points. This results in a tree structure that separates the data based on feature values. For a new sample, each tree in the forest casts a vote for the target class, and the class with the most votes is considered the prediction of the Random Forest. By combining the outputs of multiple trees, the algorithm reduces the variance of individual trees, resulting in more accurate and robust predictions. To assess the accuracy of the model, an out-of-bag evaluation is used, where each tree is tested on data not used in its training, and the accuracy is calculated by averaging the results of all trees. The Random Forest algorithm is a fast and effective tool for classification and regression tasks that can handle noisy or complex data and handle missing data or irrelevant features effectively.

3.5.2. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning technique that can be utilized for intrusion detection by analyzing log datasets. The goal of the SVM is to find a boundary, known as a hyper plane, that separates normal from abnormal behavior. The log data can be utilized to train the SVM model to recognize the difference between normal and abnormal behavior, and the severity of each intrusion can be predicted as the target variable. SVM is useful in high-dimensional spaces and can handle non-linear relationships between features, making it an appropriate choice for

intrusion detection. However, proper preprocessing of the log data and careful selection of relevant features are crucial to avoid over fitting and improve the accuracy of the SVM model.

3.5.3. K Nearest Neighbor (KNN) Algorithm

The K-Nearest Neighbors (KNN) algorithm is a suitable choice for intrusion detection with log datasets in machine learning, due to its non-parametric and lazy learning approach that relies on the similarity of data points to make predictions. The log data can be trained using the KNN model to differentiate between normal and abnormal behavior, with the severity of each intrusion being predicted as the target variable. When making a prediction, the KNN algorithm finds the K closest data points to the new sample and classifies the new sample based on the majority class among the neighbors. The performance of the K-Nearest Neighbors (KNN) algorithm in intrusion detection can be optimized by adjusting the value of its adjustable hyper parameter, K, which represents the number of nearest neighbors. KNN is straightforward to implement and can handle non-linear relationships between features, but it can be computationally intensive for large datasets..

3.5.4. Decision Tree

The Decision Tree algorithm can be applied to intrusion detection using log datasets in machine learning. It is a tree-structured model that splits the data into smaller subgroups based on feature values. The aim is to separate the classes by splitting the data effectively. In this scenario, the log data can be trained using the Decision Tree model to distinguish normal and abnormal behavior, with the severity of each intrusion serving as the target variable to be predicted by the model. The Decision Tree approach begins by selecting the feature that provides the best class separation and creating a node based on that feature. The process is repeated for the subgroups formed by the split until a stopping condition is met, such as a maximum depth or a minimum number of samples in a group. The final outcome is a tree structure that maps input features to target predictions. Decision Trees are simple to understand and can handle non-linear relationships between features.

3.6. Evaluation

Evaluation is the assessment of a machine learning model's performance on a set of data, aimed at determining the model's ability to differentiate between normal and intrusive behavior as reflected in log data.

3.6.1. Confusion Matrix

A confusion matrix is a tool used to evaluate the accuracy of a machine learning classification model. In the context of intrusion detection from a log dataset using severity, the matrix displays the number of correct and incorrect predictions made by the model. The matrix consists of four categories: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN).

True Positive: TP represents the number of actual positive instances that were correctly identified as positive by the model.

False Positive: FP represents the instances that were incorrectly predicted as positive. True Negative: TN represents the actual negative instances that were correctly classified as negative.

False Negative: FN represents the actual positive instances that were misclassified as negative by the model.

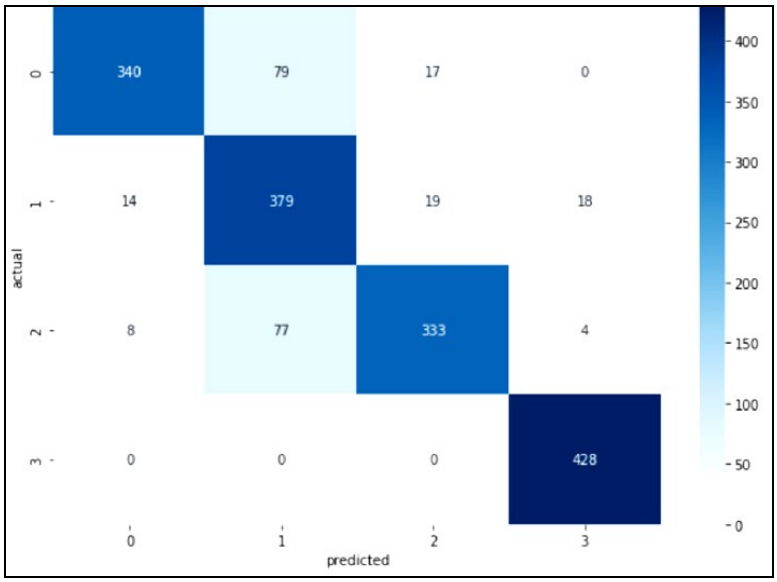


Fig 3: Random Forest Classifier Confusion Matrix

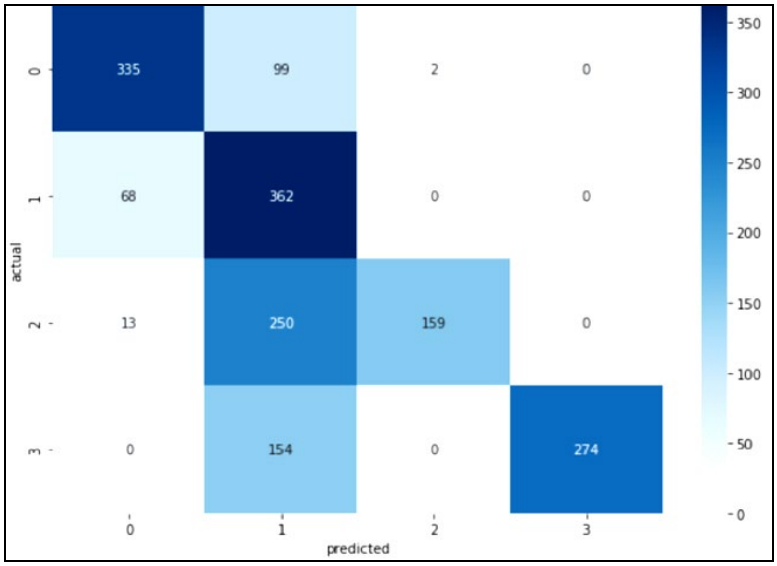


Fig 4: SVM Classifier Confusion Matrix

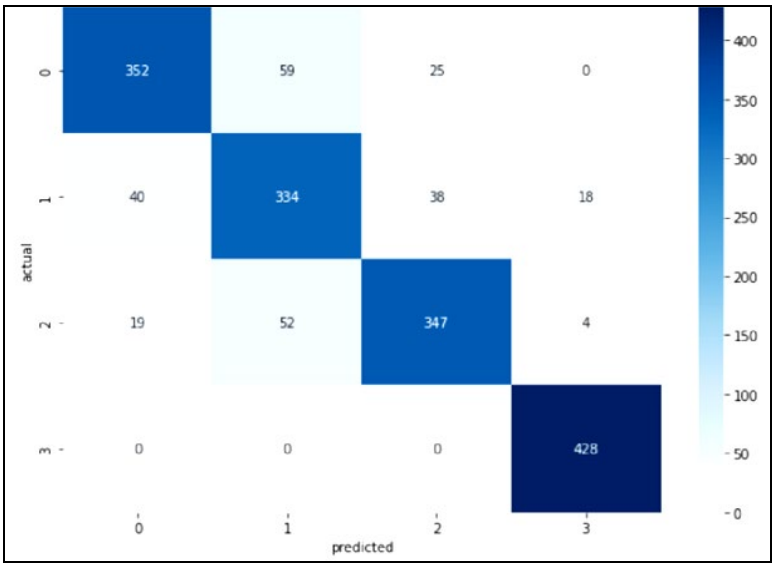


Fig 5: KNN Classifier Confusion Matrix

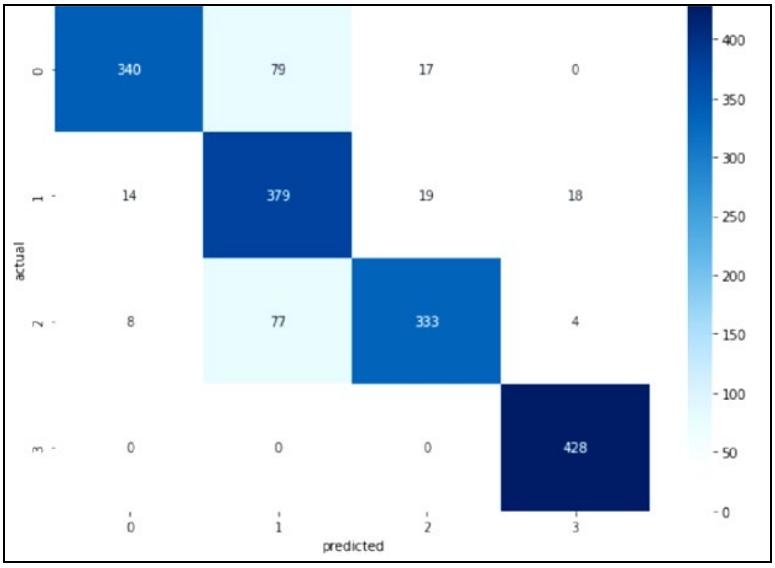


Fig 6: Decision Tree Classifier Confusion Matrix

3.6.2. Classification Report

The performance of an intrusion detection model based on log data with severity can be evaluated using a classification report. This report provides important metrics such as precision, recall, F1-score and support that determine the accuracy of the model in classifying instances as normal or intrusive.

Precision: Measures the proportion of true positive predictions made by the model to all positive predictions.
Precision = True Positives/(True Positives + False Positives)
Recall: Calculates the proportion of true positive predictions made by the model to all actual positive instances.
Recall = True Positives/(True Positives + False Negatives)
F1 Score: It is the harmonic mean of precision and recall, which balances both metrics.
F1-Score = 2 * (Precision * Recall)/(Precision + Recall)
Accuracy: Is the ratio of correct predictions made by the model to the total number of predictions.
Accuracy = (True Positives + True Negatives)/Total Predictions

Table 1: Random Forest Classification Report

Classification Report of Random Forest				
	Precision	**Recall**	**F1-score**	**Support**
0	0.94	0.78	0.85	436
1	0.71	0.88	0.79	430
2	0.90	0.79	0.84	422
3	0.95	1.00	0.97	428
Accuracy			0.86	1716
Macro avg	0.88	0.86	0.86	1716
Weighted avg	0.88	0.86	0.86	1716

Table 2: SVM Classification Report

Classification Report of SVM				
	Precision	**Recall**	**F1-score**	**Support**
0	0.81	0.77	0.79	436
1	0.42	0.84	0.56	430
2	0.99	0.38	0.55	422
3	1.00	0.64	0.78	428
Accuracy			0.66	1716
Macro avg	0.80	0.66	0.67	1716
Weighted avg	0.80	0.66	0.67	1716

Table 3: KNN Classification Report

Classification Report of knn				
	Precision	**Recall**	**F1-score**	**Support**
0	0.86	0.81	0.83	436
1	0.75	0.78	0.76	430
2	0.85	0.82	0.83	422
3	0.95	1.00	0.97	428
Accuracy			0.85	1716
Macro avg	0.85	0.85	0.85	1716
Weighted avg	0.85	0.85	0.85	1716

Table 4: Decision Tree Classification Report

Classification Report of Decision Tree				
	Precision	**Recall**	**F1-score**	**Support**
0	0.94	0.78	0.85	436
1	0.71	0.88	0.79	430
2	0.90	0.79	0.84	422
3	0.95	1.00	0.97	428
Accuracy			0.86	1716
Macro avg	0.85	0.86	0.86	1716
Weighted avg	0.85	0.86	0.86	1716

Results and Discussion

The intrusion detection using ML project has successfully created models that can accurately identify network security threats. These models were made using machine learning techniques like Random Forest Classifier, K Nearest Neighbor, Support Vector Machine, and Decision Tree. The models were trained on 2500 log data points, containing both normal and malicious activities. The models analyze log inputs to detect signs of intrusion and classify the activity as such. The models were tested and evaluated, with Random Forest and decision tree showing the highest accuracy of 85%. This algorithm is now utilized in real-time to detect potential security threats in a network.

Table 5: Result of the Experimental Analysis of the Proposed System

No	**Model**	**Accuracy**
1	Random Forest	86%
2	Decision Tree	86%
3	K Nearest Neighbour	85%
4	Support Vector Machine	66%

For better understanding a user interface is created using Gradio. Gradio is a tool for building graphical interfaces for machine learning models. It can be used to create an interface for an intrusion detection system using a log dataset and severity as the input features. The interface would allow the user to input log data and the severity of the intrusion, and the machine learning model would make a prediction of whether or not an intrusion has occurred based on the input data. The result of the prediction could be displayed in the interface, along with any other relevant information, such as a confidence score or recommended actions to take in response to the intrusion.

Fig 7: User Interface for Intrusion Detection

Conclusion & Future Scope

The objective of an Intrusion Detection System is to detect and prevent malicious activities and attacks within a network while minimizing false positive alerts. By incorporating machine learning algorithms, the system produces precise, sophisticated and reliable results. The system also assesses the accuracy of detected attacks by evaluating the performance of various machine learning algorithms. With the widespread usage of technology, there is an increasing amount of data that requires secure storage and processing. Security is a paramount concern for users and a secure system guarantees privacy. The higher the security level of a system, the more trustworthy it is perceived to be. An effectively designed Intrusion Detection System that provides adequate protection for user data is considered a successful system.

The future holds immense promise for utilizing machine learning in intrusion detection by analyzing log data. This is due to advancements in ML algorithms, the growing availability of big data technologies, the importance of log data in detecting security threats, particularly in the cloud and IoT, the emergence of new data analysis techniques, and the growing demand for automated and cost-effective security solutions. The significance of machine learning in intrusion detection using log datasets is expected to increase in the future. It would be beneficial to develop a method or tool that can effectively extract relevant information from log files and prioritize it based on whether the log was generated from successful hardware or software tests. This work has the potential to significantly enhance the overall efficiency and accuracy of intrusion detection.

References

1. Hassan D. Mining intrusion detection Alerts for predicting severity of detected attacks. In 2015 11th International Conference on Information Assurance and Security (IAS). IEEE, 2015, 38-43.
2. Revathi S, Malathi A. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT).* 2013; 2(12):1848-1853.
3. Gao X, Shan C, Hu C, Niu Z, Liu Z. An adaptive ensemble machine learning model for intrusion detection. IEEE Access. 2019; 7:82512-82521.
4. Liu H, Lang B. Machine learning and deep learning methods for intrusion detection systems: A survey. Applied sciences. 2019; 9(20):43-96.
5. Sommer R, Paxson V. Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE symposium on security and privacy. IEEE, 2010, 305-316.
6. Haq NF, Onik AR, Hridoy MAK, Rafni M, Shah FM, Farid DM. Application of machine learning approaches in intrusion detection system: a survey. IJARAI *International Journal of Advanced Research in Artificial Intelligence*. 2015; 4(3):9-18.
7. Sabhnani M, Serpen G. Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection context. In MLMTA, 2003, 209-215.
8. Halimaa A, Sundarakantham K. Machine learning based intrusion detection system. In 2019 3rd International conference on trends in electronics and informatics (ICOEI. IEEE), 2019, 916-920.
9. Zaman M, Lung CH. Evaluation of machine learning techniques for network intrusion detection. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2018, 1-5.
10. Kilincer IF, Ertam F, Sengur A. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. Computer Networks. 2021; 188:107-840.
11. Athira AB, Pathari V. "Standardisation and Classification of Alerts Generated by Intrusion Detection Systems", IJCI, *International Journal on Cybernetics.* 2016, 5(2).
12. Johansson Daniel, Andersson Par, "Intrusion Detection Systems with Correlation Capabilities".Yasm Curt, "Prelude as a Hybrid IDS Framework", March, 2009.
13. Kumar Vinod, Sangwan Prakash Om, "Signature Based Intrusion Detection System Using SNORT", IJCAIT, *International Journal of Computer Applications & Information Technology*, 2012, I(III).
14. Singh Deepak Kumar, Gupta Jitendra Kumar, "An approach for Anomaly based Intrusion detection System using SNORT", IJSER, *International Journal of Scientific & Engineering Research*, 2013, 4(9).
15. Vijayarani S, Maria Sylviaa S. "Intrusion Detection System-A Study", IJSPTM, *International Journal of Security, Privacy and Trust Management*. 2015; 4(1):31-44.
16. Yang Guangming, Chen Dongming, Xu Jian, Zhu Zhiliang, "Research of Intrusion Detection System Based on Vulnerability Scanner", ICACC, Advanced Computer Control, 2010.
17. Chakraborty Nilotpal, "Intrusion Detection System and Intrusion Prevention System: A Comparative Study", IJCBR, *International Journal of Computing and Business Research*, 2013, 4(2).
18. Kothari Pravin, "Intrusion Detection Interoperability and Standardization", 2002.
19. TIMOFTE Jack, "Intrusion Detection using Open Source Tools", Revista Informatica Economică nr. 2008; 2(46):75-79.