International journal of engineering science and management (IJESM).
ISSN: XXXX-XXXX.
https://doi.org/XX.XXXXX/XXXXXXXX/IJESM-XXXXXXXXX

Volume: 24, Issue 1, 118-130, October 2024.
© 2024 Thanvish Nandan Research Group®

Enhancing SDN Security: Machine Learning-based Detection of Network Intrusion Attacks

M. kalidas[1], D. Venkata Sumanth[2]
[1]Assistant Professor, Department of MCA, Chaitanya Bharathi Institute of Technology(A), Gandipet, Hyderabad, Telangana State, India.
[2]MCA Student, Chaitanya Bharathi Institute of Technology(A), Gandipet, Hyderabad, Telangana State India.

Received: 17 September 2024 Revised: 08 September 2024 Accepted: 1 October 2024 Published: 4 October 2024

ABSTRACT:
The rapid proliferation of Software-Defined Networking (SDN) has introduced numerous advantages for managing complex networks. However, this paradigm shift has also escalated the risks of network intrusion attacks, demanding advanced security measures to safeguard critical infrastructures. Traditional security approaches often fall short in addressing the dynamic and sophisticated nature of modern cyber threats, necessitating the integration of intelligent systems for enhanced detection and mitigation capabilities. This paper investigates the application of Machine Learning (ML) techniques to bolster the security framework of SDN against various forms of network intrusion attacks. By leveraging the inherent capabilities of ML algorithms, such as deep learning and anomaly detection, the proposed model aims to fortify the SDN architecture by continuously monitoring network traffic patterns, identifying anomalies, and predicting potential intrusion events with greater accuracy and efficiency. Through a comprehensive analysis of real-time network data and simulations, this research demonstrates the efficacy of the ML-based approach in detecting both known and unknown intrusion attempts, thereby mitigating potential threats before they can compromise the network's integrity. Furthermore, the integration of ML-driven security measures fosters adaptability to evolving attack strategies and enables proactive responses to emerging threats, ensuring a robust and resilient SDN infrastructure. The findings underscore the significance of integrating intelligent security mechanisms within SDN environments to proactively detect, prevent, and mitigate intrusion attacks. By harnessing the power of Machine Learning, this study contributes to the development of a more secure and reliable SDN ecosystem, fostering trust and confidence in the integrity of modern network infrastructures. A DDoS attack occurs when multiple systems collaborate to target a specific host simultaneously. In SDN, the control layer software, situated between the application and infrastructure layers, governs the devices within the infrastructure layer. To identify and thwart malicious traffic, we propose a machine learning-based approach utilizing Random Forest, AdaBoost, CatBoost, and XGBoost algorithms in this study. Our experimental results demonstrate that the Random Forest, CatBoost, and XGBoost algorithms exhibit superior detection rates and accuracy, showcasing their efficacy in fortifying SDN against potential security breaches.

KEYWORDS: machine learning, ddos attacks, Random Forest, AdaBoost, CatBoost, XGBoost, Streamlit, Machine Learning.

1. INTRODUCTION:

The rapid growth of Internet-connected devices in various sectors such as agriculture, healthcare, and commerce has engendered a range of practical solutions. However, the increased need for connectivity has put pressure on established network designs. The Software Defined Network (SDN) architecture was developed to overcome these difficulties, which effectively separates the user plane and control plane, has been proposed. This architecture not only simplifies network management but also enhances overall network efficiency. Despite its advantages, the SDN architecture remains susceptible to various security threats, such as Secure Shell (SSH) brute force attacks, which can be initiated by potential attackers, posing substantial security risks. The real-time identification and mitigation of concurrent attempts by the network administrator can be challenging [1].

Consequently, configuring the SDN controller with appropriate security policies, akin to firewall rules, becomes imperative. Formulating these security rules, however, presents its own challenges, as they must bar unauthorized nodes or attackers while facilitating legitimate user access. It is possible to identify malicious individuals by analyzing their behavior, which often follows recognizable patterns such as coordinated attacks & the usage of shared password dictionaries. Machine learning-based approaches have shown significant promise in discerning and classifying users, leveraging specific characteristics to differentiate malicious actors from legitimate users. Software Defined Networking (SDN) has emerged as a transformative technology, revolutionizing the way networks are managed and operated. By decoupling the control and data planes, SDN offers unprecedented flexibility, scalability, and efficiency in handling complex network infrastructures. However, the heightened programmability and dynamic nature of SDN networks have exposed them to a myriad of security threats, including sophisticated network intrusion attacks that can jeopardize the integrity and functionality of the entire network. To mitigate these evolving security challenges, the integration of advanced security mechanisms has become imperative to ensure the resilience and robustness of SDN architectures [2].

In recent years, the synergy between SDN and machine learning has garnered significant attention as a promising approach to bolster the security posture of SDN environments. By harnessing the capabilities of machine learning algorithms, SDN networks can proactively detect, analyze, and respond to potential network intrusion attacks, thereby fortifying the overall security framework and mitigating potential vulnerabilities. This paper delves into the critical intersection of SDN and machine learning, focusing specifically on the implementation of machine learning-based techniques for the detection and prevention of various forms of network intrusion attacks. Through a comprehensive analysis of the challenges posed by contemporary security threats in SDN environments, this research endeavors to propose a novel approach that leverages the power of machine learning algorithms to enhance the security resilience of SDN infrastructures [3].

The integration of machine learning-based detection mechanisms is poised to revolutionize the landscape of SDN security, empowering network administrators and operators to proactively safeguard their networks against ever-evolving cyber threats [4].

2. LITERATURE SURVEY:

The utilization of machine learning strategies to address Attacks on Software Defined Networks (SDNs), Ashraf & colleagues looked into the full spectrum, including invasions and DDoS attacks. The paper examined the application of genetic algorithms, neural networks, Bayesian networks, fuzzy logic, and support vector machines in detecting anomalies within SDNs. The strengths and limitations of these approaches for detecting irregularities were extensively explored [5].

In their work, provide a comprehensive guide on Using Software-Defined Networks (SDNs) to bolster network security and advocate for SDNs as a security-as-a-service (SaaS) solution. The survey compiles a range of challenges and potential solutions from the literature to address network threats [6].

They provide a thorough analysis of programmable networks, zeroing in on SDNs in particular. This paper provides a thorough exploration of SDN architecture & explains why it is so important in the realm of programmable networks. The discussion encompasses SDN protocol testing and alternative solutions compatible with OpenFlow [7].

In their overview of SDNs, Hu and co-authors emphasize the core concept, applications, and security attributes of Open Fl. The work sheds light on the fundamental aspects of SDNs, providing insights into their applications and security implications [8].

They investigate the finer points of scripted SSH brute force assaults. The research extensively examines both the behavior of attackers Considering the complexities of various attacks, such as the widespread use of password dictionaries and concerted cracking efforts, based on data sourced from the LongTail project. The insights drawn from this analysis offer actionable advice to SSH users and network managers [9].

Comprehensively covers a variety of SDN anomaly detection techniques, specifically addressing automated SSH brute force attacks. The k-nearest-neighbor (KNN) algorithm, Bayesian networks, & Support Vector Machines are all examples of such methods [10].

This study delves into the intricacies of attacker behavior and attack dynamics, utilizing insights garnered from the Longtail project. The investigation encompasses aspects methods include group guessing and exchanging password dictionaries. The findings of this investigation are helpful for both network administrators and SSH users. The k-Nearest Neighbors (KNN), Bayesian Networks, Support Vector Machines, and Expectation Maximization anomaly detection methods are all covered by Sommer. The author also explores the development of SDN applications tailored to different attack scenarios. Qazi and colleagues propose the innovative Atlas architecture, which uses SDN's application-awareness to successfully implement layer-2-4-based policies.

Atlas employs the C5.0 classifier, a machine learning technique, for SDN traffic classification. The framework employs crowdsourcing to collect ground truth data and integrate it with the SDN's centralized control and data reporting system. The proposed system excels at fine-grained application detection, achieving a 94% average accuracy in identifying the top 40 Android applications. Kim and coworkers provide an exhaustive overview of SDN, coupled with recommendations for enhancing network management. They specifically emphasize addressing present challenges within network setup and administration systems. Noteworthy features of their work include the capability to dynamically adjust network states and conditions, setup through higher-level languages, enhanced troubleshooting interfaces, and finer command.

FlowN enables tenants to tailor their address space, topology, and control logic. The use of databases aids in scaling the mapping between virtual and physical networks. Eskca et al. conduct an in-depth exploration of the security aspects of SDNs. Their study encompasses various security strategies, including machine learning techniques. The research introduces the B4 system— a private WAN linking Google's servers all around the world. B4 is designed to accommodate dynamic traffic demand, high bandwidth requirements, and scalability. The study further analyzes a range of approaches, including machine learning, for addressing security-related concerns. It describes the innovative B4 system, characterized by private WAN connections between Google's worldwide data centers. Key features include enhanced control over edge servers, high bandwidth capacity, and adaptability to dynamic traffic demand.

3. METHODOLOGY:

This section looks at the various SDN research initiatives made to recognise DDoS attacks. Several techniques, such as Random forest, Naive bayes, KNN, Neural Network, SVM, and SOM, have been found to be effective in stopping DDoS attacks. The suggested study uses Random Forest, AdaBoost, CatBoost, XGBoost algorithms to examine traffic's essential properties and identify DDoS attacks.
Moreover, the absence of robust and adaptive security measures within the SDN framework hinders real-time threat detection and response capabilities of the network. The reliance on manual intervention for threat identification and mitigation renders the system vulnerable to human error and latency in response time, thereby compromising the overall integrity and reliability of the network. Given the limitations of the existing system, there is a pressing need to incorporate advanced security solutions that leverage the capabilities of machine learning algorithms.

By integrating machine learning-based detection mechanisms, SDN networks can proactively monitor and analyze network traffic, identify anomalies, and predict potential intrusion events with greater accuracy and efficiency. These systems can learn from historical data patterns, identify deviations from normal network behavior, and autonomously adapt to new attack patterns, thus fortifying the SDN infrastructure against a wide array of security threats. The integration of machine learning techniques into the existing SDN security framework presents a paradigm shift, enabling the network to dynamically adapt and respond to emerging cyber threats. This transformative approach empowers SDN networks to detect and mitigate intrusion attacks in real time, thereby enhancing the overall security posture and ensuring the uninterrupted and secure operation of critical network infrastructures.

Software defined network Architecture: Our architecture has two distinct layers, the control layer and the infrastructure layer, as depicted in Fig. 1. The IoT network's physical switch resides in the infrastructure layer and is responsible for traffic forwarding per the control layer's directives. Our network architecture consists of a single switch (S1), many IoT hosts, a Ryu controller, and a machine learning–based SDN Ryu Controller. Each host is wired into the network through switch (S1). The OpenFlow protocol [32] is used for communication between the switch and the Ryu controller. The Ryu controller makes use of a switching programme to complete its tasks. In real-time, the switch feeds packet data to the Ryu controller's machine-learning classifier. This classifier utilizes our enhanced ML-SDN model to determine whether the traffic constitutes an attack. In the case of legitimate traffic, the controller examines the packet's destination MAC address and determines the destination port, and then creating a new rule in the underlying infrastructure to allow the traffic to flow. However, if the controller determines that the traffic is malicious, it will tell the infrastructure layer to discard the packets by means of a flow entry.

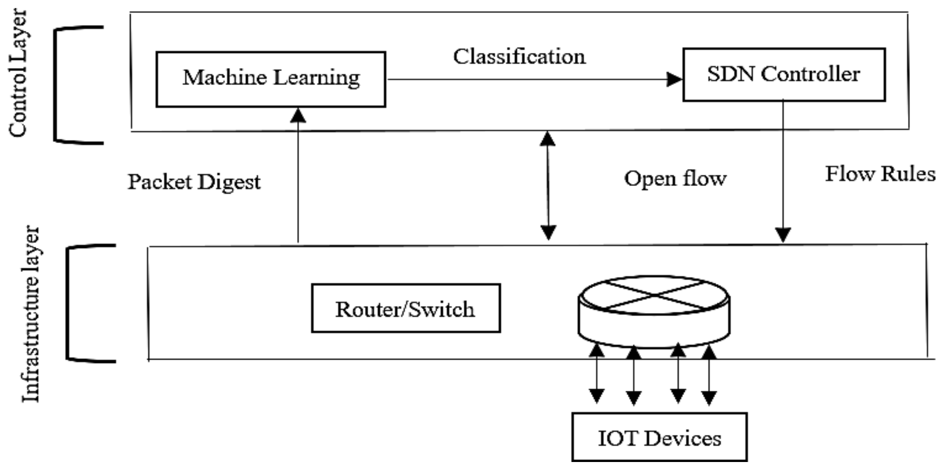


Figure 1: Predicting traffic flows using an SDN controller framework with machine learning

In this section the below diagram Fig 2 displays Our recommended method for using ML in SDN to detect DoS threats. Because of the clarity & precision of its classifications, we employed the Random Forest, AdaBoost, CatBoost, XGBoost algorithms to detect attacks.

System Architecture:

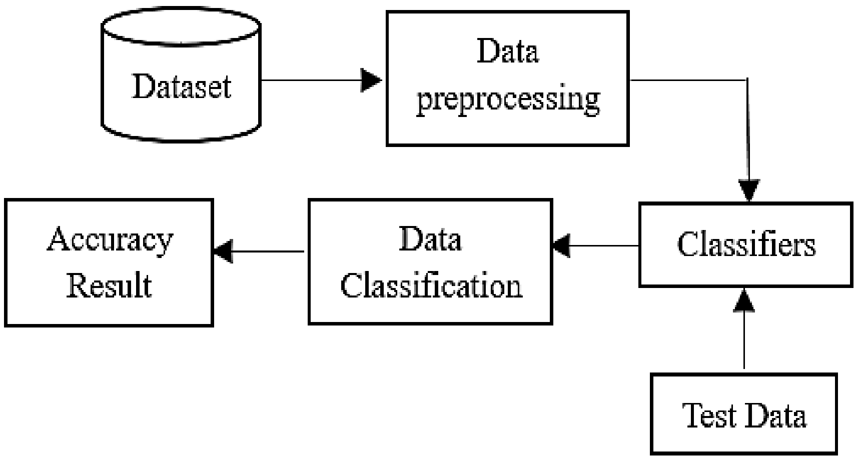


Figure 2. System architecture.

Data Gathering: This paper's information assortment comprises of various records. The determination of the subset of all open information that you will be working with is the focal point of this stage. Preferably, ML challenges start with a lot of information (models or perceptions) for which you definitely know the ideal arrangement. Marked information will be data for which you are as of now mindful of the ideal result.

Pre-Processing of Data:
Organize your data by formatting, cleaning, and sampling from it.
There are three typical steps in data pre-processing:

- Designing
- Information cleaning
- Inspecting

Designing: It's conceivable that the information you've picked isn't in a structure that you can use to work with it. The information might be in an exclusive record configuration therefore either a text document or social data collection would suffice, or vice versa. Alternatively, the information could already exist in a social data set, in which case a level document would suffice.

To remove inaccurate or outdated data, information cleansing is typically used. Sometimes the data examples you're given fall short of what you need to actually solve the problem. These events could should be eliminated. Moreover, a portion of the traits might contain delicate data, and it very well might be important to anonymize or totally eliminate these properties from the information.

Inspecting: You might approach significantly more painstakingly picked information than you want. Calculations might take significantly longer to perform on greater measures of information, and their computational and memory prerequisites may likewise increment. Prior to considering the whole datasets, you can take a more modest delegate test of the picked information that might be fundamentally quicker for investigating and creating thoughts.

Feature Extraction: The following stage is to A course of quality decrease is include extraction. Highlight extraction really modifies the traits instead of element choice, which positions the ongoing ascribes as indicated by their prescient pertinence. The first ascribes are straightly joined to create the changed traits, or elements. Finally, the Classifier calculation is utilized to prepare our models. We utilize the Python Normal Language Tool stash's classify module. We utilize the gained marked dataset. The models will be surveyed utilizing the excess marked information we have. Pre-handled information was ordered utilizing a couple of AI strategies. Irregular woodland classifiers were chosen. These calculations are generally utilized in positions including text grouping.

Assessment Model: Model The method involved with fostering a model incorporates assessment. Finding the model that best portrays our information and predicts how well the model will act in what's to come is useful. In information science, it isn't adequate to assess model execution utilizing the preparation information since this can rapidly prompt excessively hopeful and overfitted models. Information scientists use both the wait and cross-approval processes when assessing models. Both approaches prevent fitting by measuring model performance on a test set that is hidden from the user. When compared to the usual, every classification model's presentation is assessed. The result will take on the structure that was envisioned diagram portrayal of information that has been ordered.

Algorithms:

Random Forest: In the landscape of machine learning, the Random Forest algorithm has emerged as a pivotal technique, combining prediction accuracy & robustness that come from using several decision trees. Introduced as an ensemble learning method, Random Forest has gained popularity due to its efficacy in addressing issues like overfitting, variance, and interpretability.

At its core, Random Forest generates a multitude of decision trees, each operating on a random subset of the dataset. This process, known as bootstrapping, introduces an element of randomness that diversifies the trees' learning. Furthermore, during each tree's growth, only a random subset of features is considered for splitting nodes. This double-layered randomness imparts resilience to the model against overfitting, ensuring that no single tree dominates the predictive outcome.

The real strength of Random Forest emerges from its ensemble approach. Once the individual trees are constructed, their predictions are aggregated to yield a final prediction. This process takes advantage of the "wisdom of the crowd" principle, where the collective decision of numerous trees is often more accurate and reliable than that of a single tree. The ensemble nature of Random Forest not only enhances predictive accuracy but also combats the issue of bias by averaging out individual errors.

The balance between bias and variance, a critical aspect of model performance, is a feat achieved gracefully by Random Forest. By aggregating the predictions of diverse trees, the algorithm strikes a delicate equilibrium. High-variance, biassed trees are countered by low-variance, high-bias ones, resulting in an ensemble that captures the nuances of the data while maintaining generalizability.

One of the algorithm's distinctive attributes is its interpretability. While many modern machine learning models operate as black boxes, Random Forest retains transparency. The decision path of each tree is comprehensible, allowing analysts to understand the factors that contribute to a specific prediction. In an era where model interpretability is increasingly valued, Random Forest offers a unique advantage, especially in domains where insights into the decision-making process are crucial.

XGBoost: Gradient-boost decision trees are implemented using XGBoost. In C++, this library was created. It is a type of software library whose main goal is to enhance the performance and speed of models. In recent years, applied machine learning has given it more significance. Numerous Kaggle competitions are dominated by XGBoost models. This algorithm creates decision trees in a sequential manner. With XGBoost, weights are important. To make predictions, a decision tree is fed scores from each independent variable.

The factors are then used to populate a second decision tree, with greater emphasis placed on those that were incorrectly predicted by the first. The combination of these individual predictors and classifiers results in a more reliable and accurate model. Regression, classification, ranking, and custom prediction issues are just a few examples of applications.

Features of XGBoost the library's focus is on model performance and computational speed, so it has few frills. The Model's Features There are three main types of supported gradient boosting, parallel construction of trees, training massive models with the help of distributed computing, Data Structure and Algorithm Caching Optimisation Improved and Optimised XGBoost Training on huge datasets is accelerated by XGBoost's unique tree-generation and tree-pruning approach, as well as a variety of in-built optimisations. Regularized Inclination Supporting Framework Highlights Here are a couple of the main ones'

A resemblance to a Greedy Algorithm: This calculation utilizes weighted quantiles to choose the best hub split as opposed to assessing every competitor split.

Access with Cash Awareness: XGBoost utilises the CPU's cache memory to save information.

Sparsity: When there is some missing information, Aware Split Finding takes advantage of lacking observation information to determine Gain. Gain-maximizing scenarios are placed in the appropriate leaf, and the procedure is repeated.

Benefits:
Regularization Techniques: XGBoost's innovative use of L1 (Lasso) and L2 (Ridge) regularization terms revolutionized the way models handle complexity. By penalizing large coefficient values, these techniques prevent overfitting, making XGBoost models robust and resilient to noise in the data.

Gradient-Based Optimization: The algorithm's namesake, the gradient boosting approach, is further optimized by employing a novel technique called "Gradient Boosting with Exact Greedy Algorithm." This method accelerates convergence and enhances the algorithm's overall efficiency.

Customizable Loss Functions: XGBoost allows users to define their own loss functions, enabling the algorithm to cater to specific business objectives. This flexibility lends itself to applications in diverse fields, from finance to healthcare.

Handling Missing Values: XGBoost's ability to learn patterns from missing data values reduces the need for data preprocessing, saving time and effort in feature engineering.

Parallel and Distributed Computing: The algorithm's design prioritizes efficiency, leveraging parallel and distributed computing capabilities to accelerate training and prediction times. This feature makes XGBoost particularly well-suited for big data applications.

Interpretability: Despite its advanced techniques, XGBoost retains a level of interpretability that sets it apart from many other complex models. Feature importance scores can be extracted, aiding in understanding model decisions.

Catboost: We frequently come across datasets with categorical characteristics, and in order to fit these datasets into the Boosting model, we use a variety of encoding strategies, such as One-Hot Encoding or Label Encoding. However, using One-Hot encoding results in a sparse matrix, which can occasionally cause the model to get overfitted. To address this problem, we employ CatBoost. CatBoost manages category features automatically.

CatBoost, often known as categorical boosting, boosting library. It is intended to be used with issues like regression and classification that have a substantial number of independent features.

Catboost is a gradient boosting variation that works with both category and numerical features. Numerical features can be generated from categorical data without the need of feature encoding techniques like One-Hot Encoder or Label Encoder. Additionally, to lessen overfitting and enhance the overall performance of the dataset, it makes use of an approach called symmetric weighted quantile sketch (SWQS), which automatically manages the missing values in the dataset.

CatBoost characteristics:
CatBoost has a built-in method for managing categorical features and can do so without feature encoding.

Internal mechanisms for dealing with missing values CatBoost, in contrast to other Models, can readily accommodate any missing values in the dataset.

While in other models we need to significantly modify columns, CatBoost internal automatically scales all the columns to the same scaling.

Cross-validation is already included into CatBoost, and it uses it to select the model's ideal hyperparameters.

Regularizations to lessen overfitting, CatBoost supports both L1 and L2 regularization techniques.
It may be applied to both Python and R.

AdaBoosting Classifier: Ada-boost, also known as Adaptive Boosting, is a categorization of aid groups developed by Yoav Freund and Robert Schapire in 1996. It mixes various classifiers to improve classifier precision. AdaBoost is an iterative outfit approach. The AdaBoost classifier builds regions of strength for a, providing you high areas of strength for exactness by combining many classifiers that combine inefficiently. Adaboost's main principle is to set up the classifier loads and get ready for each cycle's information test to the point where it guarantees precise forecasts of unanticipated impressions. Any artificial intelligence computation capable of identifying loads on the training set can serve as the primary classification. There are two rules that Adaboost has to follow.

The classifier needs to be prepared intelligently using a number of weighed preparation models. In order to provide these samples with the greatest fit possible throughout each iteration, it works to decrease training error.

How does the AdaBoost algorithm work? Here is how it works:

A training subset is originally selected by Adaboost at random. It iteratively trains the AdaBoost AI model by choosing the preparation set in consideration of the precise expectation of the prior preparation. It gives incorrectly characterized perceptions a heavier burden, increasing their likelihood of grouping in the attention that follows. Additionally, it transfers the burden to the trained classifier in each emphasis in accordance with the classifier's accuracy. The classifier that is more accurate will be given more weight. This cycle repeats until there are the predefined maximum number of assessors or until the entire preparation information fits with virtually minimal error. Play out a "vote" involving all of the artificial learning computations to determine the ranking.

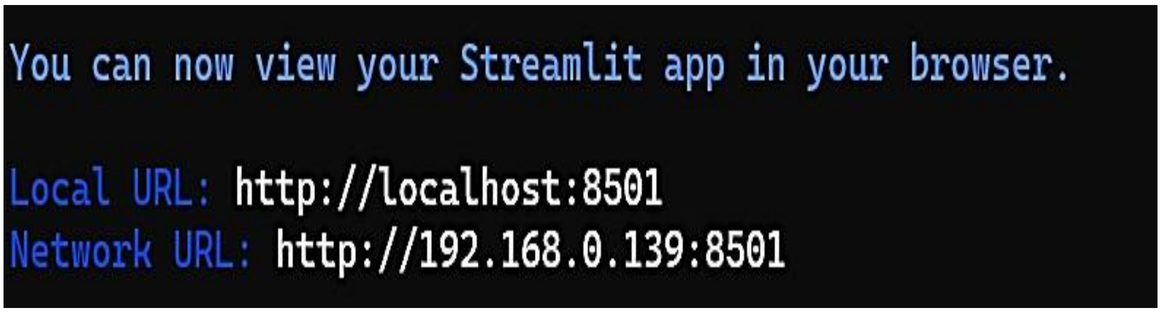
Accuracy: The level of precise expectations for the test information is implied by precision. By partitioning the quantity of exact expectations by the complete number of forecasts, it very well might still up in the air.

User Interface: The pattern of Information Science and Examination is expanding step by step. From the information science pipeline, one of the main advances is model sending. We have a ton of choices in python for sending our model. A few well-known systems are Carafe and Django. Yet, the issue with utilizing these systems is that we ought to have some information on HTML, CSS, and JavaScript. Remembering these requirements, Adrien Treuille, Thiago Teixeira, and Amanda Kelly made "Streamlit". Presently utilizing streamlit you can send any AI model and any python project easily and without stressing over the frontend. Streamlit is very easy to use.

In this article, we will get familiar with a few significant elements of streamlit, make a python project, and convey the task on a nearby web server. How about we introduce streamlit. Type the accompanying order in the order brief.

pip installs streamlit : When Streamlit is introduced effectively, run the given python code and in the event that you don't get a mistake, then streamlit is effectively introduced and you can now work with streamlit. Instructions to Run Streamlit record and then the below diagram Fig 3 shows the output screen.

How to Run Streamlit file:



```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://192.168.0.139:8501
```

Figure 3 output screen.

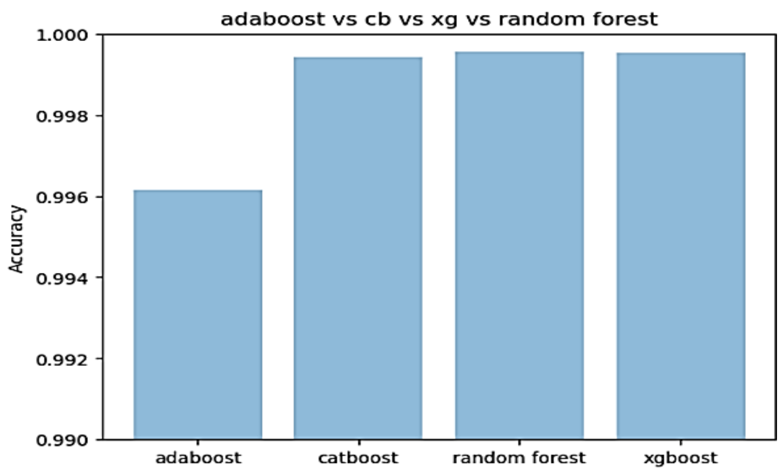


Figure 4 (Overall Results)

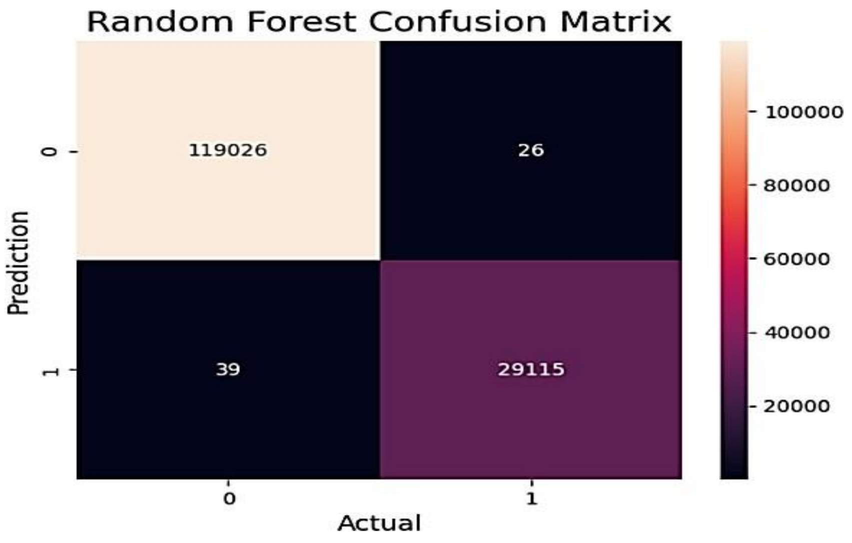


Figure 4(Confusion Matrix for Random Forest)

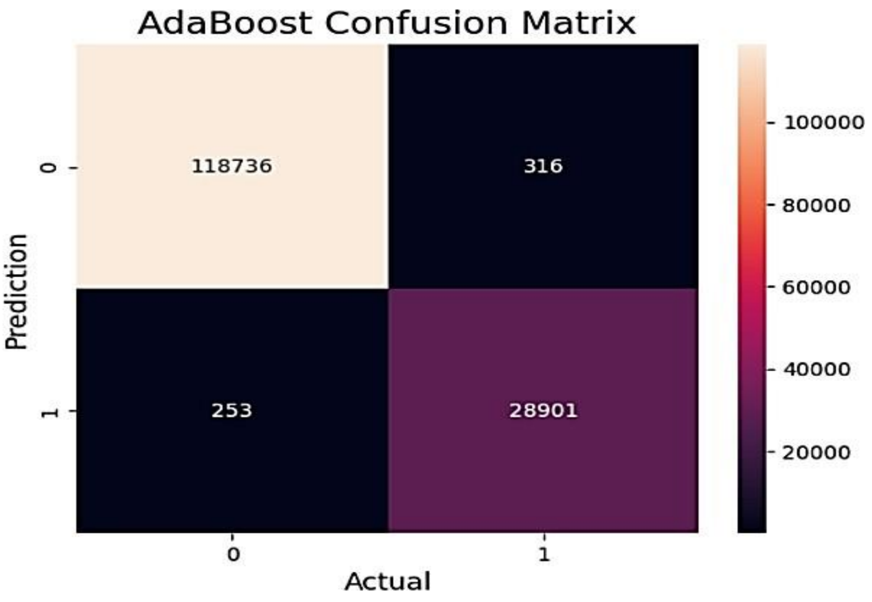


Figure 5(Confusion Matrix for AdaBoost)

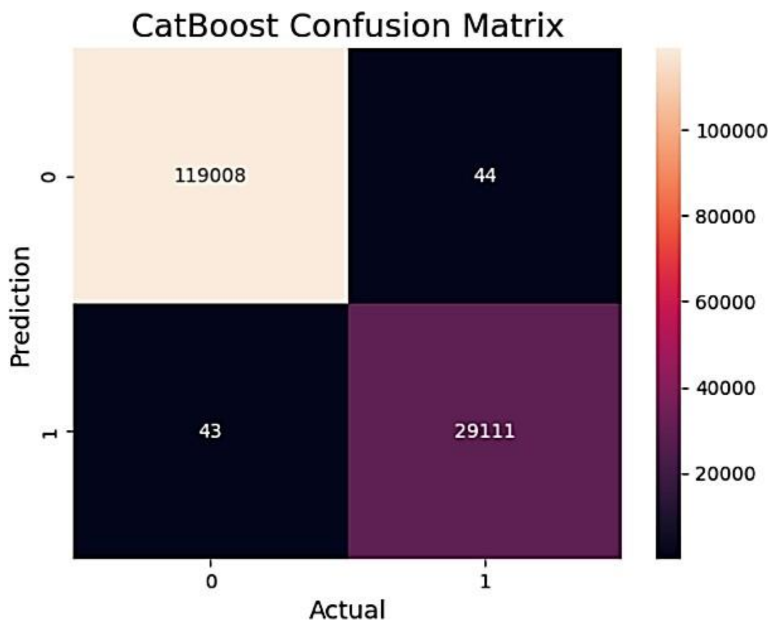


Figure 6(Confusion Matrix for CatBoost)

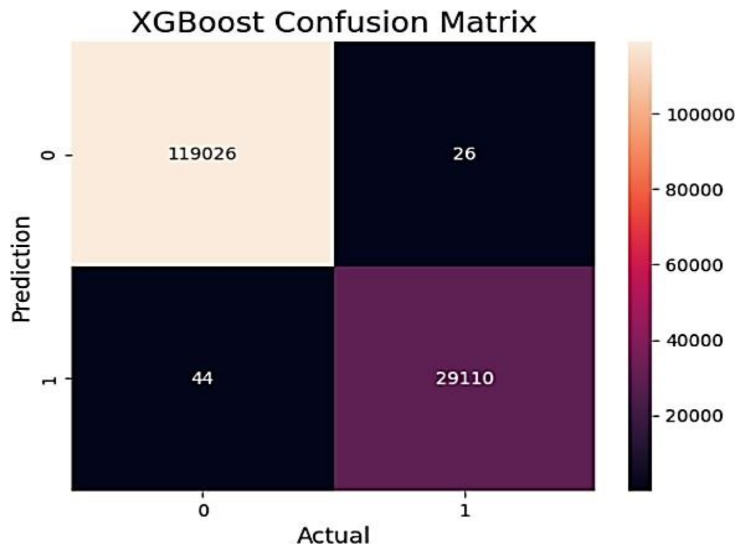


Figure 7(Confusion Matrix for XGBoost)

4. CONCLUSION:

In this paper, we utilized the KDD99 dataset to prepare and test our proposed model. The DDoS attack was identified by employing the Random Forest, AdaBoost, CatBoost, XGBoost algorithms. On the SDN environment, the classification module is implemented. To differentiate between legitimate traffic data and malicious traffic data, Random Forest, AdaBoost, CatBoost, XGBoost methods are utilized. Our experiment demonstrates that, in our simulated environment, Random Forest, CatBoost, XGBoost performs better than AdaBoost and in the below diagram fig 4 shows the overall results of the proposed approach and then the below diagram Fig 5 shows the confusion matrix for the adaboost matrix and then the Fig 6 displays the confusion matrix for the catBoost and then the fig 7 displays the Confusion matrix for XGBoost. By analyzing and interpreting the vast amounts of network data in real-time, machine learning algorithms have proven their capability to identify anomalous patterns and potential security breaches with heightened accuracy and efficiency. This not only aids in fortifying the resilience of SDN architectures but also empowers network administrators and security professionals to proactively respond to emerging threats, thereby minimizing the potential impact of cyber-attacks. However, while the application of machine learning presents a promising avenue for bolstering SDN security, it is imperative to acknowledge the ongoing need for continuous refinement and adaptation of these algorithms to combat the dynamic nature of modern-day cyber threats. Furthermore, integrating a multi-layered approach that combines traditional security measures with machine learning-based detection mechanisms will undoubtedly reinforce the overall defense posture of SDN infrastructures.

5. REFERENCES:

[1] Alazab, M. Hobbs, J. Abawajy, and M. Alazab, “Using feature selection for intrusion detection system,” 2012 Int. Symp. Commun. Inf. Technol., pp. 296–301, 2012.

[2] M. P. K. Shelke, M. S. Sontakke, and A. D. Gawande, “Intrusion Detection System for Cloud Computing,” Int. J. Sci. Technol. Res., vol. 1, no. 4, pp. 67–71, 2012.

[3] S. Suthaharan and T. Panchagnula, “Relevance feature selection with data cleaning for intrusion detection system,” 2012 Proc. IEEE Southeast on, pp. 1–6, 2012.

[4] S. Suthaharan and K. Vinnakota, “An approach for automatic selection of relevance features in intrusion detection systems,” in Proc. of the 2011 International Conference on Security and Management (SAM 11), pp. 215-219, July 18-21, 2011, Las Vegas, Nevada, USA.

[5] L. Han, "Using a Dynamic K-means Algorithm to Detect Anomaly Activities," 2011, pp. 1049-1052.

[6] R. Kohavi, et al., "KDD-Cup 2000 organizers report: peeling the onion," ACM SIGKDD Explorations Newsletter, vol. 2, pp. 86-93, 2000.

[7] J. McHugh, “Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory,” ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, 2000.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A Detailed Analysis of the KDD CUP 99 Data Set,” Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[9] P. Ghosh, C. Debnath, and D. Metia, “An Efficient Hybrid Multilevel Intrusion Detection System in Cloud Environment,” IOSR J. Compute. Eng., vol. 16, no. 4, pp. 16–26, 2014.

[10] Dhanabal, L., Dr. S.P. Shantharajah, "A Study on NSL_KDD Dataset for Intrusion Detection System Based on Classification Algorithms," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, issue 6, pp. 446-452, June 2015