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.

ABSTRACT

Digitally constructing and designing hardware components is done with a network architecture known as a "software-defined network" (SDN). The settings for the network connection can be changed on the fly. In the conventional network, dynamic change is impossible because the link is fixed. Although SDN is an excellent strategy, DDoS attacks can still occur. The internet is in danger as a result of the DDoS attack. DDoS attacks can be stopped with the help of the machine learning algorithm. When multiple systems collaborate to simultaneously target a specific host, this is called a DDoS attack. Software from the control layer, which is located in the middle of the application and infrastructure layers, controls the devices in the infrastructure layer in SDN. For the purpose of identifying malicious traffic, we propose a machine learning approach which implements Random Forest, AdaBoost, CatBoost, XGBoost algorithms in this essay. The results of our test show that the Random Forest, CatBoost, XGBoost algorithms have a higher detection rate and accuracy.

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

1. INTRODUCTION

Numerous useful solutions have emerged as a result of the dramatic rise in the number of Internet-connected devices across a variety of industries, including agriculture, health care, and commerce. Traditional network architectures have been challenged by the enormous increase in connectivity demand. Software Defined Network (SDN) architecture, which separates the conventional user plane and control plane, was proposed to address the issues. An advantage of this architecture is that it makes it easier to manage networks and boosts network efficiency as a whole. Even though this kind of architecture has a lot going for it, it's also vulnerable to a lot of threats, like security attacks.

On the SDN controller, Attacks like the Secure Shell (SSH) brute force attack, which pose major security risks, can be launched by an attacker. Even if the network administrator recognises a potential attack and an attacker, it might not be possible to adequately account for concurrent attempts in real time.As a result, the SDN controller must be configured with appropriate security policies, much like firewall rules. Determining these rules, however, can be challenging because their goal is to keep rogue nodes or attackers out of the system while maintaining normal user access. Malicious users possess specific characteristics that can be used to distinguish them from legitimate users. Attackers frequently follow patterns like coordinated attacks and sharing password dictionaries. These patterns can be found using a variety of methods, including machine learning. Approaches based on machine learning have demonstrated significant potential for user classification.

LITERATURE SURVEY

The utilization of machine learning strategies to address Attacks on Software Defined Networks (SDNs), encompassing intrusions and Distributed Denial of Service (DDoS) attacks, was investigated by Ashraf and colleagues [2]. 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.

In their work [6], Ali et al. provide a comprehensive guide on harnessing SDNs to enhance network security and promote SDNs as a security-as-a-service solution. The survey compiles a range of challenges

Copyright @ 2023 Author

ISSN: 2278-4632
Vol-13, Issue-09, September 2023

and potential solutions from the literature to address network threats.

Astuto et al. [7] present a comprehensive review of programmable networks, focusing specifically on SDNs. The paper delves into the architecture of SDNs and highlights their significance in the context of programmable networks. The discussion encompasses SDN protocol testing and alternative solutions compatible with OpenFlow.

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

Abdou and colleagues [9] delve into the specifics of automated SSH brute force attacks. The research extensively examines both the behavior of attackers and the dynamics of these attacks, including password dictionary sharing and coordinated attempts, based on data sourced from the LongTail project [5]. The insights drawn from this analysis offer actionable advice to SSH users and network managers. Sommer [10] comprehensively covers a variety of SDN anomaly detection techniques, specifically addressing automated SSH brute force attacks. These techniques include k-Nearest Neighbors (kNN), Bayesian Networks, and Support Vector Machines.

This study delves into the intricacies of attacker behavior and attack dynamics, utilizing insights garnered from the LongTail project. The investigation encompasses aspects such as coordinated attempts and the sharing of password dictionaries [5]. The conclusions drawn from this analysis provide valuable guidance to network managers and SSH users. Additionally, Sommer [10] discusses several SDN anomaly detection techniques, encompassing k-Nearest Neighbors (KNN), Bayesian Networks, Support Vector Machines, and Expectation Maximization. The author also explores the development of SDN applications tailored to different attack scenarios. Qazi and colleagues propose the innovative Atlas architecture, which leverages application-awareness within SDN and proves effective for policy enforcement based on layers 2, 3, and 4.

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, high-level language support for configuration, and improved troubleshooting interfaces and control.

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. [1] 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 Wide Area Network (WAN) interconnecting Google's global data centers. 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.

EXISTING SYSTEM

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.

Copyright @ 2023 Author

DIS ADVANTAGES:

- The experiment results shows less accuracy.
- More complexity

PROPOSED METHOD

In this section, we go over our suggested approach for applying ML in SDN to identify DDoS attacks. Due to its precise categorization and simplicity, we employed the Random Forest, AdaBoost, CatBoost, XGBoost algorithms to detect attacks.

ADVANTAGES:

- The experiment result shows high accuracy
- More effective detection of the attacks due to its accurate classification.
- Less complexity

2. SYSTEM ARCHITECTURE

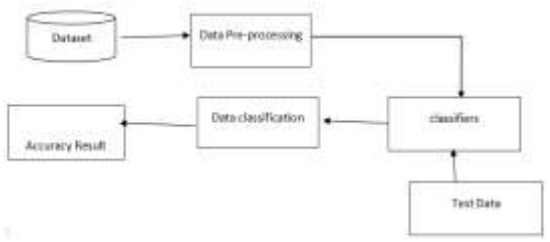


Figure 1

3. METHODOLOGY

i. Data Gathering,
ii. preprocessing of the data,
iii. feature extraction,
iv. evaluation model, and
v. user interface

3.1 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.

3.2 Pre-Processing of Data

Format, clean, and sample from your chosen data to organize it.

There are three typical steps in data pre-processing:

1. Designing
2. Information cleaning
3. 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 and you would like it in a social data set or text document, or the information might be in a social data set and you would like it in a level document.

Information cleaning; is the most common way of eliminating or supplanting missing information. There can be information examples that are inadequate and come up short on data you assume you really want to resolve the issue. These events could should be eliminated. Moreover, a portion of the traits might contain delicate data, and it very well might be important to antonymize 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.

3.3 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.

3.4 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. Wait and Cross-Approval are two procedures utilized in information science to evaluate models.

The two methodologies utilize a test set (concealed by the model) to survey model execution to forestall over fitting. In light of its normal, 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:

1) Random Forest

In the landscape of machine learning, the Random Forest algorithm has emerged as a pivotal technique, combining the strength of multiple decision trees to deliver accurate and robust predictions. 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 and low-bias trees are balanced by those with low-variance and high-bias tendencies, 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.

***2)* 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. Each independent variable is given a weight, which is then used to feed a decision tree that forecasts results.

The variables are then loaded into the second decision tree by giving the variables that the previous tree mistakenly predicted more weight. 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:

The following features are provided by this library for use in a variety of computing environments: parallel construction of trees; Using distributed computing to train large models; Cache Optimization of Data Structures and Algorithm XGBoost Enhancements and Optimizations XGBoost's unique method of generating and pruning trees and a number of built-in optimizations speed up training when working with large datasets. 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 stores data in the CPU's cache memory.

Sparsity: Aware Split Finding uses observations with missing values to calculate Gain when there is some missing data. The process of selecting the scenario with the greatest Gain and placing it in the appropriate leaf is then 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.

3. *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.

Regularisations - To lessen overfitting, CatBoost supports both L1 and L2 regularisation techniques.

It may be applied to both Python and R.

4. AdaBoosting Classifier

Ada-boost or Adaptive Boosting is one of the help group classifications made 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. The fundamental classifier can be any AI computation that recognises loads on the training set. Adaboost must abide by two conditions:

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 characterised 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

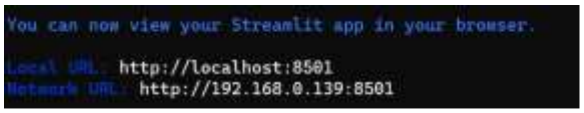
The pattern of Information Science and Examination is expanding step by step. From the information science pipeline, one of the main advances is model sending. We have a ton of choices in python for sending our model. A few well known systems are Carafe and Django. Yet, the issue with utilizing these 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 install 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:

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 2

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.

OUTPUT RESULTS:

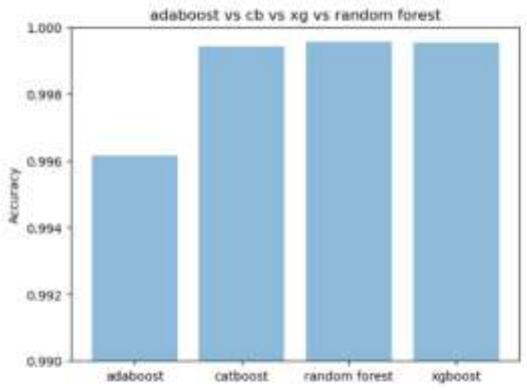


Figure 3(Overall Results)

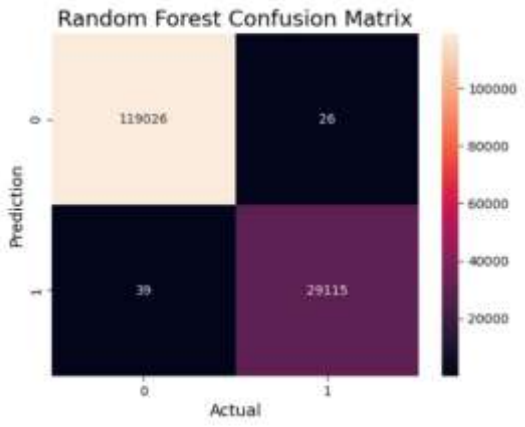


Figure 4(Confusion Matrix for Random Forest)

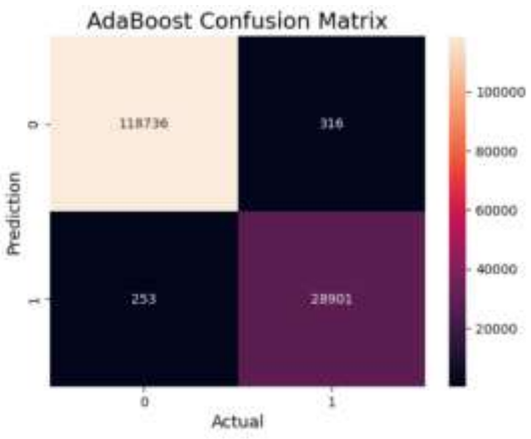


Figure 5(Confusion Matrix for AdaBoost)

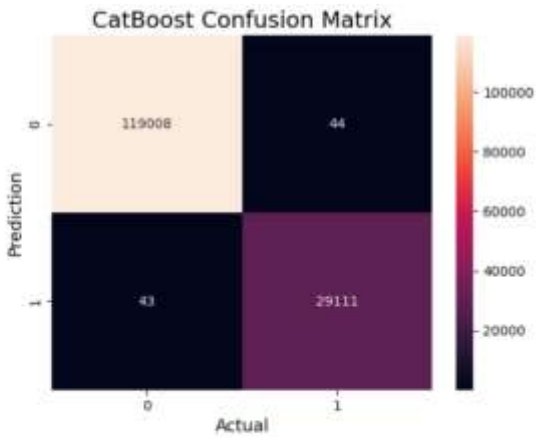


Figure 6(Confusion Matrix for CatBoost)

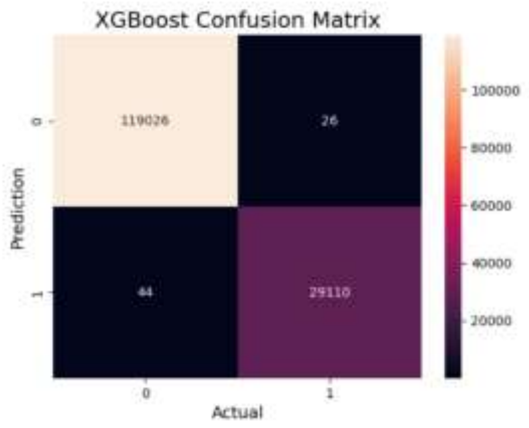


Figure 7(Confusion Matrix for XGBoost)

REFERENCES

[1] A. 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 Southeastcon, 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. Comput. Eng., vol. 16, no. 4, pp. 16–26, 2014.

[10] Dhanabal, L., Dr. S.P. Shantharajah, "A Study on NSL_KDD Daaset 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

Copyright @ 2023 Author