International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Boosting Attack Detection Capabilities in Multi-Tenant Distributed Systems via Meta-Ensemble Classifiers and Weighted Averaging

Pravin Patil[1], Dr. Geetanjali Kale[2], Maanav Chandwani[3], Aarushi Wagh[4]
[1]Dept. of Computer Engineering
Pune Institute of Computer Technology
Pune, India
e-mail: prppatil@pict.edu
[2]Dept. of Computer Engineering
Pune Institute of Computer Technology
Pune, India
e-mail: gvkale@pict.edu
[3]Dept. of Computer Engineering
Pune Institute of Computer Technology
Pune, India
e-mail: maanavchandwani@gmail
[4]Dept. of Electronics & Telecomm. Engineering
Pune Institute of Computer Technology
Pune, India
e-mail: aarushiwagh16@gmail.com

Abstract— Multi-tenant distributed systems are a great way to share resources and scale performance, but they come with their share of security issues due to the introduction of more than one client on the cloud. Sharing of hardware and software resources among tenants gives rise to vulnerabilities and malicious tenants can misuse this shared data to launch attacks on other tenants. While efforts have traditionally focused on building secure network architectures, it is impossible to create a completely secure system due to its open-ended nature. This paper explores ways to detect malicious tenants on the cloud using machine learning algorithms. This paper proposes an ensemble-based meta-classifier to predict the probability of attack instantiation based on certain system parameter values. Additionally, this paper creates a dataset for analysis purposes and address the class imbalance problem often found in this domain where attack instances are rare. Satisfactory results were produced to distinguish between non attach and attack instances.

Keywords- multi-tenant distributed systems, machine learning, cloud security.

I. INTRODUCTION

Most modern applications and organizations are shifting to cloud computing and multi-tenant distributed systems to reduce overall IT costs and increase the implementation time of services. In a multi-tenant architecture, various clients share the same physical and computational resources and have access to other clients' information and data, which gives rise to vulnerabilities. Since cloud operators have no way of making sure that the clients are legitimate, many times malicious tenants pose as legitimate clients and attack other tenants on the cloud. The timely detection of such attacks is crucial to prevent any harm to entities within the system [1].

In the past, various methods have been attempted to detect attacks in distributed systems, including statistical methods and computing function values. More recently, novel methods involving machine learning, including the use of neural networks, have been explored [2]. Existing approaches have faced challenges such as false inference of results due to class imbalance and confusion related to feature selection due to the availability of a variety of parameters.

This paper studies the signatures and records of previous attacks to predict the tenants that are most likely to be malicious based on their activity. The paper proposes a meta ensemble-based classifier approach to detect potential attacks in a multi-tenant distributed system. The study trains five classifiers on pre-processed data and assigns a weight factor to each model's prediction, to combine them using weighted average ensemble to generate a result that is not overfitted.

One of the biggest issues for this study was the absence of reliable attack datasets. Hence, generated a publicly available dataset with over 50,000 timeframe instances for attack detection using a virtual network setup. To address the heavy imbalance in the data caused by the scarcity of attacks, the study uses undersampling and oversampling penalization techniques [3].

The remainder of the paper is organized as follows: Section 2 discusses the related work in this area. Section 3 explains the data collection and feature engineering methods. Section 4 and 5 describes the classifiers and meta-ensembling process. The results are presented in section 6, and the paper concludes in section 7.

II. RELATED WORK

The security issues posed by multi-tenant distributed systems are a well-researched topic. Brown et al. discussed specific risks in cloud computing due to Multi Tenancy and the measures that can be taken to combat the same [8].

Signature-based detection is a widely used method for detecting network attacks. The approach involves analyzing the unique characteristics of an attack, known as its "signature," to predict potential future risks [9]. Hilker et al. proposed techniques for identifying optimal signatures for attacks [10], while Han et al. suggested the use of different features to craft network traffic for detection [11]. However, this method has a major drawback in that it cannot detect new, previously unknown attacks due to a lack of relevant information. Additionally, creating new signatures requires manual intervention and can be time-consuming, making the overall process expensive.

Machine learning has also been employed in this area, with supervised learning-based algorithms being the preferred choice for network attack detection [12]. Zseby et al. advocated for feature selection and mapping for detecting attacks [13], while Rafique et al. used evolutionary algorithms to evaluate the performance of malware classification [14]. It should be noted, however, that due to the low probability of an attack occurring, a model can achieve high accuracy by predicting all samples as non-negative (no attack). Previous studies have also focused on using boosting methods and feature compression in transfer learning. For instance, Dai et al. presented TrAdaBoost, which re-weights the positive and negative class samples to ensure that rare samples indicating attacks contribute more to the result [15]. Pan et al. used TCA-transfer component analysis to bring the domains closer together by projecting features into the shared space [16], while Shi et al. proposed HeMap, a method that uses linear transformations to project features [17]. Model-based approaches have also been used for attack detection, which falls under transfer learning and assumes that some model priors or parameters are shared between the source and target tasks. In unknown environments, transfer learning can improve the robustness of malware detection, as demonstrated by Bekerman [18].

A key observation in these previous approaches is that the significant class imbalance in network attacks is often overlooked. As a result, accuracy cannot be the only performance metric used to evaluate results, and performance on other metrics must also be considered. Furthermore, the use of statistical ensemble models in this field has been limited, and there is a lack of information on the ideal features to be selected in a multi-tenant distributed system. This paper aims to address these gaps and provide better results.

III. DATA PRE-PROCESSING (FEATURE SELECTION, CLASS IMBALANCE)

A. Feature Selection

There were too many non-essential features in the data set which needed to be processed or removed. Feature selection plays an important part in deciding the quality of the results generated. The common problem of overfitting can be solved if appropriate features are selected from the data set. The features in our dataset represent various performance metrics and specifications of tenants in the cloud. This also increases the accuracy of the results and reduces the training time of the algorithm. These features were shortlisted using feature selection techniques:

a) app_cpu_netdata_x
b) app_cpu_apps.plugin_x
c) app_cpu_tc-qos-helper_x
d) app_cpu_ssh_x
e) running
f) freeused
g) cached
h) buffers

B. Class Imbalance

The original data set contains 4986 non-attack instances and 60 attack instances. This shows an imbalance in the ratio of majority class to minority class instances. This may lead to a bias in the dataset, which can in turn skew the performance and results of the model. To solve this problem, the process of under-sampling of the majority class and oversampling of the minority class is used. Before applying these methods, the data set was split into train and test sets. The train and test sets are divided so that equal proportions of instances of both classes are present in both divisions and can be used to check the accuracy of the ML model. These divisions were made in the ratio 3:1 (majority: minority). The balancing of classes was applied to the train division of the original dataset.

1) Oversampling of Minority Class: The lower ratio of minority class instances can hamper the result and make it more biased towards the majority class (non-attack instances in our case). Hence, this paper needs to increase the instances of the minority class. Use of the RandomOverSampler function from the Python library of scikit-learn to balance this data set. First, the paper label every non-attack instance as '0' and attack

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

instance as '1'. RandomOverSampler picks out random objects from the data set and replaces them back into the set. Thus, using this technique increases the minority instances without affecting the quality of the data present. This process decreases the imbalance in the ratio of training data to 2:1 (majority: minority).

2) Undersampling of Majority Class: The ratio still being 2:1, there was still need to decrease the number of majority class instances as they are much greater in number. So, the use RandomUnderSampler from the scikit-learn library on the dataset to bring down the ratio to 1.8:1. This technique will select the instances which have a replacement if possible and remove them from the data set. This will get the ratio of the data to our desired value with comparable proportions of both instances.

It is now observed that a ratio of 9:5 is achieved where there are 2336 non-attack instances and 1869 attack instances to work with. Furthermore, the experiment can start training our models as the data is now cleaned and processed.

IV. MACHINE LEARNING MODELS

Here the experiment will train 5 different classifiers on the dataset, which will give us the required results.

A. Decision Tree Classifier: The decision trees classifier is a supervised machine learning algorithm. A tree structure is created while training the algorithm. The size and shape of the tree depend on the amount of variation in the dataset. Division of this tree structure is made by answering a few basic questions regarding the properties of the features selected for training. For example, one side of the tree will contain instances that provide 'yes' to a specific question and the other branch will have all instances that return 'no' to the same question. This process will terminate at a specific level to avoid overfitting of the algorithm. The last nodes or leaf nodes may be pure leaf nodes i.e., with instances of only one class or a mixed leaf node with instances of both classes. These leaf nodes will decide on assigning further instances with a class. Every split in the tree is done so that the loss function is minimized. Here entropy of each node is measured and minimized.Mathematical representation of a Decision Tree Classifier

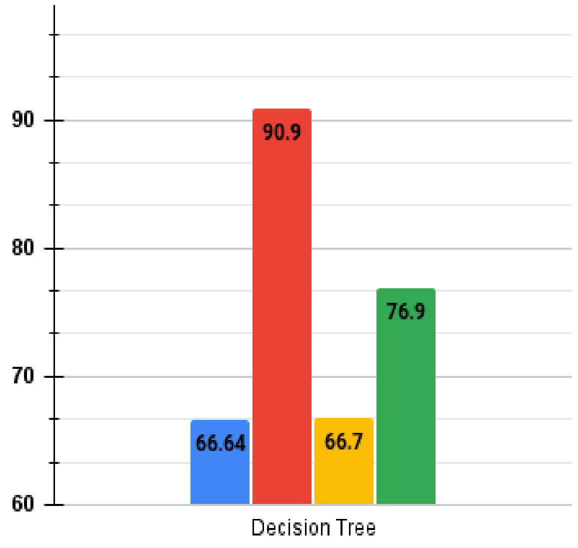

Figure 1. Decision Tree

B. Random Forest Classifier: Random Forest Classifier is based on the general concept of using multiple decision trees generated using different features and logic. These multiple rees that are generated to provide a class to an instance will predict a class for an instance. The class that is predicted may be different, so voting is performed and the majority class is assigned to the instance. Hence, random forest classifier consists of a large number of decision trees to operate as an ensemble. The results produced by a random forest are more accurate and precise than the ones produced by individual decision trees as all the decision trees that make up a random forest are different and uncorrelated and they hide their individual errors from each other.

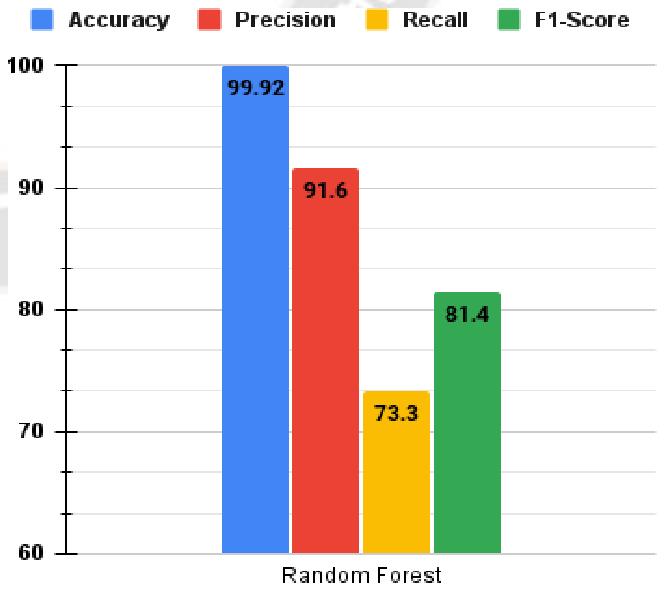

Figure 2. Random Forest

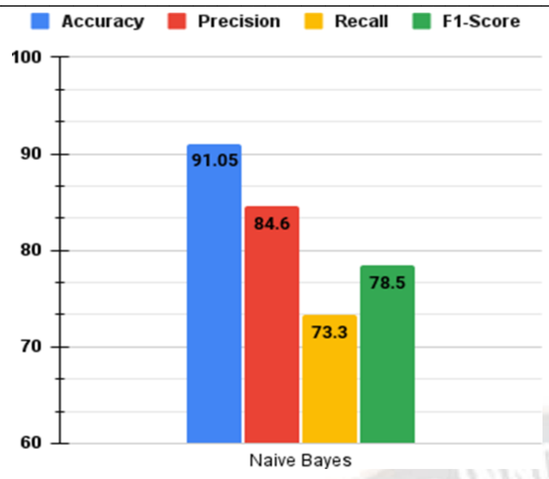

Figure 3. Naïve Byes

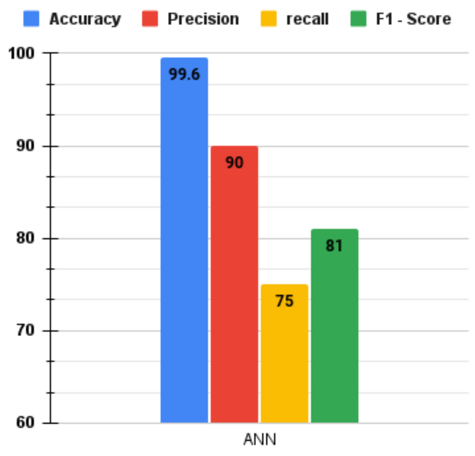

Figure 4. ANN

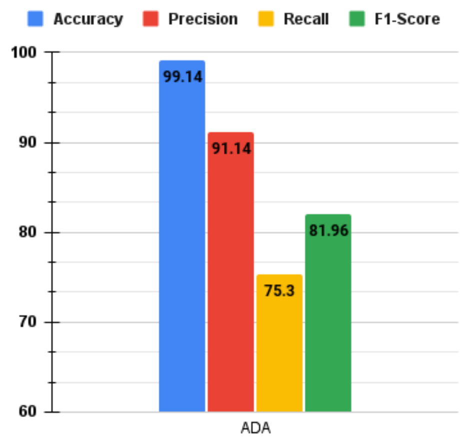

Figure 5. ADA

C. Gaussian Naïve Bayes Classifier: Taking Inspiration from the Bayes probability theorem, this classifier uses conditional probability on a Gaussian distributed dataset. Conditional probability is the probability of an event when some other event has already taken place. This concept is used for classification and is thus known as a probabilistic ML algorithm. Also, during classification, the features of the dataset are assumed to be independent and have a Gaussian (normal) distribution.

D. Support Vector Machine (SVM): It is a supervised learning algorithm. It creates a line or a decision boundary in the data set that segregates it into classes so as to put any new data point into its correct class. These partitions or hyperplanes are extreme cases in a feature class. When appropriate hyperplanes are created, getting ranges or spaces which depict a specific class and thus perform classification.

E. Multi-layer Perceptron: The multi-layer perceptron or Artificial Neural Network(ANN) can be best considered a weighted directed graph. Here, the artificial neuron in ANN is represented as a node present in the graph and the weighted edges are viewed as neuron inputs and outputs. When an input reaches a node, it is processed through a mathematical function and an output is generated. These outputs are conditioned by weights accordingly, so as to produce an appropriate classification. These weights generally define the amount or extent of interconnection between neurons in the artificial neural network. using a binary function to have a binary

V. GETTING RESULTS

After training all 5 models, their confusion matrices were generated. A confusion matrix consists of 4 parts - True Positive, False Positive, True Negative, and False Negative. They represent a combination of the original class value and the predicted class. For example, if an instance with 0 as its original class label is predicted as belonging to class 1, it will be categorized as False Positive. Other evaluation measures are calculated from the matrix such as recall, precision, accuracy and F1 Score.

With the help of the weighted average technique, combining all the results from the trained models and past work. A weight between 0 to 1 was assigned and multiplied by the respective models and the sum of the products was taken to achieve the final prediction. The steps followed for assigning weights are shown in algorithm 1.

Step 1: Weights are decided in accordance with the performance of every model trained.

Step 2: Weight for each model is calculated according to the given formula.

$$\text{weight} = \frac{x_i}{\sum_{i=0}^{n} x_i}$$

here, x_i is the model result.

Step 3: This weight is multiplied by every metric of each model, and the summation of all the results of models after multiplying with the weight will give us the weighted average.

Currently, we have trained five models on our dataset - Decision Tree, Random Forest, Naive Bayes, SVM, ANN, and calculated performance metrics for each model.

It is observed that a few models have given us exceptionally high-performance metrics, touching 99% in some cases.

VI. Result

When training classification models on highly biased data, results can be favored in accordance with the majority class. Effective data sampling methods were used to reduce this problem and generate an unbiased dataset. The classifications made by models are evaluated on multiple metrics rather than only looking at the accuracy of the classifications. A confusion matrix made up of F1 score, recall value, precision and accuracy is developed to measure the performance of the model. Confusion Matrix is created with the help of 4 values namely True Positive, True Negative, False Positive, False Negative. Precision can be interpreted as the number of true positive instances out of all the positive instances predicted by the classification model. This can be calculated as:

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (1)$$

Recall is the number of true instances predicted by the model to the actual number of instances belonging to the True class. This can be calculated as:

$$\text{Recall} = \frac{TP}{TP+FN} \qquad (2)$$

Taking the harmonic mean of precision and recall we get the value of F1 score. The individual performance of all the models is given in the table below.

TABLE I. PERFORMANCE ANALYSIS OF DIFFERENT MODELS

	Accuracy	Precision	Recall	F1-Score
Decision Trees	0.9964	0.909	0.667	0.769
Naive Bayes	0.9105	0.846	0.733	0.785
ANN	0.996	0.9	0.75	0.81
SVM	0.996	0.9	0.75	0.81
Random Forest	**0.9992**	**0.916**	0.733	0.814
Weighted average	0.9914	0.9114	**0.753**	**0.8196**

Comparing the performance metric of various models, we can conclude random forest to be the best performing model. To produce more accurate results as compared to individual models, we have used the weighted average ensemble. Using this technique, we have produced results with an accuracy of 99.14%, precision of 91.14% and recall and F1 Score crossing the 75.3% and 81.96% mark respectively. Comparing the best performing individual model with our weighted average model:

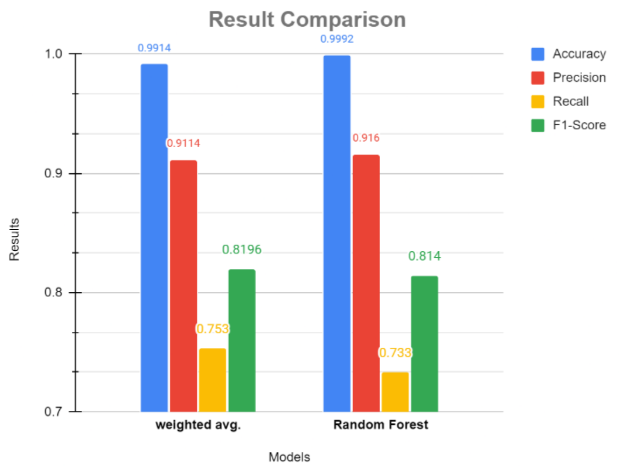

Figure 6. Comparing the performance metrics of the best performing model and the hard voting ensemble model

After applying all the ensemble methods to results produced by individual models, getting improved results in terms of precision, and a more generalized and less overfit model. 100% precision means that there are no false positives in the predictions of the hard voting ensemble.

VII. Conclusion

Here the experiment was able to note out and create a successful system that can detect a malicious tenant on a virtual machine in a multi-tenant distributed system by using ensemble weighted averaging and majority voting techniques. Basic problems that can occur in these types of highly biased data sets can be resolved by using over and under-sampling techniques which help standardize the dataset. Coupling and ensembling all models together gives us results that are not prone to overfitting. Attackers prefer multi-tenant attacks as they affect multi-tenant distributed storage architectures. In the future, these results can be improved by using more negatively biased data samples and using advanced techniques in machine learning like multi-node artificial neural networks. As day-to-day data is being collected to improve and maintain logs, immense amounts of new opportunities can be explored in the field of cyber security. This will lead to better safety in distributed storage systems by decreasing the possibility of a cyber-attack by detecting malicious tenants early on.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their very helpful comments and feedback to improve the manuscript. We are deeply thankful to all these individuals for their contributions, which have significantly enriched our project and academic journey.

REFERENCES

[1] P. Patil and R. Ingle, "Meta-ensemble based classifier approach for attack detection in multi-tenant distributed systems," 2020 International Conference for Emerging Technology (INCET)

[2] Semwal, P., Handa, A. (2022). Cyber-Attack Detection in Cyber-Physical Systems Using Supervised Machine Learning. In: Choo, KK.R., Dehghantanha, A. (eds) Handbook of Big Data Analytics and Forensics. Springer, Cham

[3] Singh, A.K., Chhabra, S., Gupta, R. *et al.* A Reliable Client Detection System during Load Balancing for Multi-tenant Cloud Environment. *SN COMPUT. SCI.* **4**, 86 (2023).

[4] Inayat U, Zia MF, Mahmood S, Khalid HM, Benbouzid M. Learning-Based Methods for Cyber Attacks Detection in IoT Systems: A Survey on Methods, Analysis, and Future Prospects. Electronics. 2022

[5] D. Du *et al.*, "A Review on Cybersecurity Analysis, Attack Detection, and Attack Defense Methods in Cyber-physical Power Systems," in *Journal of Modern Power Systems and Clean Energy*, vol. 11, no. 3, pp. 727-743, May 2023

[6] Q. Li, J. Zhang, J. Zhao, J. Ye, W. Song and F. Li, "Adaptive Hierarchical Cyber Attack Detection and Localization in Active Distribution Systems," in *IEEE Transactions on Smart Grid*, vol. 13, no. 3, pp. 2369-2380, May 2022

[7] M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, P.N. Suganthan, Ensemble deep learning: A review, Engineering Applications of Artificial Intelligence, Volume 115, 2022, 105151, ISSN 0952-1976.

[8] Wayne J. Brown, Vince Anderson, Qing Tan, "Multitenancy - Security Risks and Countermeasures", 2012 15th International Conference on Network-Based Information Systems

[9] D. Bekerman, B. Shapira, L. Rokach, A. Bar, "Unknown malware detection using network traffic classification", 2015 IEEE Conference on Communications and Network Security (CNS).

[10] F. Iglesias, T. Zseby, "Analysis of network traffic features for anomaly detection", Mach. Learn.101(1-3), 59–84 (2014).

[11] M. Z. Rafique, P. Chen, C. Huygens, W. Joosen, "Evolutionary algorithms for classification of malware families through different network behaviors", Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Pages 1167-1174.

[12] N. Stakhanova, M. Couture, A. A. Ghorbani, "Exploring network-based malware classification", 2011 6th International Conference on Malicious and Unwanted Software

[13] S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang, "Domain adaptation via transfer component analysis", IEEE Transaction on Neural Netw.22(2), 199–210, 2011

[14] X. Shi, Q. Liu, W. Fan, P. S. Yu, R. Zhu, "Transfer learning on heterogenous feature spaces via spectral transformation", IEEE International Conf. on Data Mining (ICDM), 2010.

[15] M.R. Aswin, M. Kavitha, "Cloud intelligent track - Risk analysis and privacy data management in the cloud computing", 2012 International Conference on Recent Trends in Information Technology

[16] W. Dai, Q. Yang, G. -R. Xue, Y. Yu, "Boosting for transfer learning", 24th International Conf. on Machine Learning(ICML) 2007

[17] Michael Hilker, Christoph Schommer, "Description of bad-signatures for network intrusion detection", n Conf.s in Research and Practice in Information Technology Series, vol. 54, ACSW, 2006, pp. 175–182

[18] T. Evgeniou, M. Pontil, "Regularized multi–task learning", SIGKDD International Conf. on Knowledge Discovery and Data Mining(KDD), 2004.

[19] A. Valdes, K. Skinner, Adaptive, "Model-Based Monitoring for Cyber Attack Detection", International Workshop on Recent Advances in Intrusion Detection, Berlin, Heidelberg, 2000.