NIDS Based False Positive Alert Screening Approach Using Machine Leaning

Prof. Priyanka R. Raval*

*Computer Engineering Department, Government Engineering College, Rajkot, Gujarat, India
priyankaraval.gec@gmail.com

***Corresponding Author:** Prof. Priyanka R. Raval
*Computer Engineering Department, Government Engineering College, Rajkot, Gujarat, India
priyankaraval.gec@gmail.com

Abstract— One of the most crucial security challenges of the modern day is detecting cyber attacks, and a network monitoring system to detect any intrusion commonly known as Network Intrusion Detection System (NIDS) are essential for this. Various machine learning approaches have been used in numerous research to build robust NIDS that can identify cyberthreats. While the majority of NIDS research focuses on developing novel AI/ML models to increase classification/detection accuracy, every model generates a percentage of false positive (FP) alarms in the real world. The mechanism for handling FP alarms is rarely covered in studies. Managing the volume of FP alarms on a busy network takes a lot of time for security staff. Automation of FP alert filtering is crucial because of this. In this research, we leverage kernel density estimation to present an automated FP alert filtering technique. Regardless of the NIDS that is in place, our suggested plan can help security staff with the alert verification process. Our tests demonstrate that, in terms of error ratio, our suggested system performs 32% to 60% better than alternative algorithms. Additionally, our suggested plan cuts down on the alert verification process's duration by 73%.

Keywords: Cyber security, Intrusion Detection System (IDS), Alert verification, Alert prioritization, Alert Fusion, Machine learning (ML), False Alert, Adaptive Filtering.

I. INTRODUCTION

An essential function of network intrusion detection systems, or NIDS, is to identify various network intrusion and threats within the internal network and against the external threat sources to the system network. Network administrator or a monitoring system can rely on the configured firewall rules to stop network conversations that causes NIDS alarms in response to external network risks. Regarding the internal network danger, security staff should keep a tight eye on network traffic in order to spot hosts that have been affected by malware and take additional steps to remove it.

The two primary NIDS [5][6][7][8][11] types that are now in use are rule-based and anomaly-based. Since detection rules for rule-based NIDS are meticulously built by security specialists upon incident summarization, they are dependable solutions for threat detection. Although rule-based systems generate fewer FP alerts, the process of building rules takes time, and the rules are limited to detecting known network threats. However, anomaly-based network intrusion detection systems (NIDS) are designed to identify unfamiliar network threats by identifying abnormalities in network data. The drawback of anomaly-based solutions is that they generate more FP alerts compared to the rule-based solutions.

Due to the advancements in applications of machine learning approaches, numerous studies have been conducted [1][2][3][4][9][10][12] add AI/ML techniques to NIDS; therefore, they will be referred to as ML based intrusion detection approach. Depending on the goals, ML based intrusion detection approach might be anomaly- or rule-based. While retaining a trustworthy detection accuracy, ML based intrusion detection approach eliminate the laborious job of creating rules, as efficiency is determined by assessing rate of FP and FN alerts which defines the error rate of the approach.

When FN alerts are displayed, there is a problem with FP alert filtering. FP alarms persist even though ML based intrusion detection approach can obtain results (85% to 99%) with a relatively good accuracy in the evaluation of test datasets. It will therefore be necessary for network security professionals to verify alerts. The number of alarms generated by ML based intrusion detection approach in a busy network might be astounding. In order to save up security staff time to verify actual positive warnings, we therefore devise an automated system that can help decrease the amount of FP alerts.

In this research, we use kernel density estimator (KDE) trying to solve the FP alert filtering problem. KDE is based on the anomaly score that we initially compute for every host that causes an alert. KDE is used to identify hosts that are maliciously infected and ones that are not. Security staff can concentrate on the alarms that are brought on by infected hosts due to malware propagation once they have been discovered. Workload for alert verification is thereby decreased.

II. RELATED WORK

As far as we are aware, no relevant work has been done to directly solve the FP filtering of the alert issue. We will concentrate on introducing various ML based intrusion detection approach and related accuracy in this section, as the problem of FP filtering of the alert is related to the accuracy that these systems create.

Based on AI/ML techniques, supervised, unsupervised, and hybrid ML based intrusion detection approach can be classified into three categories. Malicious network traffic must be accurately classified, regardless of the difficulty in gathering data, which is the primary objective of ML based intrusion detection approach. Classification accuracy is therefore a crucial indicator.
One benefit of the supervised machine learning approach is its high accuracy. Labelled data is necessary, though, and additional testing is needed to ensure that malicious conversations in the network traffic are not included in the training data sets is accurately detected in cases of unknown threat. In order to create a malicious network traffic detection model, this study area uses feature extraction or selection on labelled data samples. The goal of Narang et al. [10] was to differentiate safe P2P software from P2P botnet activity. The writers combined 5-tuple network flows into 2-tuple chats, then took characteristics out of the dialogues. In order to train a detection model, the collected features were fed into supervised machine learning methods such as BayesNet, J48, and Adaboost with REP tree. When assessing the model's ability to classify data samples, it was 95% correct. Using the NSL-KDD dataset, Aljawarneh et al. [1] suggested a hybrid approach in feature-selection and ensemble classifier method. When it came to testing data sample categorization, our model was 99.81% accurate.
Unsupervised learning techniques have the advantage of not requiring labelled data and being adept at identifying unknown risks. The drawbacks of unsupervised techniques include reduced detection accuracy in comparison to supervised ones and ad-hoc threshold value configuration for the classification of harmless and dangerous network conversations. In order to cluster multi-resolution flow (network flow in various time intervals), Casas et al. [3] presented a clustering technique that combined sub-space clustering, density-based clustering, and evidence accumulation clustering. A predetermined threshold was employed to identify the abnormality of the network flows, which were ranked according to their degree of irregularity. Considering that the model produced categorical predictions, its accuracy ranged from 50% to 100% when assessing data sample classification. Prior to network traffic clustering, Bhuyan et al. [2] used a feature selection approach based on generalized entropy and mutual information. To find anomalies in the network flows, the authors employed an outlier score function and a user-specified threshold value on the clustering outcomes. Considering that the model produced categorical predictions, its accuracy ranged from 78% to 99% when assessing data sample classification.
While improving detection accuracy in unsupervised algorithms, a hybrid approach supplements the requirement for labelled data fields in supervised ML approaches. Even if a supervised method is employed to identify harmful network activity, more verification is still needed to determine the accuracy of detecting unknown threats. By using clustering techniques—the cluster centroid and nearest neighbour—Lin et al. [9] were able to compute two distances that were employed as features and train a supervised model for the purpose of identifying malicious network flows. When evaluating the model for data sample classification, the accuracy rate was 99.76%. For feature extraction, reference [12] suggested the nonsymmetric deep autoencoder (NDAE). In addition to limiting the feature dimensions, NDAE banned handcrafted feature engineering.
A random forest model was trained with the retrieved features in order to classify malicious network flows. When examining the model's ability to classify data samples, its accuracy was 85.42%.

III. The Proposed Approach

When FP alerts are generated, we want to develop an automated way to reduce the amount of work that needs to be done by humans to verify AI/ML-based NIDS signals. Stated differently, our goal is to develop an algorithm—which we refer to as the FP filtering algorithm—that will lower the proportion of alerts that are deemed FPs.
Preprocessing and the algorithm are the two subsections that make up our suggested approach. Alerts are converted into host-based anomaly ratings through preprocessing. In the algorithm portion, a FP filtering algorithm for detecting hosts infected with malware is provided. Security staff can now concentrate on confirming the alarms that the malware-infected hosts are generating.

A. Preprocessing

Our suggested strategy aims to reduce the workload associated with alert verification by filtering out ML based intrusion detection approach FP alerts. Thus, as a prerequisite, we will require an ML based intrusion detection approach to generate alerts.
Three different forms of ML based intrusion detection approach are known to exist, based on the corresponding work in section II; regardless of the techniques used in these types, classification accuracy is a crucial metric for assessing the models. As a result, we choose a model for network conversation preprocessing based more on accuracy than technique. We first summarize the relevant literature [2][12][3][10][4][9], and then we build a 96% accurate random forest model for the classification of harmful network conversation. We will so receive roughly 4% of FP and FN warnings.
We utilize Su et al. [13] to determine the anomalous score for every host once all of the alerts have been gathered. [13] used the dendrogram of the clustering result to compute the anomaly score after applying the

hierarchical clustering approach to the gathered alerts. Based on the inclination that harmless hosts are simple to aggregate early into the common groups since these hosts issued less alarms, the anomaly scoring system was developed. However, since they produced more alarms and differed from harmless hosts, infected hosts would be grouped into clusters later. An anomaly score that is near to 0 indicates a potentially harmless host, whereas a score that is close to 1 indicates the reverse. An anomaly score ranks the host according to the likelihood of malware infection.

B. Alert Filtering Algorithm

The classification of hosts infected with malware is guided by a threshold value of anomaly scores. On the other hand, choosing the threshold point might be subjective and takes practice. As a result, we provide an automated method to determine the threshold.
It has been determined that the KDE methodology is appropriate for computing threshold values. Based on our observations, hosts that are not afflicted with malware likely to have similar anomaly ratings to those of harmless hosts. In accordance with our findings, we use KDE to project the anomaly score density and measure the difference between two densities to get a threshold value that will separate hosts free of malware from those that are infected.

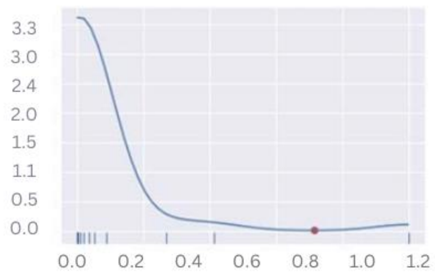

Figure 1: KDE findings plotted on a line

We apply the minimal search locally on the line graph that KDE generates (Figure 1) to get the threshold point. The density distribution of anomaly scores is computed by KDE and displayed as a line graph. Density is shown by the Y-axis, and anomaly scores are represented by the X-axis. Populations are said to consist of peaks. In the best scenario, a threshold value is immediately discernible if there are only two peaks (indicating a distinct difference between hosts infected with malware and hosts that are not) or if there is no local minimum (signifying that all hosts are deemed innocent). In the event that several local minimums are given, the threshold point is determined by taking the median of the local minimums.

C. Dataset

A combination of sampled malicious network conversation and actual network conversation makes up the experimental dataset. We gather four hours of actual network conversation from each of the forty hosts that make up our faculty. Consequently, more than 12,000 network flows are gathered. The CTU-13 [4] dataset is the source of the sampled malicious network conversation. The sample is made up of seven different malware samples, which represent a variety of botnet malware: Neris, Rbot, Virut, Menti, Sogou, Murlo, and Nsis.ay. We mix in sampled malicious network conversation and randomly choose hosts from actual network conversation to simulate malicious network activity.
Once the relevant work has been summarized in section 2, we choose and train an ML model. When it comes to classifying malicious network conversation, the ML model is roughly 92% accurate. Our dataset causes the machine learning model to produce approximately 1000 FP alerts, or FP network flows. If security staff works eight hours a day, there will be a spike in FP alerts above two thousand, which will be challenging for any staff member to handle without assistance from a FP filtering approach.

D. Experimental Results

The four false alert filtering techniques under comparison are used to categorize and arrange the trial outcomes. Based on the suggested scheme performance, the comparative findings are divided into three categories: Better (if there are less FP/FN/Error alerts), Neutral (if there are equal FP/FN/Error alerts), and Worse (if there are more FP/FN/Error alerts).

i. Scenario A: One compromised host One compromised host is the source of the malicious network conversation in this scenario. Seven scenario A examples are accessible for the studies because there are seven types of malwares available.
ii. Scenario B: Two compromised hosts Two compromised hosts are the source of the malicious network conversation in this scenario. Various malware infections affect the malevolent hosts. There are 21 scenario B examples available for the experiments because there are 7 malwares available.
iii. Scenario C: Three compromised hostsThree compromised hosts are the source of the malicious network

activity in this scenario. Similar to case B, distinct malware infects the hosts. 35 scenario C examples are offered for the trials, with 7 malwares accessible.

iv. Scenario D: 0 Host Infection The network conversation in this case is entirely innocuous. Consequently, there is only one case for the experiment in this scenario.

The four scenarios (A-D) have improved by 42%, 60%, 33%, and 32%, in that order. Considering that our suggested strategy produces more FP in scenario D, it is only 34% better; In addition to eliminating FPs, an effective FP filtering algorithm should work to keep FNs to a minimum. Overall, our suggested plan produces a few more FP warnings than the competition, but we decrease a lot more FN notifications. Compared to the techniques mentioned, our suggested scheme is more efficient under the evaluation metric assumption.

IV. CONCLUSIONS

As previously indicated, ML based intrusion detection approach is excellent at spotting malicious network conversation. Nevertheless, no ML based intrusion detection approach could guarantee a 100% detection accuracy rate in the field. To identify risks, security staff will need to rigorously verify ML based intrusion detection approach notifications. Due to time constraints, security staff is only able to confirm a few notifications every day. We suggest a FP filtering algorithm to help with the alert verification task in order to reduce the workload associated with it. Our tests reveal that, when compared to four common FP filtering methods, our suggested strategy reduces FP alarms by 32% to 60%. By implementing our suggested plan, security staff can reduce alert verification time by 75%.

REFERENCES

[1] Aljawarneh, S., Aldwairi, M., and Yassein, M. B. 2018. Anomaly-based Intrusion Detection System through Feature Selection Analysis and Building hybrid Efficient Model. *Journal of Computational Science*, vol. 25, 2018, pp. 152– 160.

[2] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. 2016. A Multi-Step Outlier-based Anomaly Detection Approach to Network-Wide Traffic. *Information Sciences*, vol. 348, 2016, pp. 243–271.

[3] Casas, P., Mazel, J., and Owezarski, P. 2012. Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. *Computer Communications*, vol. 35, Issue 7, 2012, pp. 772–783.

[4] Garca, S., Grill, M., Stiborek, J., and Zunino, A. 2014. An Empirical Comparison of Botnet Detection Methods. *Computers & Security*, vol. 45, 2014, pp. 100–123

[5] Gu, G., Perdisci, R., Zhang, J., and Lee, W. 2008. BotMiner:Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Proceedings of the 17th Conference on Security Symposium (SS'08)*. USENIX Association, Berkeley, CA, USA, pp. 139–154.

[6] Hu, W., Hu, W. and Maybank, S. 2008. AdaBoost-Based Algorithm for Network Intrusion Detection. In I*EEE Transactions on Systems, Man, and Cybernetics, Part B(Cybernetics)*, vol. 38, no. 2, April 2008, pp. 577–583.

[7] Kim, G., Lee, S., and Kim, S. 2014. A Novel Hybrid Intrusion Detection Method Integrating Anomaly Detection with Misuse Detection. *Expert Systems with Applications*, vol. 41, Issue 4, Part 2, 2014, pp. 1690–1700.

[8] Koc, L., Mazzuchi, T. A., and Sarkani, S. 2012. A Network Intrusion Detection System based on a Hidden Naïve Bayes Multiclass Classifier. *Expert Syst. Appl*. 39, December 2012, pp. 13492–13500.

[9] Lin, W. C., Ke, S. W., and Tsai, C. F. 2015. CANN: an Intrusion Detection System based on Combining Cluster Centers and Nearest Neighbors. *Knowledge-Based Systems*, vol. 78, 2015, pp. 13–21.

[10] Narang, P., Hota, C., and Venkatakrishnan, V. 2014. Peershark: Flow-Clustering and Conversation-Generation for Malicious Peer-to-Peer Traffic Identification. *EURASIP Journal on Information Security 2014*, 2014, Article 15.

[11] Roesch, M. 1999. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration (LISA '99)*. USENIX Association, Berkeley, CA, USA, pp. 229–238.

[12] Shone, N., Ngoc, T. N., Phai, V. D. and Shi, Q. 2018. A Deep Learning Approach to Network Intrusion Detection. In *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, February 2018, pp. 41–50.

[13] Su, Y. H., and Tzeng, W. G. 2017. The Forward-Backward String: A New Robust Feature for Botnet Detection. M. S. Thesis, CS Dept., NCTU, Hsinchu, Taiwan ,2017, Accessed on: July 2019, Available: http: //etd.lib.nctu.edu.tw /cdrfb3/record /nctu /#GT070456522

DOI: https://doi.org/10.15379/ijmst.v10i3.3789