A Multi-model Rouge Nodes Detection System for Fog Computing

Thanh Duc Bui*
Dept. of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, USA
tbui@uark.edu

Brajendra Panda
Dept. of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, USA
bpanda@uark.edu

***Abstract*—With the increasing speed of advance of both Internet of Things (IoT) and the demands of utilizing IoT devices in different locations with high compatibility, fog computing has been foreseen as a sustainable solution. However, new challenges related to fog computing's cyber security also emerge. Among various cyber-attacks on the fog computing system, one standard method is to implement corrupted internal nodes that allow malicious/compromised access from attackers, threatening the confidentiality of users' information and the system's performance. In this paper, we propose a method to quantify the distribution of each attribute value from historical access logs from each node within a fog computing system. We also utilize an unsupervised machine learning method to separate nodes into smaller clusters with respect to different models to improve the forecasting ability of the system. Finally, our models are evaluated over simulated data using real access logs. Our experimental results illustrate that our proposed method achieves some advanced performance over applying an overall classification, even using the same classification machine learning methods and dataset.**

***Index Terms*—Cloud Computing, Cyber Security, Data Mining, Machine Learning**

I. Introduction

IoT has been hurrying from an idea to reality. From 0.8 billion devices in 2010; it has been reported that 10 billion IoT devices are in service as of 2019. Moreover, 30.9 billion IoT devices have been predicted by the end of 2025 [1]. While this is a booming technology sector, IoT devices are prime targets for cyber-attacks due to their lack of limitation of storage and processing capacity [2]. According to a report from IBM in 2022, roughly 45% of data breaches are cloud-based storage [3]. The report also states that the U.S. has lost twice as much money due to the breaches as the rest of the world just within 2022, and healthcare had the highest number of attacks among other sectors. Even the giant technology corporations are not immune from these attacks. For example, Microsoft has been attacked by Lapsus$, where a significant amount of data, including some source code for Bing, Bing's Map and Cortana have been leaked from their Azure DevOps server [4]. On the other hand, it is not only the company's projects that are under the threat of being attacked while being archived on their cloud server; customers' and users' identity information are also being compromised. One example is the Medibank data breach, where over 9 million personal data of customers were shared on the dark web [5]. The severity of this case has affected not only their current customers but Medibank's former and international clients. The attacks are not limited to corporations but also government agencies. As of July 2022, a leak of over one billion citizens' records from the Shanghai police database has been revealed [6]. These examples show the emergency of a countermeasure toward the growing risk of exposure of personal and sensitive data as the number of cloud-based implementations among technology firms and agencies soars.

Meanwhile, in recent years, the paradigm of cloud-based computing is shifting toward fog computing as a new solution for reducing communication time between IoT devices and the central cloud. The fog platform serves as an intermediate extension for the cloud computing service. Fog architecture enables higher flexibility of the cloud-based service with mobility, a vastly increased number of users, and lower latency [7]. Conversely, fog computing also brings a new set of problems. Since several fog nodes are filled in the fog platform, the system is turning into a distributed structure. While this arrangement is effectively easing the heavy data stream, it also comes with exploitable flaws due to a gain in the number of access points to the system [8]. Cybersecurity researchers are intensively studying these emerging issues, and many state-of-the-art security measures are proposed. However, most of these approaches focus on attacks performed by outsiders [9]. According to a report published by Ponemon Institution in 2018, there were 3,269 insider attacks out of 159 studied organizations within one year, with an average cost of over $600,000 per attack [10]. In the report in 2020 by the same institution with 204 organizations show a total of 4,716 insider incidents. The average cost per incident had escalated to over $750,000 per incident, which is over 25% increase in loss per attack in 2 years [11]. These statistics show the profound impact of insider attacks on any system and indicate that they should not be neglected. Fig. 1 illustrates the increase in average loss due to insider attacks over various activities.

Due to the severity of imminent security threats of internal attacks, cyber security researchers have recently focused on solutions. Among those commonly known internal threats currently under the scope, man-in-the-middle attacks can potentially be a regular type of risk for fog computing

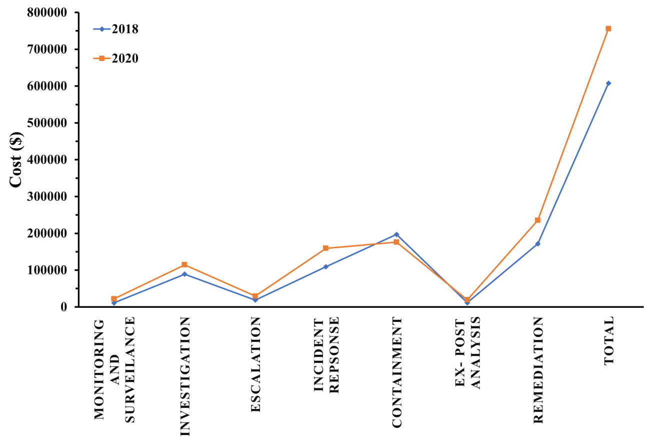

Fig. 1: Revenue Loss due to Insider Attack over Different Activities

systems [12]. The idea of man-in-the-middle attack is that a perpetrator disguises as a trusted source of connection to collect information from victims. In fog computing, a man-in-the-middle can function similarly to other fog nodes within the system. However, it also acts as a gathering point to collect users' data, a backdoor for malicious accesses, or a launchpad for attacking the system. There can be more than one rouge node planted in a fog computing system to obstruct the detecting ability of the system. In this regard, we propose a method utilizing attributes from historical access logs from each node to distinguish rouge ones. Section II overviews our contribution and defines the objectives and parameters we use in this paper. In Section III, we propose our method to approach a solution for fog computing internal attacks. In Section IV, we experiment using our proposed method to demonstrate the effectiveness of our feature extraction and multi-model approaches. Finally, we discuss limitations and possible future expansion for our research in Section V.

II. Background Work

There have been noticeable approaches toward rouge node detection in the past, but these methods are restricted due to their dependency on trust measurement upon fog nodes system [13]. As an approach to mitigate faulty trust factor due to corrupted fog nodes, Zahra et al. proposed a system using a defined trust value and fuzzy logic to determine a set of malicious nodes [14]. This model allows more tolerance to fraud nodes by defining an adapting trust benchmark between nodes, high versatility with different fog computing systems, and robustness against various forms of attack, such as black holes, collusion, DDoS, and meticulous methods. However, the model is sensitive to the threshold settings of truth values between nodes and requires an alteration in the hardware level of the system. On the other hand, we obtain ideas from the work on the attribute-based access control (ABAC) model of [15] where they can capture an entity's behavior through its attributes and rule out exception under different circumstance. However, the ABAC method is effective with only a single access log, while each fog computing node generates a different set of access logs. Therefore, we need some method to capture the unique properties of each node.

To the best of our knowledge, there has yet to be any work that integrates the ideas of rouge node detection of a fog computing system and access logs from each node. By combining by the idea of an ABAC system and malicious node detection within a fog computing system, as well as the Bag of Words (BoW) model from Information Retrieval, we propose an idea of two distinct phases [16]. We separated each access log into three parts: request attributes, resource ID, and system decision. In the first phase, access logs from each node will accumulate the frequency of resource usage from different resources from the system. Using the BoW model, each node is associated with a vector of the acquired frequency with the resource ID. At the end of this phase, we use the collected information from this process to separate nodes into different clusters through the similarity between their resources' requests. In the second phase, we extract the distribution of each attribute's value toward system decision as node's features. To quantify the distribution, we define a trust value using the frequency of each attribute associated with each decision within an individual node. We perform this process on every node within the fog computing system. Finally, using the results from both phases, we construct a multi-model system to classify fog nodes using extracted features and clustering values.

A. Our Contribution

Since our focus is to capture We present a multi-model rouge node detection for rouge nodes within a network of fog computing, and our contributions are:

- First, we separate nodes into different clusters using the k-means clustering method. We perform a clustering algorithm on the fog computing system using the resource request behavior of each node. We pool all possible resources from the network and assign the corresponding frequency of resource usage per node.
- Then, we perform a feature extraction method using our proposed trust computation formula for each node to capture the value distribution concerning the system decision. This method allows our classification models to differentiate between normal fog nodes (benign) and intrusive nodes (malicious).
- Finally, we conduct experiments on implementing our proposed model to demonstrate its practicality and efficiency.

B. Problem Definition

Throughout the paper, we use user attributes, objects, and decisions to refer to each requesting user's attributes, the resource/object they request to operate, and the system's decision on which action to perform with a given request. We will define each as a set and how they interact to make the definition proper.

Definition 1: Let A is a set of all attribute of a user u, for a given attribute $a \in A$, V_a is defined as a range of all possible values for a,

Definition 2: We define a request r as $r\langle u, o\rangle$ as request made by user u, each request contains user's attributes a and the object/resource o they wish to grant access to: For example, a username John is an Engineer from the Retail Department wishing to have access to use sector 4 of the supercomputer can be represented as:

r $\langle\{$*Name: John, Pos: Engineer, Dept: Retail*$\}$, *Sector 4*$\rangle$

To simplify the request access, we will keep them a single value (atomic). However, the process remains similar for multiple values since they can be separated into many single-value requests.

Definition 3: A tuple of authorization showing decision d for user u concerning object o is defined as $t\langle r, d\rangle$, for r, is a request notation from Definition 2, and d is a value from a set of a decision given regarding the request.

Definition 4: A log file L_n is a collection of authorization tuple t that logged within node n from a fog computing system.

Definition 5: a frequency of $F_{L_n}(v_i)$ is a quantity of appearance of a value $v_i \in V$ for each access log L_n of node n. Subsequently, $F^+_{L_n}(v_i)$ is the frequency of v_i where the tuple decision d is permitted, and $F^-_{L_n}(v_i)$ is the frequency of v_i where the tuple decision is denied.

Definition 6: A classification value c_n of a node n is a predicted class of node n based on a collected log file L_n. Also, c^+ is a set of nodes n labeled malicious, and c^- is the set of true benign nodes n.

Table I summarizes defined notations used within this paper.

C. Evaluation Metrics

Our primary metrics to evaluate the performance of our proposed classifier is how accurately predicted classes of nodes match with original labels from our simulated dataset. That means our predicted class for each node within the simulated system should be similar to the provided class value from the experimental dataset. For example, if our predicting classifier detects a node as malicious from the simulated dataset, that node should also be malicious. If a node is malicious, but our classifier defines it as benign, it will be scored as False Negative. We define True Positive, False Positive, True Negative, and False Negative rates as below:

True Positive rate: *Given an access log L_n and a classified value c_n of node n, the true positive rate of c_n given log L_n denoted as $TP_{c_n|L_n}$ represents the portion of the predicted malicious nodes for which the nodes are labeled as malicious*, as defined:

$$TP_{c_n|L_n} = \frac{|c_n \in c^+|n = malicious|}{|c^+|} \tag{1}$$

Where $|s|$ is the cardinality of set s

False Positive rate: *The false positive rate of c_n given log L_n denoted as $FP_{c_n|L_n}$ represents the portion of the predicted malicious nodes for which the nodes are labeled as benign*, as defined:

$$FP_{c_n|L_n} = \frac{|c_n \in c^+|n = benign|}{|c^+|} \tag{2}$$

True Negative rate: *The true negative rate of c_n given log L_n denoted as $TN_{c_n|L_n}$ represents the portion of the predicted benign nodes for which the nodes are labeled as benign*, as defined:

$$TN_{c_n|L_n} = \frac{|c_n \in c^-|n = benign|}{|c^-|} \tag{3}$$

False Negative rate: *The false negative rate of c_n given log L_n denoted as $FN_{c_n|L_n}$ represents the portion of the predicted benign nodes for which the nodes are labeled as malicious*, as defined:

$$FN_{c_n|L_n} = \frac{|c_n \in c^-|n = malicious|}{|c^-|} \tag{4}$$

The *precision* and *recall* are calculated as follows:

$$P_{c_n|L_n} = \frac{TP_{c_n|L_n}}{TP_{c_n|L_n} + FP_{c_n|L_n}} \tag{5}$$

$$R_{c_n|L_n} = \frac{TP_{c_n|L_n}}{TP_{c_n|L_n} + FN_{c_n|L_n}} \tag{6}$$

From defined rates for positive and negative, our classifier *accuracy* of predicted c_n given access log L_n for node n is calculated as below:

$$A_{c_n|L_n} = \frac{TP_{c_n|L_n} + TN_{c_n|L_n}}{TP_{c_n|L_n} + FP_{c_n|L_n} + TN_{c_n|L_n} + FN_{c_n|L_n}} \tag{7}$$

Since accuracy can be misleading when the dataset's distribution among classes is heavily leaning toward one class [17], we also use *F-score* to evaluate the performance of models:

$$F_{c_n|L_n} = 2 \times \frac{P_{c_n|L_n} \times R_{c_n|L_n}}{P_{c_n|L_n} + R_{c_n|L_n}} \tag{8}$$

For our approach, we are more conscious about the malicious nodes; hence *recall* is our key metric, followed by *precision*. *F-score* and *accuracy* is used to overview our feature extraction performance but are not considered as important as the other two.

III. Proposed Approach

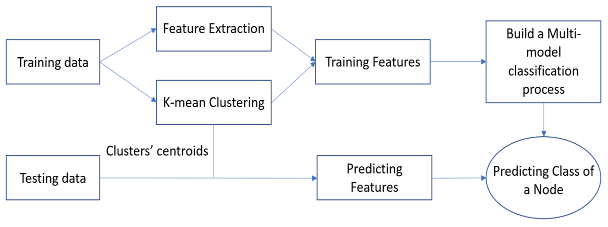

Fig. 2: Pipeline for modelling with proposed method

The steps of our proposed classification process are described in Fig.2. Since our collected data are node access

TABLE I: Notations

Notation	**Definition**
U, O, D	Set of users, objects, and decisions
$A = A_U \cup A_O \cup A_D$	Set of all attributes
V_U, V_O, V_D	Attribute Range
$r\langle u, o\rangle$	Access request
$t\langle r, d\rangle$	Authorization tuple, showing decision d with respect to request r
L_n	Access log of node n
$F^+_{L_n}(v_i)$	Frequency of permitted decision of $v_i \in V$ in access log L_n of node n
$F^-_{L_n}(v_i)$	Frequency of denied decision of $v_i \in V$ in access log L_n of node n
$F_{L_n}(v_i) = F^+_{L_n}(v_i) + F^-_{L_n}(v_i)$	Frequency of of $v_i \in V$ in access log L_n of node n
c_n	Predicted class of a node n
c^+	Set of true malicious nodes n
c^-	Set of true benign nodes n
$TP_{c_n\|L_n}$, $FP_{c_n\|L_n}$,$TN_{c_n\|L_n}$,$FN_{c_n\|L_n}$	True Positive, False Positive, True Negative, False Negative
$A_{c_n\|L_n}$	Accuracy rate, proportion of nodes n that are correctly classified
$R_{c_n\|L_n}$	Sensitivity, proportion of malicious nodes correctly classified from set of true malicious nodes
$P_{c_n\|L_n}$	Precision, proportion of malicious nodes correctly classified from set of predicted malicious nodes
$F_{c_n\|L_n}$	F-score

logs, we cannot efficiently classify the whole node using the raw input data. Hence, a pipeline is proposed to remedy the issue. First, we have two distinct datasets for training and testing purposes. Afterward, we perform a feature extraction process and concurrently apply a k-mean clustering algorithm on the training dataset. For feature extraction, we aim to determine the distribution of the system's decisions based on each attribute's value from the individual node's access log. We attempt to use our defined measurement to quantify those distribution's values and associate them with each node to understand individual nodes' behavior and pattern for classification. We apply k-means clustering algorithms based on the resource request pattern from each node. This setup is considered since nodes requesting similar resources should be close to each other. Finally, we combine the clustering information and features extracted from the two processes above for modeling. Access logs from each cluster are used to construct classification models for nodes' indications. By doing this, we expect our setup to consistently identify rouge and faulty nodes within a fog computing system.

Each testing node is queried by the frequency of requests for each resource; then, they are associated with the closest centroid passed from the above clustering process. Next, each node undergoes the same feature extraction process as training. Finally, trained models using train data are deployed according to the cluster labels to classify whether a node is malicious or benign.

A. Data Clustering

1) Cosine Distance

It would make more sense to cluster fog nodes based on how likely objects/resources will be requested into clusters. Hence, we use the set of all possible objects/resources as nodes' features for clustering V_o. Then we count the frequency of objects/resource requests from each L_n to use as a feature vector for each node. Finally, a k-means clustering algorithm using cosine distance is deployed to separate the node into separated clusters. Cosine distance has been widely used in information retrieval and Natural Language Processing (NLP) since it represents the similarity between different corpus without having issues of magnitude due to differences in size between each corpus [18]. This application fits well into our proposed model since each node contains vastly diverse records. However, we are only interested in the contents of resources that are being requested by each node. Therefore, a k-means clustering method using cosine distance will likely yield a more desirable result.

2) Number of Clusters (k)

One of the most challenging issues we face with the unsupervised clustering process is determining an optimal number of clusters, k. If the number of clusters is large, the clusters are fragmented into significantly minute ones and may be insufficient to construct a sub-model. Additionally, the processing time exponentially increases as the number of clusters increases. On the other hand, a small value for k also causes each node within a cluster to be diverse and less likely to provide details of their characteristic. One of the popular methods to select an optimal k is *Elbow Method* [19]. This method is built upon the distortion within each cluster, or the average of squares of distances between each node and its respective centroid. As the number of k increases, the distortion value decreases. However, the rate of change will diminish as k grows, and we can figure out the optimal value for k by observing where higher k does not result in a lower distortion value on the graph (i.e., the elbow point).

B. Features Extraction

We proposed a feature extraction process similar to the Bag of Words (BoW) model. Firstly, all possible values for any entity's attribute from all training nodes are accumulated. Then each unique value is assigned with our defined truth value for each. We calculate the truth value for each value in each node using the following equation:

$$T(v_i|L_n) = \begin{cases} \frac{F^{+}_{L_n}(v_i) - F^{-}_{L_n}(v_i)}{F_{L_n}(v_i)} & F_{L_n}(v_i) > 0 \\ 0 & otherwise \end{cases} \tag{9}$$

Each time a unique attribute value v_i scores a permitted decision in a log node n, we will reward v_i with a positive point toward the truth score and a negative point for v_i if a decision is denied. Afterward, we divide the final score for each value v_i in log node n by the total number of users associated with v_i in log node n to normalize the truth score. We use this to measure how each value i should be distributed in each node. This measure ranges from -1 to 1, with -1 being entirely unreliable and 1 being fully truthed. The value of 0 can either be interpreted as an absence of v_i in L_n or value v_i has an equally odd for either a positive or negative decision. Since the odds are evenly distributed in the latter case, it is not much different from a coin-toss decision. Therefore, it is more reasonable to assign with newly introduced value v_i to a L_n when deciding since it would not have impacted the truth distribution of each known v_i to L_n. On the other hand, this would enhance our feature extraction steps by providing an equal dimension for each node n.

IV. Experimental Evaluation

A. Dataset

We perform our experiments on real datasets provided by Amazon in the Kaggle competition, which contains the historical access log of employees in Amazon. The data documented over 30,000 access logs from Amazon, collected from 2010 to 2011 [20]. While the dataset is over a decade old, it is still relevant and being used to experiment with different ABAC models. There are different categories with related information for each employee requesting resources from Amazon's server. Overall, the dataset includes over 12,000 distinct users accessing 7,000 resources, and it is heavily skewed toward permitted access requests with a 1-to-16 ratio. This setup is ideal for generating our simulated data, as attacking accesses are extremely rare in a real scenario.

B. Model Selection

As mentioned in our proposed method, we have extensively searched for other models that utilize access logs to predict rogue fog nodes. However, to the best of our knowledge, there exists no other work on this area. Thus comparing our model against something that is unrelated would make it an unfair comparison. Hence, we have decided to build different models using state-of-the-art machine learning methods as a baseline to demonstrate the effectiveness of our feature extraction technique. Additionally, we will compare our multi-model approach against a single model. We select Support Vector Machine (SVM), Decision Tree, and Extreme Boosting Tree (XGB) for the experiment due to their simplicity and efficiency. We test our proposed multi-model method against each of these models, where they are trained on our feature-extracted data.

1) Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning model developed by Vladimir Vapnik and researchers from AT&T Bell Laboratory [21]. The idea of SVM is to find a multi-dimensional hyperplane that successfully separates different classes between data points using their multi features while maximizing margin distance from the plane to data points. SVM has been widely used in many research fields, notably in language classification [22], computer vision [23], and different fields of science. SVM has been proven to be a robust machine learning model; hence we choose to use this model with our feature extraction to demonstrate our proposed method better using the same feature extraction process [24].

2) Decision Tree

Decision Tree is also a popular supervised learning model due to its efficiency and simplicity [25]. Moreover, it requires little to none with data preparation, can handle various data types, has strong explainability, and is flexible with different conditions. For these reasons, the decision tree is a strong contender for a baseline model.

3) Gradient Boosting (GB) Tree

Gradient Boosting Tree is an ensemble learning method developed using the decision tree model. Instead of building a giant decision tree with several steps, this method unitizes boosting, which is to build weak decision trees based on the error of the previous iteration of weak decision trees [26]. Interestingly, it usually performs better than Random Forest, another ensemble learning method that uses the bagging method (multiple weak decision trees built parallel) [27]. Unlike a traditional decision tree, a gradient-boosting tree complicates the training process of a classification model to the point of loss of traceability and a vast increase in training time. However, the model still keeps itself robust and flexible. On top of that, it has been a strong contender for prediction and classifying tabular dataset [28].

C. Experimental Setup

To set up our experiment for the proposed model, we randomly split our dataset into testing and training sets, where 75% of the data will be used for training, and the remaining 25% will be used for testing. Then we generate an arbitrary number of nodes for each set, where each node selects a random number of records from their respective section (training vs. testing). Finally, we will annotate each node as benign or malicious using the following process:

- Benign node: we introduce a small fraction of noise to represent some error while data is collected where a subset of recorded decisions are altering. However, the noise fraction is relatively small and presented to test the robustness of our method when the system also includes some unknown factor that can cause false detection.

- Malicious node: access value is randomly switched based on a probability distribution, and each malicious node follows a different distribution.

We gather predicting attributes from the proposed feature extraction method and construct three models for each classification method (SVM, Decision Tree, and GB). These three are used as a baseline to benchmark our multi-model classifier. To optimize the baseline classifier, we use grid search with various parameters to tune our models. We also construct a multi-model classifier using SVM, Decision Tree, and XGB. However, each sub-model is built upon cluster data instead of the overall dataset. This step generates one model for each cluster. Then, we sort each testing entry into the respective cluster before acquiring its predicted tag using the sub-model associated with its assigned cluster.

We deploy a grid search for various hyper-parameters for each classification method to obtain the near-optimal settings for each baseline model and sub-model for our classifier. We construct SVM and Decision Tree model using *sklearn* package on *Python* [29], and *XGBoost* package for Gradient Boosting Tree [30]. To ensure each model's performance is consistent, we conduct a 5-fold cross-validation for every combination of hyper-parameter settings for each classifier. Since we focus on minimizing the error of wrongful classifying malicious nodes as benign, our focused criteria for grid search is *recall* score instead of *accuracy*. For SVM, we test with various cost (C) values ranging from 0.1 to 1000, gamma value from 0.0001 to 1, with rbf kernel. With the decision tree, we experiment with various criteria to build a tree branch (gini, entropy, and log-loss) and various max depth values. In the case of the GB tree, enormous amounts of hyper-parameters can be adjusted. Hence we only concentrate on sub-sample (from 0.75 to 1), column sample by tree (from 0.75 to 1), tree max depth (from 2 to 6), and learning rate (from 0.05 to 0.3). Finally, we construct our proposed classifier using the same settings for each cluster instead of applying it toward the whole dataset as base models are deployed. Our goal is to demonstrate that with similar settings, our method still performs better than similar models built on the overall dataset.

D. Result

We evaluate our proposal against baseline models using our simulated dataset. Table II shows the results of these experiments. Our model is only slightly better than the best baseline model (XGB). However, our main concern is the Recall ($R_{c_n|L_n}$) value since this evaluation metric determines how much false negative or the proportion of the actual number of malicious nodes that the trained model correctly classifies. For this category, our proposed classifying method outperforms every baseline model. On our second most important metric, Precision ($P_{c_n|L_n}$), our model slightly performs less than the best baseline model. This result means our method will likely put more benign nodes into the quarantine process than the XGB model, yet this problematic issue is more of a hassle than a threat to the system.

TABLE II: Comparison of Our Proposed Approach with Other Models

Model	$A_{c_n\|L_n}$	$R_{c_n\|L_n}$	$P_{c_n\|L_n}$	$F_{c_n\|L_n}$
SVM	90.32%	59.38%	73.08%	65.52%
Decision Tree	88.55%	58.33%	64.37%	61.20%
GB Tree	93.55%	66.67%	88.89%	76.19%
Proposed Multi-model	93.59%	68.75%	86.84%	76.74%

Another advantage of our proposed multi-model classifier is the ability to understand a threat for each cluster, as shown in table III. From the illustrated results, malicious nodes from **cluster 4** are likely to be identified; however, numerous benign nodes also behave like malicious nodes, affecting the sub-model precision for this cluster. On the other hand, the sub-model for **cluster 2** can accurately classify benign nodes while performing poorly with malicious nodes. The interpretation can also be applied to **cluster 5** and **cluster 7**¸ which also indicates why **accuracy** ($A_{c_n|L_n}$) is not a reliable metric to measure performance.

TABLE III: Best Results of Our Approach for each Cluster

	$A_{c_n\|L_n}$	$R_{c_n\|L_n}$	$P_{c_n\|L_n}$	$F_{c_n\|L_n}$
Cluster 0	94.12%	70.59%	92.31%	80.00%
Cluster 1	93.49%	80.00%	82.71%	82.76%
Cluster 2	93.28%	38.46%	100.00%	55.56%
Cluster 3	97.30%	66.67%	100.00%	80.00%
Cluster 4	89.80%	87.50%	63.64%	73.68%
Cluster 5	93.75%	50.00%	100.00%	66.67%
Cluster 6	92.42%	76.92%	83.33%	80.00%
Cluster 7	94.29%	66.67%	100.00%	80.00%

This problem can also be well determined as we plot the ratio of resources requested by failed to detect malicious nodes for the top 20 most demanding resources by cluster. Fig. 3 highlights significant differences between the bottom three recall score clusters from other clusters, as their graphs are much denser. This result correlates with our findings from the previous table. It explains the irregularities of our classification model, which support system administrators in alleviating the issues related to common resources caused by malicious nodes within a fog computing system.

V. Discussion and Limitations

As discussed in the result section, our proposed approach for feature extraction can yield competitive results using baseline models and improve with our multi-model classifier. Additionally, our proposed classification method demonstrates the potential to support the decision-making process for system administrators or develop into an automated system manager.

However, our approach utilizes k-means clustering algorithms, which depend heavily on the predetermined number of

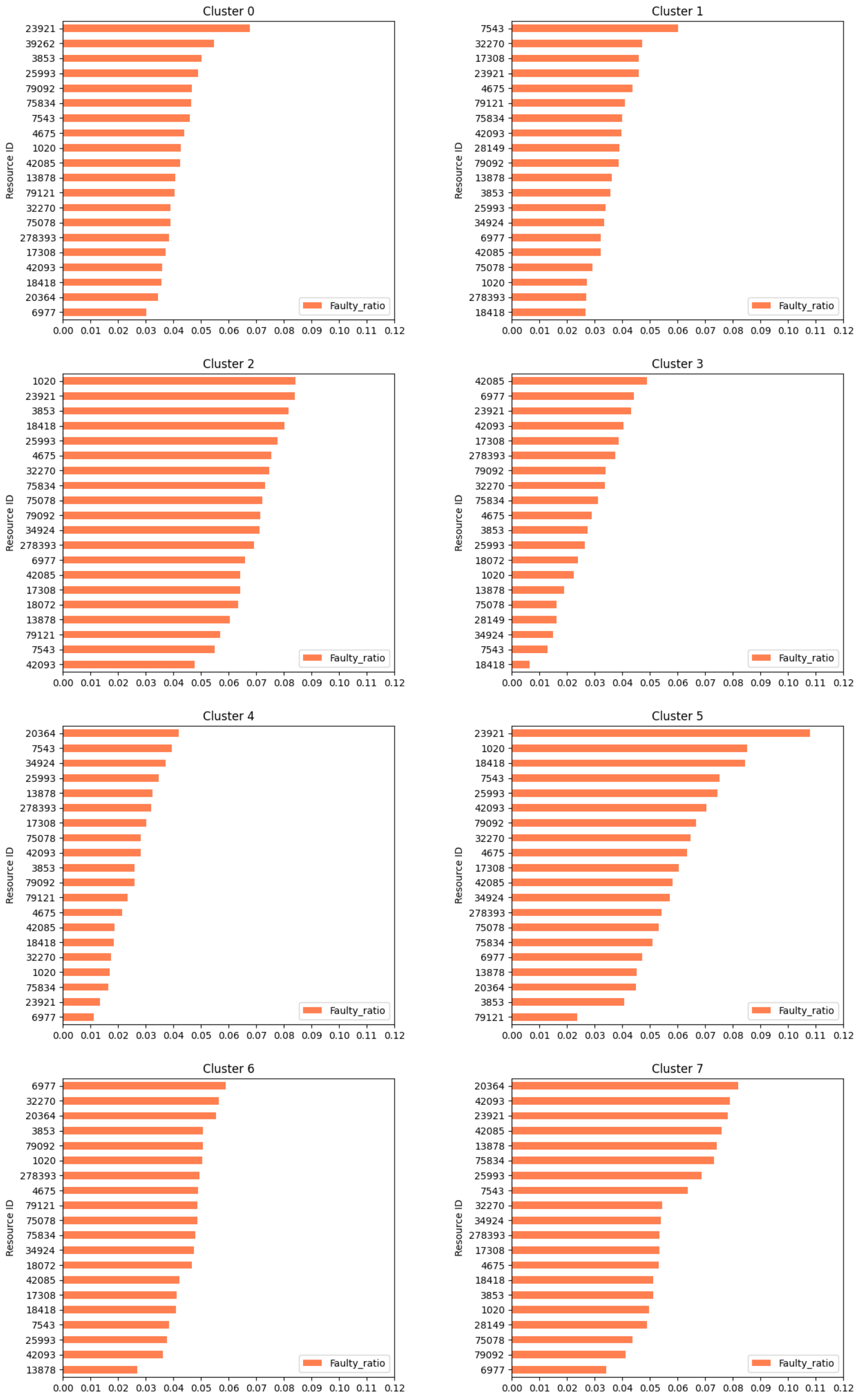

Fig. 3: Ratio of Resource Impacted by Failed Detection of Top 20 requested Resources per Cluster

clusters (k). It is challenging to find the optimal value for k, which directly impacts our multi-model approach and hinders our proposed system from being fully automated. Another challenge we need to discuss is the correlation between each attribute. For example, the attribute of one's position may be strongly correlated to the department the entity is currently operating. Finally, rigorous tests on sparse and diverse datasets are essential to determine the robustness of our multi-model classifier.

Acknowledgment

The research effort of Brajendra Panda has been supported in part by grant H98230-20-1-0419 issued by the National Security Agency as part of the National Centers of Academic Excellence in Cybersecurity's mission to expand cybersecurity research and education for the Nation.

References

[1] J. Jowarth, "80+ Amazing IoT Statistics," 28 11 2022. [Online]. Available: https://explodingtopics.com/blog/iot-stats.

[2] Zahra, Syed, and Chishti, Mohammad Ahsan. (2019). Assesing the Services, security Threaths, Challenges and Solutions in the Internet of Things. Scalable Computing: Practice and Experience. 20. 457-484. 10.12694/scpe.v20i3.1544.

[3] "Cost of a data breach 2022," IBM. [Online]. Available: https://www.ibm.com/reports/data-breach. [Accessed: 24-Apr-2023].

[4] "Dev-0537 criminal actor targeting organizations for data exfiltration and destruction," Microsoft Security Blog, 28-Mar-2022. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2022/03/22/dev-0537-criminal-actor-targeting-organizations-for-data-exfiltration-and-destruction/. [Accessed: 24-Apr-2023].

[5] "Medibank hackers announce 'case closed' and dump huge data file on Dark Web," The Guardian, 01-Dec-2022. [Online]. Available: https://www.theguardian.com/australia-news/2022/dec/01/medibank-hackers-announce-case-closed-and-dump-huge-data-file-on-dark-web. [Accessed: 24-Apr-2023].

[6] Z. Soo, "Alleged Chinese police database hack leaks data of 1 Billion," AP NEWS, 06-Jul-2022. [Online]. Available: https://apnews.com/article/technology-china-hong-kong-shanghai-ce27c6ef6f6d31ce917ee27f80c9589d. [Accessed: 24-Apr-2023].

[7] M. Mukherjee et al., "Security and Privacy in Fog Computing: Challenges," in IEEE Access, vol. 5, pp. 19293-19304, 2017, doi: 10.1109/ACCESS.2017.2749422.

[8] Patwary, Abdullah & Sami, Mohammad. (2019). A Detection Approach for Finding Rogue Fog Node in Fog Computing Environments. American Journal of Engineering Research. 8. 90-98.

[9] L. Kleinman, "Attack from DOS: In Zero we trust," SecurityBrief New Zealand, 20-Sep-2022. [Online]. Available: https://securitybrief.co.nz/story/attack-from-dos-in-zero-we-trust. [Accessed: 25-Apr-2023].

[10] "2018 Cost of insider threats: global." [Online]. Available: https://www.insiderthreatdefense.us/pdf/Ponemon%20Institute%202018%20Report%20-%20The%20True%20Cost%20Of%20Insider%20Threats%20Revealed.pdf. [Accessed: 25-Apr-2023].

[11] "Cost of Insider Threats: Global Report 2020." [Online]. Available: https://www.ibm.com/downloads/cas/LQZ4RONE. [Accessed: 25-Apr-2023].

[12] Stojmenovic, I., Wen, S., Huang, X., and Luan, H. (2016) An overview of Fog computing and its security issues. Concurrency Computat.: Pract. Exper., 28: 2991– 3005. doi: 10.1002/cpe.3485.

[13] A. A.-N. Patwary, R. K. Naha, S. Garg, S. K. Battula, M. A. Patwary, E. Aghasian, M. B. Amin, A. Mahanti, and M. Gong, "Towards secure fog computing: A survey on trust management, privacy, authentication, Threats and Access Control," Electronics, vol. 10, no. 10, p. 1171, 2021.

[14] Zahra, S.R., Chishti, M.A. A generic and lightweight security mechanism for detecting malicious behavior in the uncertain Internet of Things using fuzzy logic- and fog-based approach. Neural Comput and Applic 34, 6927–6952 (2022). https://doi.org/10.1007/s00521-021-06823-9

[15] L. Karimi, M. Aldairi, J. Joshi and M. Abdelhakim, "An Automatic Attribute-Based Access Control Policy Extraction From Access Logs," in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, pp. 2304-2317, 1 July-Aug. 2022, doi: 10.1109/TDSC.2021.3054331.

[16] H. Du et al., "Twitter vs News: Concern Analysis of the 2018 California Wildfire Event," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 207-212, doi: 10.1109/COMPSAC.2019.10208.

[17] "Accuracy paradox," Wikipedia, 18-Mar-2023. [Online]. Available: https://en.wikipedia.org/wiki/Accuracy_paradox. [Accessed: 24-Apr-2023].

[18] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press, USA.

[19] R. L. Thorndike, "Who belongs in the family?," Psychometrika, vol. 18, no. 4, pp. 267–276, 1953.

[20] "Amazon.com - Employee Access Challenge," 2013. [Online]. Available: https://www.kaggle.com/competitions/amazon-employee-access-challenge/data?select=train.csv.

[21] Cortes, C., Vapnik, V. Support-vector networks. Mach Learn 20, 273–297 (1995). https://doi.org/10.1007/BF00994018

[22] Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds) Machine Learning: ECML-98. ECML 1998. Lecture Notes in Computer Science, vol 1398. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0026683

[23] Gaonkar B, Davatzikos C. Analytic estimation of statistical significance maps for support vector machine based multi-variate image analysis and classification. Neuroimage. 2013 Sep;78:270-83. doi: 10.1016/j.neuroimage.2013.03.066. Epub 2013 Apr 10. PMID: 23583748; PMCID: PMC3767485.

[24] Decoste, D., Schölkopf, B. Training Invariant Support Vector Machines. Machine Learning 46, 161–190 (2002). https://doi.org/10.1023/A:1012454411458

[25] Wu, Xindong; Kumar, Vipin; Ross Quinlan, J.; Ghosh, Joydeep; Yang, Qiang; Motoda, Hiroshi; McLachlan, Geoffrey J.; Ng, Angus; Liu, Bing; Yu, Philip S.; Zhou, Zhi-Hua (2008-01-01). "Top 10 algorithms in data mining". Knowledge and Information Systems. 14 (1): 1–37. doi:10.1007/s10115-007-0114-2. hdl:10983/15329. ISSN 0219-3116. S2CID 2367747.

[26] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine.." Ann. Statist. 29 (5) 1189 - 1232, October 2001. https://doi.org/10.1214/aos/1013203451

[27] Hastie, T.; Tibshirani, R.; Friedman, J. H. (2009). "10. Boosting and Additive Trees". The Elements of Statistical Learning (2nd ed.). New York: Springer. pp. 337–384. ISBN 978-0-387-84857-0. Archived from the original on 2009-11-10.

[28] Borisov, Vadim & Leemann, Tobias & Seßler, Kathrin & Haug, Johannes & Pawelczyk, Martin & Kasneci, Gjergji. (2022). Deep Neural Networks and Tabular Data: A Survey. IEEE Transactions on Neural Networks and Learning Systems. PP. 1-21. 10.1109/TNNLS.2022.3229161.

[29] Pedregosa, F. et al., 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825–2830.

[30] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785