An Ensemble Approach for Suspicious Traffic Detection from High Recall Network Alerts

Peilin Wu, Jinlei Li, Yan Meng, Haojin Zhu*
Department of Computer Science and Engineering, *Shanghai Jiao Tong University*, Shanghai, China
{wplisMIT, ricardolee, yan_meng}@sjtu.edu.cn, zhu-hj@cs.sjtu.edu.cn*

***Abstract*—Web services from large-scale systems are prevalent all over the world. However, these systems are naturally vulnerable and incline to be intruded by adversaries for illegal benefits. To detect anomalous events, previous works focus on inspecting raw system logs by identifying the outliers in workflows or relying on machine learning methods. Though those works successfully identify the anomalies, their models use large training set and process whole system logs. To reduce the quantity of logs that need to be processed, high recall suspicious network alert systems can be applied to preprocess system logs. Only the logs that trigger alerts are retrieved for further usage. Due to the universally usage of network traffic alerts among Security Operations Center, anomalies detection problems could be transformed to classify truly suspicious network traffic alerts from false alerts.**

In this work, we propose an ensemble model to distinguish truly suspicious alerts from false alerts. Our model consists of two sub-models with different feature extraction strategies to ensure the diversity and generalization. We use decision tree based boosters and deep neural networks to build ensemble models for classification. Finally, we evaluate our approach on suspicious network alerts dataset provided by 2019 IEEE BigData Cup: Suspicious Network Event Recognition. Under the metric of AUC scores, our model achieves 0.9068 on the whole testing set.

***Index Terms*—Ensemble Model, Imbalanced Dataset, Suspicious Network Event Recognition, 2019 IEEE BigData Cup**

I. Introduction

Web services have become indispensable due to their convenience and wide usage. Companies build web services to provide online videos, data storage, social network service (SNS) and so on. To offer such all-sided services, serving systems become so complicated that they are always full of implicit bugs and secure weaknesses. Since these services store lots of user sensitive information, they are inherently vulnerable to various attacks. Based on the potential consequences, anomalous behaviours of the adversaries can be classified into information distortion, destruction and discovery [1]. According to the Data Breach Statistics [2], the companies from all over the world lost around 6 million records due to the humans faults. Researchers predict that cybercrime will cost the world in excess of $6 trillion annually by 2021, up from $3 trillion in 2015 [3]. Thus, it is urgent to build an effective suspicious network event detection model to prevent potential economical losses.

Previous works have considered to detect anomalies based on raw logs. DeepLog proposes a deep neural network using Long Short-Term Memory (LSTM) to detect anomaly by expressing the raw logs in terms of a natural human language [4]. Xu et al. [5] first transform console logs to numerical features. They then use pattern based filters and Principle Component Analysis (PCA) to detect anomalies. Despite of machine learning ways, Lou et al. [6] propose a workflow mining algorithm. Even though they are successful detection models, none of them aimed to detect anomalies with the combination of so-called network traffic alerts used by Security Operations Center (SOC) and log databases. This combination improves the efficiency of detection. With the help of network traffic alerts, detection models only need to analyze the logs related to alert, which can be retrieved from efficient log database [7], [8]. They highly reduce the number of logs which needs to be inspected. Besides, log databases provide statistical attributes to improve classifiers' performance.

For being intruded by adversaries costs disastrous loss, traffic alert systems sacrifice precision to improve their recall, which tend to generate lots of false alerts. Though they ensures systems' security, they can be annoyed and hard to be dealt with. By introducing network traffic alerts to raw logs preprocessing, anomaly detection problems transform to distinguish truly suspicious network event alerts from false alerts. To improve the performance of classification models, the log database is queried to retrieve crucial attributes of logs which are related to alerts.

Figure 1 shows the architecture of the whole process. System logs record system provides logs which describe current running processes' states in the system. Despite of others, logs corresponding to network traffic are distilled from record system. Network traffic log database records these logs for further usage. Suspicious network event analyzer monitors traffic logs, and when such event is detected, it reports the alert with scores given by its analyzing system. When receiving the alert from analyzer, alert classification system queries log database to gather more attributes of the alert. Based on its present dataset, the log database summarizes the attributes of the alert, which is shown in Section IV, and returns them to the classifier. Given all this information, the alert classification system predicts whether the alert is true or not.

In this paper, we focus on building an alert classification

Haojin Zhu is the corresponding author.

This work is supported by National Key Research and Development Program of China (No.2018YFB1003500).

978-1-7281-0858-2/19/$31.00 ©2019 IEEE

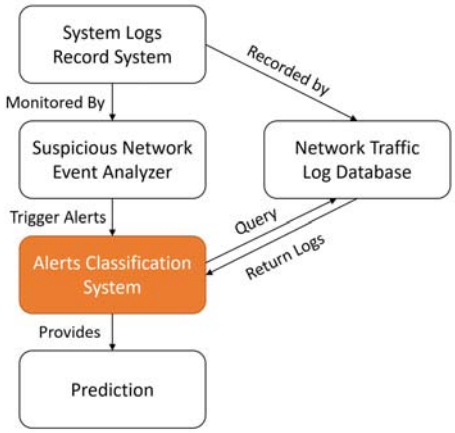

Fig. 1. Architecture of suspicious network events recognition

systems. We propose an model to distinguish truly suspicious network events from false alerts. The model uses network traffic alerts and queries results related to corresponding alerts from the log database with summary-based approximate query engine described in [7], which provides tabular data instead of raw logs. Our model consists of two sub-models, called XCSS and XLRN. They extract features from the dataset following different intuition for improving the robustness of our model. Each sub model builds an ensemble model, including random forest based boosters and deep neural networks, to predict whether an alert is a truly suspicious network event. The results from both models are combined together before submitting final prediction of the authenticity of network traffic alerts.

The challenge in building the classification model is that our dataset is small and imbalanced. Suspicious network event analyzer pursues generating alerts to guarantee the security level. Therefore, negative class becomes majority in our dataset. Nevertheless, our model achieves a high performance with a small dataset. Instead of using all records from log database, our model utilizes a small subset of these logs which are sampled from the original dataset by using [7]. By sacrificing little performance on prediction, our model is effective since we only use 0.2% records in average to train our model and classify alerts.

The structure of this paper is described as follow: Backgrounds of this work are introduced in section II. Section III describes the attributes in our dataset. Section IV explains the methodology we used to build our model and results from the experiments are presented in Section V. We summarize our paper in section VI.

II. Background and Related Work

A. Anomalous Behaviors on Web Services

For the huge diversity and complexity of web services, the previous works have proposed different taxonomies about classifying the behaviour of adversaries [9], [10], [11] and [12]. Due to the various anomalous behaviors on web service, the remainder of this section will introduce the related works of some sub-problems in practice.

One of the crucial attacks aim at electric smart grid. [13] proposes a strategy to expose the weakness in smart girds' framework. It can be used as a guideline before launching cyber attacks on smart grids. Instead of attacking the weakness of the web service systems directly, [14] proposes a Generative Adversarial Networks (GAN), which camouflages its network traffic to deceive defenders. Network traffic generated by this method acts like 'normal' traffic. Besides, for prevalent web services, some works focus on inferring users' private information by collecting overhead packets [15], [16]. Detection strategies which depend on system's logs cannot prevent these kind of attacks.

B. Sampling Strategies on Imbalance Dataset

Suspicious network traffic alerts are generated by high recall systems to guarantee that no suspicious events can evade from the detection. Therefore, most of reported alerts are false labeled and result in an imbalanced dataset. Dataset with imbalanced class labels cause bias of the classifiers. Classifiers are apt to ignore minorities to gain optimum evaluation scores, especially when improper feature extraction strategies or evaluation methods are chosen to train the classifiers. Many previous works have proposed different specific techniques to diminish the bias, such as [17], [18]. Despite of designing specific models, training set could be transformed to new balanced datasets by applying well-designed sampling strategies. [19] and [20] compare average performance of sampling strategies on several public imbalanced datasets with chosen models. Generally, there are three kinds of sampling strategies to balance the dataset on minorities classes:

1) *Oversampling*: Artificially create new minorities points by analyzing the feature space or rise the probabilities of minorities to be sampled [21], [22], [23].
2) *Undersampling*: Assign majorities lower probabilities to be sampled or us certain criteria to eliminate majorities [24], [25], [26], [27].
3) *Combined-sampling*: Combine both methods in oversampling and undersampling to improve the robust of transformed dataset [28], [29].

C. Decision Tree Based Model

Currently, there are many excellent machine learning models based on decision tree. As an emerging and highly flexible machine learning algorithm, random forest [30], [31] have broad application prospects. It integrates multiple decision trees into one model through the idea of ensemble learning. The final prediction of the random forest is obtained by averaging the prediction of each tree. Different from the random forest method, in the boosting method, decision trees are no longer independent but associated. The Gradient Boosting Decision Tree (GBDT) [32] model uses the negative gradient direction of the loss function instead of the residual direction. XGboost [33] adds a regularization term to the loss function, uses the second-order Taylor expansion of the

TABLE I
ALERT AAN'S ATTRIBUTES IN LADT

Attribute Name	Example Values		
alert_ids	AAN	AAN	AAN
alerttype	IDPS Alert	IDPS Alert	IDPS Alert
devicetype	NIPS	NIDS	NIDS
reportingdevice	sZp	WmQ	WmQ
devicevendor	BU	TR	TR
srcip	10.NF.RH.9	10.NF.RH.9	10.NF.RH.9
dstip	UJ.QJ.173.138	UJ.QJ.173.138	UJ.QJ.173.138
srcipcategory	PRIV-10	PRIV-10	PRIV-10
dstipcategory	INTERNET	INTERNET	INTERNET
srcport	43073	43073	51035
dstport	80	80	80
srcportcategory	3	3	4
dstportcategory	2	2	2
direction	3	3	3
alerttime	0	10	1280
severity	5	5	5
count	1	1	1
domain	0	0	0
protocol	6	TCP	TCP
username	0	1	1
signature	1	1	1

objective function, and speeds up the search for the optimal segmentation point through the greedy algorithm. LightGBM [34] utilizes a histogram-based algorithm and Gradient-based One-Side Sampling (GOSS) training method to further improve model efficiency. In 2018, Prokhorenkova1 et al. [35] proposed Catboost, which is very friendly to category features and has performed well on some public datasets.

III. DATASET

Our dataset is provided by 2019 IEEE BigData Cup: Suspicious Network Event Recognition [36]. Two tables, initial training data table (ITDT) and localized alert data table (LADT), are chosen from the dataset to train our alerts classification models. ITDT and LADT are generated from an summary-based approximate query engine described in [7]. It provides crucial information from log event datasets for classification in a standard relational style. To distinguish logs from different alerts, *alert_id* is used as a unique identifier, which matches to several logs depending on the events. In ITDT, each alert has only one record, aggregated from a set of selected logs. Besides, it contains analytical attributes generated from alert systems. LADT provides variable quantities of records for each alert, with more details which are eliminated in ITDT. It is worth to mention that the records in LADT is composed from several logs of the original dataset, which is introduced in [7].

Table I shows an example of data from LADT. Alert AAN has three records in LADT triggered in different times. For these three records, the source IP address and destination IP address are same respectively, but each record aims at different devices.

IV. METHODOLOGY

In this section, we propose an alert classification system in fig. 1. Our system distinguishes truly suspicious alerts from false alerts based on the dataset described in section III. To detect truly suspicious events from false alerts, our classification model consists two sub-models, called XCSS and XLRN respectively. Figure 2 shows the overview of our model. XCSS and XLRN have their own feature extraction methods. They are totally independent and focused on different aspects, though their selected features are partially overlap. XCSS inclines to use directly features and handle category features by known embedding methods [37], while XLRN is apt to extract category features by carefully inspecting the interior connections between different categories. We believe that the usage of different feature extraction strategies could improve our final results on test dataset, for it improves the generalization of our system. Besides, XCSS and XLRN use different methods to build ensemble model, based on the empirical performance on their own features.

To ensure the clarity, following sections describe XCSS and XLRN separately. Section IV-A shows the details of XCSS while section IV-B explains the principles of XLRN. We subsequently state the importance of combining these two models in section IV-C.

A. XCSS

XCSS is designed to utilize embedding method thoroughly by concatenating different attributes together and embedding them to several dimensions. XCSS ignores human's knowledge on the suspicious network events recognition. It focuses on the techniques to embed categories from distinct attributes and categories in variable-length time sequence data instead of aggregating them by statistic methods. The intuition behind XCSS is that human may mislead their models because of their limitation on specific problem. However, using embedding methods without considering the practical meaning of each category can prevent the influence of human's fault. Under the intuition, we implement XCSS in the following ways.

1) Data Cleaning: XCSS cleans ITDT and LADT by simple rules. All categories in attribute *protocol* are converted to lowercase. Besides, *start_minutes* is chopped into 4 different time periods, each has a continuous duration of 15 minutes. For empty entries, XCSS puts 'empty' or zero to them, depending on the characters of corresponding attributes.

2) Feature Extraction: XCSS performs feature extraction mainly based on original attributes in ITDT and LADT. Table II shows the processing methods for each feature. To ensure the clarity of Table II, features which have similar names are combined into a line by using asterisks. When applying Latent Dirichlet Allocation (LDA) [37] to a selected feature, each category in the feature is transformed to 5-dimensional vectors. For *counting* method, XCSS counts each category's quantity in the whole dataset, and assigns the value to correspond categories. For *ratio* method in Table II, these features' counting results are divided by the corresponding values in *correlatedcount* and added as new features.

Considering the implicit interaction of different attributes in dataset, XCSS does counting across different attributes. For

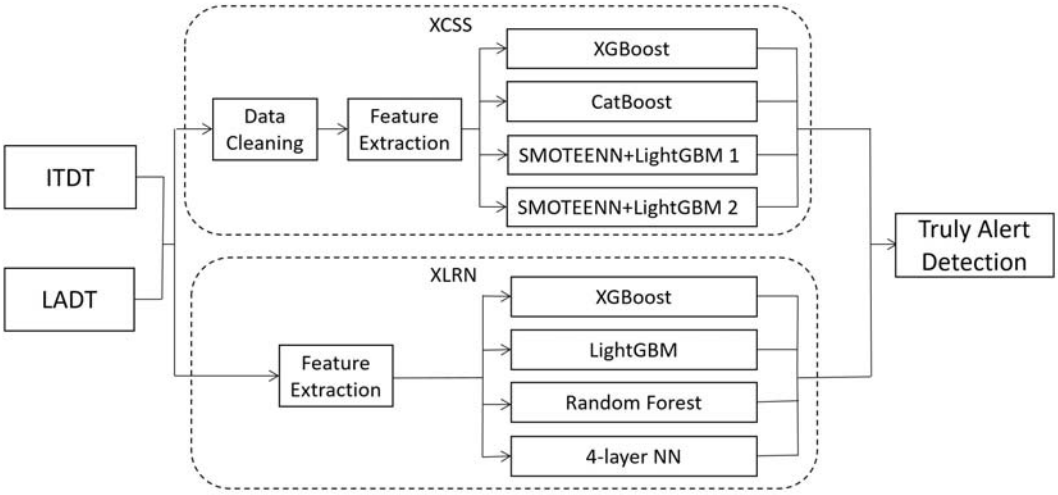

Fig. 2. Flow chart of the truly suspicious alert detection model.

TABLE II
FEATURES PROCESSING METHOD OF XCSS

Feature	Processing Method	Feature	Processing Method
overallseverity	directly input	alerttype	count & ratio
timestamp_dist	directly input & ratio	device_type	count & ratio
correlatedcount	directly input	*_code	count & ratio
isiptrusted	directly input	srcip*	count & ratio
grandparent_category	directly input	dstip*	count & ratio
*_cd	directly input & ratio	direction	count & ratio
*score	directly input	severity	count & ratio
p6, p9, ...	directly input	domain	count & ratio
ip	counting	protocol	count & ratio
weekday	counting & LDA	username	count & ratio
client_code	counting & LDA	parent_category	LDA
categoryname	LDA	start_hour	LDA
ipcategory_name	LDA	ipcategory_scope	LDA
*_dominate	LDA		

every pairs of *client_code*, *ip*, *categoryname*, *weekday*, *dstipcategory_dominate*, *srcipcategory_dominate*, XCSS counts the number of different second chosen attribute categories for each category in the first chosen attribute. After brute-forcing selected attributes, we obtain 30 new features.

Besides of counting, XCSS applies LDA to different sets of feature in three ways:

1) XCSS concatenates pairs of attributes from six assigned attributes (*client_code*, *ip*, *categoryname*, *weekday*, *dstipcategory_dominate*, *srcipcategory_dominate*) in brute-force and treats every line as a two-words sentence. It applies LDA to these sentences and get a 5-dimensional vector for every line. Algorithm 1 shows the embedding method. By this way, XCSS gets another 150 features.
2) Four attributes' categories (*alerttype*, *devicetype*, *reportingdevice_code*, *devicevendor_code*) are aggregated by alert ids respectively. For example, an alert id with eight logs will have four eight-words sentences. Each sentence corresponds to a chosen attribute. By applying LDA, each alert id obtains a 5-dimensional vector for each attribute. In total, XCSS has another 20 features.
3) Instead of applying LDA to each attribute respectively, XCSS selects first nine attributes except *alert_ids* and *protocol* in LADT and builds sentences by the time sequence according to *alerttime*. They are transformed by LDA and 15 more features are obtained.

Eventually, XCSS construct a 374-dimensional vector for every alert id. All features are scaled to standard and normalized.

3) Ensemble Model: XCSS is consisted of three different types of boosting models: LightGBM [34] [38], XGBoost [33] and CatBoost [35]. Although the training set is imbalanced, XGBoost can handle it quite well with high L2 regularization (L2 equals to 100). Besides, Catboost can do well with its default setting. However, when using LightGBM, we found that imbalanced dataset had a non-trivial impact on the metrics scores. To improve its performance against imbalance problem, training set is transformed by SMOTEENN [28], a sampling method to balance training set, before being fed to

Algorithm 1 First Way to Apply LDA

Input: First Attribute of All Alerts a_1
Input: Second Attribute of All Alerts a_2
Output: Embedding Result

$app\_sentence\_matrix \leftarrow list$
for $i = 0$ **to** num_of_alerts **do**
  $attribute\_sentence \leftarrow concatenate(a_1[i], a_2[i])$
  $app\_sentence\_metric$ append $attribute\_sentence$
end for
$token\_matrix \leftarrow CountTokenizer(sentence\_metric)$
$embedded\_matrix \leftarrow LDA(token\_matrix)$
return $embedded\_matrix$

LightGBM.

Empirically, some features are indexed by their categories in table II instead of applying to LDA. By treating them as numerical features, the AUC of XCSS improves slightly. These counter-intuitive features are fed into SMOTEENN and LightGBM. Besides, we add another LGBM model which uses LDA to handle categories features, described in Table II. Equation 1 shows the weights of different models in XCSS:

$$XCSS = lgbm_1 * 0.3 + lgbm_2 * 0.2 + xgb * 0.2 + cat * 0.3 \quad (1)$$

where $lgbm_1$ uses index and $lgbm_2$ uses LDA to embed category attributes. XCSS predicts the probability of an suspicious network event alert being a truly alert.

Even though XCSS extracts adequate features from ITDT and LADT, human's inspection of attributes characteristic is absent from the whole processes. Without inspecting, some useful features, which highly influence the prediction results, may be omitted from the models. To overcome XCSS's flaw, another model called XLRN is built to complement XCSS.

B. XLRN

To overcome the flaws in XCSS, we build XLRN, which handles categorical features by manually inspecting the distribution of different values corresponding to their labels. Different from relying on existing embedding methods, XLRN is based on human's knowledge to extract features from the dataset. The intuition behind XLRN is that features which are constructed by manual interference may have surprised effect on improving the performance. Data cleaning process is omitted in XLRN, for all selected attributes in our dataset are 'clean' enough to be used. XLRN provides another aspect of probing alerts characteristic, which improves the generalization of the final ensemble model. The remainder of this section will introduce XLRN in detail.

1) Feature Extraction: XLRN uses two way to transform ITDT's attributes into features. Table III lists the attributes which are fed to XLRN directly. To ensure the clarity, features which have similar names are combined into a line by asterisks. XLRN obtains 35 features in this way.

By carefully analyzing the distribution of the data, XLRN selects some attributes and infers the correlation between their value and the authenticity of their alarms. By manually designing some policies and specifying assignment rules, XLRN obtains another bunch of features. Table IV lists the selected features and all the possible assigned values for each of them. The intuition is that if a category is more likely to appear with positive samples, it is assigned to 1. -1 is for categories which always appears with negative samples, and if there's no obvious relationship, it is set to 0. As an example, Figure 3 shows the frequency of true alert for each category in the attribute *categoryname*. Note that the average frequency of all samples is 0.0577 (2276/39427). It is obvious that the true alert frequencies of *Malicious Activity* and *Control and Maintain* are far above the average frequency, so they are assigned to 1. Conversely, *Attack*, *Suspicious Reputation* and *Attack Preparation* are assigned to -1 due to their significantly lower frequency. The others are assigned to 0. It is worth to mention that the assignment of feature *n_sum* is based on the sum of numerical values in *n1, n2, ..., n10* in ITDT. Empty values are all filled with zero. XLRN obtains 42 features from ITDT from these two methods.

TABLE III
DIRECTLY INPUT FEATURES OF XLRN

Feature Name	Column Number in ITDT
overallseverity	9
timestamp_dist	10
correlatedcount	15
*_cd	27-42
isiptrusted	43
*score	44-47
thrcnt_*	52-54
p6, ..., p8d	55-62

TABLE IV
MANUALLY ASSIGNED FEATURES OF XLRN

Feature Name	Column Number in ITDT	Possible value
categoryname	3	1, -1, 0
ipcategory_name	5	1, 0
ipcategory_scope	6	1, 0
n_sum	16-25	1, 0
score	26	1, 0
dstipcategory_dominate	48	1, -1, 0
dstportcategory_dominate	50	1, 0

XLRN follows the intuition described above to process dataset in LADT but implements in another way. The relation between LADT and ITDT is many-to-one. It is logical not to simply replicate data points' features in ITDT several times and concatenate ITDT and LADT to a artificial one-to-one relation. For an truly suspicious alert id, we cannot infer that all its logs generated at different times in LADT are truly suspicious. Therefore, all the relevant data in LADT of a individual alert should be treated as a whole and aggregated. For numerical data, some statistical methods are used to explore their characteristics. As shown in Table V, the attribute *alerttime*'s maximum value is divided by corresponding *correlatedcount* in the ITDT. For the attribute *count*, we calculate their sum and average. Other three features' average are used in training process. XLRN obtains six new features in this

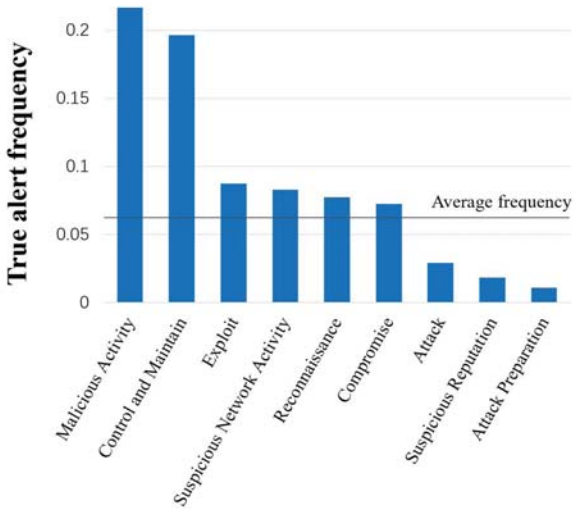

Fig. 3. Frequency of true alert for each category in *Categoryname*.

TABLE V
STATISTICAL FEATURES OF XLRN

Feature Name	Statistical Method
alerttime	division
severity	average
count	sum, average
domain	average
username	average

step.

For variable length categories, we follow the idea described in Table IV by using an additional aggregation step. For each attribute, we carefully analyze the distribution of its corresponding categories and infer the categories which are highly relevant to the positive samples or negative samples. The selected attribute then is transformed to a 4-dimensional vector. It consists of the number of highly relevant categories and irrelevant categories and their frequency. For example, in the attribute *dstipcategory*, *PRIV-192* and *TEST-NET1* are highly relevant to positive samples. *PRIV_CGN* and *PRIV-172* are highly relevant to negative samples. We count their quantities and frequency for each *alert_id*. In this step, other 7 attributes (*alerttype*, *devicetype*, *devicevendor_code*, *srcipcategory*, *srcportcategory*, *dstportcategory*, *direction*) are also selected as features in XLRN. Totally, XLRN contains 80 features.

2) Ensemble Model: XLRN consists of four models: random forest (RF) [30], XGBoost (XGB) [33], LightGBM (LGBM) [34] and a 4-layer neural networks (NN). XLRN doesn't do extra processing to handle imbalance problems. Every models except NN can achieve acceptable classification results. Though NN in XLRN has a poor performance, it improves the robustness of XLRN and it is preserved. Equation 2 shows the weights of different models in XLRN:

$$XLRN = RF * 0.3 + LGBM * 0.3 + XGB * 0.3 + NN * 0.1 \quad (2)$$

where all the weights are chosen empirically. XLRN eventually offers the probability of an suspicious network event alert being a truly alert.

XLRN provides feature extraction by specific analyzing the distribution of different categories, transforming them to numerical features before aggregation in selected attributes. Using specific knowledge makes the results become explicable, comparing to automatic feature extraction methods. Despite of providing manual inspection in dataset, the feature space of XLRN is limited by our understanding of suspicious network events, which means that truly alerts and false alerts may not be successfully distinguished or they are divided in an improper way.

C. Model Ensembling and Truly Alert Detection

XCSS focuses on applying existing methods to embed categories and concatenate attributes to find the interior relationship amongst them, while XLRN is apt to manual inspect the distribution of different categories in each selected attributes and make the results explicable. To benefit from the advantages of both XCSS and XLRN while evading their flaws, prediction results from these two sub-models are combined before giving the final decision. The difference of feature extraction strategies ensures the generalization and robustness of the final model. We combine the prediction results with equal weights, as shown in Eq.3:

$$TRUE\_PROBA = XCSS * x + XLRN * y \quad (3)$$

where the parameters x and y are both empirically set to 0.5 in this study. After combination, all alerts with $TRUE\_PROBA$ higher than 0.5 will be classified as truly suspicious alerts.

V. EVALUATION RESULTS

A. Set up

We use two NVIDIA 2080 GPUs to train our model. There are 39,427 different alerts in our training set, with 2,276 truly suspicious alerts. For testing set, we have 20,000 different alerts. 2,000 alerts in testing set are randomly selected to evaluate the performance of XCSS and XLRN.

B. Training

We use Area under the ROC Curve (AUC) scores to evaluate the performance of our model. AUC provides an aggregate measure of performance across all possible classification threshold [39]. To optimize the performance of the model, we tune the hyper-parameters of each classifier respectively by using 5-folds cross validation except CatBoost of XCSS and NN of XLRN. CatBoost originally has well-tuned hyper-parameters which produce acceptable AUC and NN's hyper-parameters are manually set based on experience. Besides, the performance of CatBoost is highly influenced by initial

TABLE VI
5-FOLDS CROSS VALIDATION AUC SCORES

Sub-Model	Method	AUC Train	AUC Val
XCSS	lgbm_1	0.996	0.9477
	lgbm_2	0.999	0.9446
	xgb	0.999	0.9342
XLRN	rf	0.999	0.9253
	lgbm	0.996	0.9156
	xgb	0.999	0.9190

TABLE VII
AUC SCORES IN TESTING SET

Model Name	AUC Scores (Partial)	AUC Scores (Whole)
XCSS	0.9325	-
XLRN	0.9303	-
XCSS+XLRN	0.9420	0.9068

random state. We train several classifiers with random initial seeds and choose the one with best performance.

Table VI shows the 5-folds cross validation results on training set. It is obvious that the performance of single models in XCSS is generally better than models in XLRN, but we found that the models in XLRN can also achieve acceptable performance after ensemble.

C. Testing

After fine-tuning the hyper-parameters of XCSS and XLRN, we evaluate them respectively. Due to the regulation of BigData Cup Challenge, we cannot access the AUC scores from the whole testing set. Our prediction results can only evaluate on the 2,000 alerts testing set before the end of the competition. Table VII shows evalution of XCSS and XLRN on testing set.

XCSS achieves 0.9325 AUC score while XLRN achieves 0.9303 AUC score. By combining the prediction results from XCSS and XLRN, our model achieves 0.9420 AUC score, which approves the combination of two parallel models which are built in distinct methods. It is worth to mention that XCSS tried to include a simple bagging-NN [40] trained by under-sampling false alerts to keep training set balanced. Though XCSS achieves higher AUC score (0.9339), the combination model's AUC score drops to 0.9396. We believe that the bagging-NN will cause XCSS overfit.

Finally, we test our ensemble model as described in Section IV-C on the whole testing containing 20,000 alerts, and it achieves the AUC score of 0.9068 eventually.

VI. CONCLUSION

In this work, we propose an ensemble model to classify truly suspicious network event alerts from false alerts with certain logs corresponding to the alerts. Our model is divided into two independent sub models, called XCSS and XLRN. Different feature extraction strategies are used to them respectively. XCSS focuses on applying LDA to find the inner connection among different attributes while XLRN inclines to analyze the relationship between different categories and alerts authenticity in several selected attributes. We use random forest based boosters and deep neural networks to build ensemble models for classification. We evaluate our model in the suspicious alerts dataset provided by 2019 IEEE BigData Cup. The dataset concludes about 40,000 training alerts and 20,000 testing alerts with their summary-based approximate event logs. XCSS achieves 0.9325 AUC scores while XLRN achieves 0.9303 AUC scores on 2,000 random selected alerts from testing set. By combining the results from two sub-models, our model achieves 0.9068 on the whole testing set. Future works can focus on reducing the classifiers in the model while keeping the performance to further improve the efficiency of detecting suspicious network traffic events.

REFERENCES

[1] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta, and Q. Wu, "Avoidit: A cyber attack taxonomy," University of Memphis, Tech. Rep. CS-09-003, 2009.

[2] (2019, Oct.) Data breach statistics. [Online]. Available: https://breachlevelindex.com/

[3] S. Morgan, "2019 official annual cybercrime report," Cybersecurity Ventures, Tech. Rep., 2019.

[4] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *CCS'17*, Dallas, TX, USA, Nov. 2017.

[5] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the 26th International Conference on Machine Learning*, Haifa, Israel, 2009.

[6] J.-G. Lou, Q. Fu, S. Yang, J. Li, and B. Wu, "Mining program workflow from interleaved traces," in *KDD'10*, Washington, DC, USA, Jul. 2010.

[7] D. Ślezak, A. Chadzyńska-Krasowska, J. Holland, P. Synak, R. Glick, and M. Perkowski, "Scalable cyber-security analytics with a new summary-based approximate query engine," in *2017 IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, Dec. 2017.

[8] M. Waugh, "System and method for adding network traffic data to a database of network traffic data," U.S. Patent US20 020 062 223A1, Nov. 08, 2001.

[9] B. M. Valúšek, "Classification of network attacks and detection methods," Master's thesis, Masaryk University, 2016.

[10] A. Magar, "State-of-the-art in cyber threat models and methodologies," Sphyrna Security, Kanata, Ontario, Tech. Rep. W7714-08FE01/001/ST, Mar. 2016.

[11] J. Ferdinand and R. Benham, "The cyber security ecosystem: Defining a taxonomy of existing, emerging and future cyber threats," SWIFT Institution, Tech. Rep. 2016-002, Oct. 2017.

[12] N. Abrek, "Attack taxonomies and ontologies," Department of Informatics, Technical University of Munich, Tech. Rep. 10.2313/NET-2015-03-1_01, Mar. 2015.

[13] A. Hahn and M. Govindarasu, "Cyber attack exposure evaluation framework for the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 835–843, Dec 2011.

[14] J. Li, L. Zhou, H. Li, L. Yan, and H. Zhu, "Dynamic traffic feature camouflaging via generative adversarial networks," in *2019 IEEE Conference on Communications and Network Security (CNS)*, June 2019, pp. 268–276.

[15] H. Li, H. Zhu, and D. Ma, "Demographic information inference through meta-data analysis of wi-fi traffic," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1033–1047, May 2018.

[16] H. Li, Z. Xu, H. Zhu, D. Ma, S. Li, and K. Xing, "Demographics inference through wi-fi network traffic analysis," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[17] A. Sun, E.-P. Lim, and Y. Liu, "On strategies for imbalanced text classification using svm: A comparative study," *Decision Support Systems*, vol. 48, pp. 191–201, Jul. 2009.

[18] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser, "Svms modeling for highly imbalanced classification," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, vol. 39, no. 1, pp. 281–288, Jul. 2009.
[19] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, pp. 113–141, Jul. 2013.
[20] Y. Sun, A. K. C. Wong, and M. S. Kamek, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 4, pp. 687–719, 2009.
[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, pp. 321–357, Feb. 2002.
[22] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 International Joint Conference on Neural Networks (IJCNN'08)*, 2008.
[23] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *2005 International Conference on Intelligent Computing (ICIC'05)*, vol. 3644, 2005, pp. 878–887.
[24] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, May 1968.
[25] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 6, pp. 448–452, Jun. 1976.
[26] J. Zhang and I. Mani, "KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction," in *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.
[27] I. Tomek, "Two modifications of cnn," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976.
[28] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 20–29, Jun. 2004. [Online]. Available: http://doi.acm.org/10.1145/1007730.1007735
[29] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing training data for automated annotation of keywords: a case study," in *the Proc. Of Workshop on Bioinformatics*, Jan. 2003, pp. 10–18.
[30] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.
[31] (2004, Jun.) Random forests by leo breiman and adele cutler. [Online]. Available: https://www.stat.berkeley.edu/ breiman/RandomForests/
[32] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.
[33] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *KDD'16*, 2016.
[34] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. yan Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.
[35] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.
[36] A. Janusz, D. Kałuża, Chądzyńska-Krasowska, B. Konarski, J. Holland, and D. Ślęzak, "IEEE BigData 2019 Cup: Suspicious Network Event Recognition," in *2019 IEEE International Conference on Big Data, BigData 2019, Los Angeles, CA, USA, December 9-12, 2019*, 2019.
[37] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.
[38] K. V. Rashmi and R. Gilad-Bachrach, "Dart: Dropouts meet multiple additive regression trees," in *18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, San Diego, CA, USA, 2015.
[39] (2019, Oct.) Classification: Roc curve and auc. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc
[40] L. Breiman, "Bagging predictors," University of California at Berkeley, Tech. Rep., Sep. 1994.