Computer Fraud and Security
ISSN (online): 1873-7056

Intelligent Anomaly Detection in Enterprise Networks Using Machine Learning and Statistical Methods

Shubh Prabhat

Project Leader, Wipro Limited, Cary, North Carolina, USA

Corresponding author: Shubh Prabhat (sprabhat.research@gmail.com)

Research Area: AI and Cybersecurity

ABSTRACT Enterprise networks see varied network traffic. Next-generation signature detection does not detect many network attacks. Anomaly detection approaches can help by building models of normal network traffic to detect abnormal traffic. Statistical anomaly detection approaches have high false-negative rates. Machine learning approaches provide limited insight into why a given network flow is suspicious. We present a hybrid statistical and machine learning approach to network anomaly detection. We build four models of normal network traffic using only normal network traffic data, a median and median absolute deviation z-score, a Mahalanobis distance with a minimum covariance determinant estimator, a squared principal component residual, and an isolation forest anomaly score. The four anomaly scores and 30 traffic flow features chosen according to mutual information are used as inputs. A random forest classifier is used. Out-of-fold cross-validation is employed to ensure that the test dataset is not used to test models built using the training dataset. Experimental results on the official training and test datasets of the UNSW-NB15 dataset are compared with the results of other machine learning and statistical methods. On the test set, using 34 features instead of 68 results in an accuracy of 86.87%, a detection rate of 98.89%, an F1-score of 0.892, and an area under the ROC curve of 0.979. This approach is comparable in performance to other complete random forest models and outperforms statistical models both individually and combined (AUC = 0.778-0.826). Evaluation suggests that the statistical model predictions are strong predictors in the combined model but do not add predictive power on top of other flow features. Evaluation of different prediction thresholds suggests that setting a higher threshold of 0.8 for predictions would lower the false-positive rate from 27.86% to 7.04% at a true-positive rate of 91.70%. Other findings about differences in cross-validation and cross-partition performance may be useful for business applications.

INDEX TERMS Anomaly detection, enterprise network security, intrusion detection, isolation forest, Mahalanobis distance, machine learning, random forest, statistical profiling, UNSW-NB15.

I. INTRODUCTION

Enterprise networks have cloud services and other networked systems in addition to remote users and other networked systems. These connections expose enterprise networks to cyber threats. Firewalls and signature-based intrusion detection and prevention systems offer some protection against cyber threats but cannot identify new exploits, slow reconnaissance or insider misuse [1]. Implementing a NIDS with behavioral modeling can help protect against unknown cyber threats because behavioral modeling identifies unusual activity [2], [3].

Anomaly detection in computer networks can be done in two ways. Statistical methods model normal computer network activities using mathematical relationships to identify anomalies [4], [5]. Benefits of statistical methods include ease of understanding the model and not requiring training data on attacks. Drawbacks of statistical methods include dependence on the similarity measure and other thresholds used to identify anomalies and an inability to identify attacks that closely resemble normal network activity [5]. Machine learning methods for anomaly detection include supervised classification and deep learning techniques trained on labeled data to identify attacks. These methods tend to be more accurate than statistical methods [6], [7], [8]. Drawbacks of machine learning methods include a need for training data on known attacks and other challenges related to understanding results.

Three drawbacks of network flow detectors in an enterprise environment include the following: excessive and irrelevant features in network flow records that can slow training and cause overfitting [9], [10]. High false alarm rates that can lower security analysts' confidence and create more work for them in the security operations center [11]. Likely overestimation of future performance on network flow records collected at different times based on research studies that use cross-

validation on a single data partition [12]. To assess a network flow detector for use in an enterprise environment, you should use an evaluation data set that the detector has not previously encountered.

This paper describes a hybrid approach that uses a combination of statistical models and machine learning for cybersecurity threat detection at the enterprise level. In the hybrid approach, outputs from statistical and other unsupervised models of enterprise traffic are provided as inputs to a supervised random forest model. These outputs, or anomaly scores, are inputs to the supervised model. The hybrid approach is similar to the two-step approach discussed by Khan et al. [7], in which outputs from one model are provided as inputs to a separate model. Differences include the use of an unlabeled statistical profiling model rather than a deep learning model in the first step of the hybrid approach.

The main contributions of the study include:

1. Hybrid model. A hybrid model includes four anomaly detection techniques applied to normal network flows (robust z-score, robust Mahalanobis distance, principal component-residual, and isolation forest) and flow features selected according to mutual information. A random forest model is used for classification.

2. A method for scoring test cases not included in the training set is described to prevent the statistical models from being used to score the training set records used to build the models. Preventing such use avoids overly favorable evaluations of the combined features.

3. We perform a comprehensive evaluation of the official UNSW-NB15 data sets. We compare seven machine learning techniques with four other statistical methods. Other evaluations we include are ablation testing and tests with varying numbers of features using fivefold cross-validation.

4. Deployment information is available from feature importance and decision threshold analyzes that help link predictions from your model to false alarm rate limits for a SOC.

The rest of this paper is organized as follows. Section II reviews related literature. Section III offers background on statistical methods. Section IV describes the proposed method. Section V presents the results of experiments. Section VI interprets experimental results and limitations. Section VII offers final comments.

II. RELATED WORK

A. STATISTICAL ANOMALY DETECTION

Early network intrusion detection methods employed traffic and packet header analysis. Kim and Reddy [4] evaluated correlation of destination IP addresses in egress traffic at one router. This method detected denial-of-service and scanning attacks efficiently in near-real time. Thottan et al. [2] surveyed stream-based and other unsupervised learning methods for network applications. Most deployed network intrusion detection methods rely on basic statistical data and a model of normal network behavior, they found. Weller-Fahy et al. [5] surveyed distance and similarity measures applied to network intrusion detection. Distance and similarity measures are seldom described in the literature but can affect network intrusion detection outcomes substantially, they found. This study applies a robust Mahalanobis distance measure to address this problem.

Statistical features have been integrated with machine learning methods. Rastegari et al. [13] proposed a genetic algorithm-based method for extracting interval-based rules using continuous statistical features. Moustafa et al. [8] proposed application-specific statistical features and integrated them with AdaBoost ensembles of decision tree, naive Bayes, and neural network classifiers. The UNSW-NB15 and botnet datasets were used to evaluate the proposed approach. Vasudevan and Selvakumar [14] developed a hierarchical model that includes a local outlier factor component and a one-class classifier. The cited publications indicate that statistical network features can be useful additions to other classification models.

B. MACHINE LEARNING BASED DETECTION

Traditional machine learning approaches have been applied to intrusion detection. Shon and Moon [15] proposed a hybrid approach combining a self-organizing map and genetic algorithm for feature selection with an enhanced SVM classifier to detect new attack types. Malik et al. [10] proposed a binary particle swarm optimization approach for feature selection when using a random forest classifier. Khammassi and Krichen [9] incorporated logistic regression into a genetic algorithm approach for feature selection with the KDD99 and UNSW-NB15 datasets, preserving performance. Meng and Kwok [11] suggest that different classifiers may work better in different situations and propose a voting approach to select an ensemble

of classifiers and reduce false alarms. Dromard et al. [16] propose an online, unsupervised intrusion detection approach that updates features over time for a time window. Grid clustering techniques can be used to apply this approach at high bandwidths.

C. DEEP LEARNING BASED DETECTION

Deep learning has been studied for NIDS since 2013. Fiore et al. [17] studied a semi-supervised anomaly detection method using a discriminative restricted Boltzmann machine and reported lower accuracy when applied to test data from other networks. Shone et al. [6] proposed and implemented a non-symmetric deep autoencoder for unsupervised feature learning and a random forest classifier. Al-Qatf et al. [18] proposed and implemented a self-taught learning deep autoencoder and an SVM classifier for autoencoder features. Choi et al. [19] proposed and implemented a fully unsupervised NIDS to avoid the costs and labor associated with manual data labeling using an autoencoder. Khan et al. [7] proposed and implemented a two-stage method in which a stacked autoencoder was used to classify traffic as normal or abnormal, and autoencoder output probabilities were used as additional features for a multi-class classifier. Kwon et al. [3] and Chaabouni et al. [20] reviewed the use of deep learning and other machine learning methods for NIDS in general and IoT networks.

D. RESEARCH GAP

The literature review identified three limitations of prior work. First, statistical and machine learning methods have been developed and tested separately, and there are few studies using multiple label-free statistical scores as features in a supervised classifier. Second, combined and stacked feature approaches have generally applied the individual statistical methods to the same data as the final classifier, resulting in a potential bias toward high estimated classifier performance. Third, few prior studies tested classifier performance on holdout data sets and reported only cross-validated performance estimates. Testing and applying models to new samples is an important step in implementing successful classifiers [12], [1]. The methods presented in this paper overcome all three limitations.

III. THEORETICAL BACKGROUND

This section reviews the statistical techniques for modeling normal network traffic. Let x∈R^m be a vector representing a transformed network flow and N={x_1,…,x_n} be a set of training data consisting of normal network traffic. Table 1 provides important notation.

TABLE 1. Main notation used in this paper.

Symbol	Description
$\mathbf{x}$, m	Pre-processed flow record and its dimension ($m = 68$)
$\mathcal{N}$	Set of normal training records used for profiling
$\tilde{x}_j$, MAD_j	Median and median absolute deviation of feature j
$\mathbf{W}_q$	Matrix of the first q principal directions
$\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\Sigma}}$	Robust (MCD) location and scatter
s_z, s_M, s_Q, s_{IF}	Robust z, Mahalanobis, PCA residual and isolation forest scores
K	Number of features kept by mutual information
$\mathbf{z}$	Fused input vector of the random forest
T, τ	Number of trees and decision threshold

A. ROBUST UNIVARIATE PROFILE

The mean and standard deviation are not appropriate measures of central tendency and variability for traffic counter data. The median and median absolute deviation are other measures of central tendency and variability. Robust z-scores are calculated for each numeric variable j.

$$r_j(\mathbf{x}) = \frac{|x_j - \tilde{x}_j|}{1.4826 \cdot \mathrm{MAD}_j}, \qquad \mathrm{MAD}_j = \mathrm{median}_i |x_{ij} - \tilde{x}_j| \tag{1}$$

The constant 1.4826 is chosen so that MAD and standard deviation are the same for a normal distribution. We use a root-mean-square approach to combine individual feature scores and limit individual feature scores to 50 to reduce the influence of a single extreme counter:

$$s_z(\mathbf{x}) = \log\left(1 + \sqrt{1/m_c \sum_{j=1}^{m_c} \min\left(r_j, 50\right)^2}\right) \tag{2}$$

where m_c is the number of continuous features.

B. PRINCIPAL COMPONENT RESIDUAL

Principal component analysis (PCA) is performed on normal records. The first q principal components that explain 95% of the variance form the columns of the matrix $\mathbf{W}_q$. To reconstruct a specific record:

$$\mathbf{t} = \mathbf{W}_q^{\top}(\mathbf{x} - \bar{\mathbf{x}}), \qquad \hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{W}_q \mathbf{t} \tag{3}$$

The squared prediction error (SPE), or Q-statistic, measures the distance of individual observations from the normal subspace.

$$s_Q(\mathbf{x}) = \log\left(1 + 10^3 \parallel \mathbf{x} - \hat{\mathbf{x}} \parallel_2^2\right) \tag{4}$$

C. ROBUST MAHALANOBIS DISTANCE

The Mahalanobis distance accounts for correlations between variables. Outliers can strongly affect estimates of the variance matrix. A method called the minimum covariance determinant (MCD) is more resistant to outliers because it involves identifying a sub-sample of observations with the lowest covariance matrix determinant. With the MCD location $\hat{\boldsymbol{\mu}}$ and scatter $\hat{\boldsymbol{\Sigma}}$ computed on the first ten principal scores $\mathbf{t}_{1:10}$, the score is

$$s_M(\mathbf{x}) = \log\left(1 + \sqrt{(\mathbf{t}_{1:10} - \hat{\boldsymbol{\mu}})^{\top} \hat{\boldsymbol{\Sigma}}^{-1} (\mathbf{t}_{1:10} - \hat{\boldsymbol{\mu}})}\right) \tag{5}$$

D. ISOLATION FOREST

The isolation forest is an unsupervised ensemble method that builds random paths to isolate data points. Anomalies are isolated with fewer splits and have a shorter average path length, so their average path length $E[h(\mathbf{x})]$ is short. With sub-sample size ψ, the score is

$$s_{IF}(\mathbf{x}) = 2^{-E[h(\mathbf{x})]/c(\psi)}, \qquad c(\psi) = 2H(\psi - 1) - \frac{2(\psi - 1)}{\psi} \tag{6}$$

where $H(\cdot)$ is the harmonic number. Scores close to 1 indicate anomalies.

E. MUTUAL INFORMATION

Mutual information measures how much knowing a feature reduces uncertainty about the class label y:

$$I(x_j; y) = \sum_{x_j} \sum_{y} p\,(x_j, y) \log \frac{p(x_j, y)}{p(x_j)\, p(y)} \tag{7}$$

It captures non-linear dependence and is estimated here with a nearest-neighbour estimator for continuous features.

IV. METHODOLOGY

A. OVERVIEW OF THE PROPOSED FRAMEWORK

The proposed method includes five steps, shown in Fig. 1. Step 1 preprocesses and normalizes raw flow data. Step 2 generates four normal flow profiles and calculates four anomaly scores for each raw flow record. Step 3 selects the top K features from the raw flow data. Step 4 combines the selected features with the anomaly scores. Step 5 trains a random forest model to detect anomalies in the flow data. Alerts are generated for all detected anomalous flows.

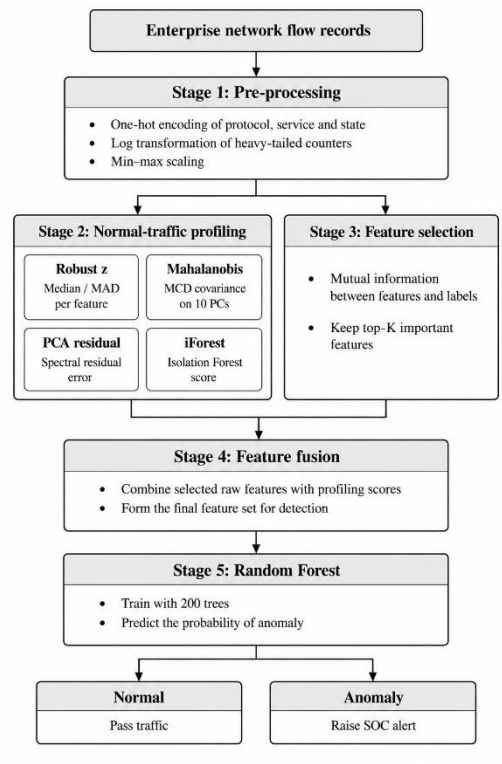


FIGURE 1. Architecture of the proposed hybrid statistical and machine learning anomaly detection framework.

B. DATASET DESCRIPTION

The UNSW-NB15 dataset was developed by the Cyber Range Lab of the Australian Centre for Cyber Security. It contains both real sensor data reflecting current normal and attack activity and simulated attack traffic captured in a test environment. This dataset is more representative of an enterprise network than other datasets [8], [7]. This dataset is used without modification. It contains 175,341 training records and 82,332 test records. There are 42 features, an attack category and a binary response variable for each record. Tables 2 and 3 provide information about class and feature distributions.

TABLE 2. Class distribution of the UNSW-NB15 partitions used in this study.

Class	Training records	Training (%)	Testing records	Testing (%)
Normal	56,000	31.94	37,000	44.94
Generic	40,000	22.81	18,871	22.92
Exploits	33,393	19.04	11,132	13.52
Fuzzers	18,184	10.37	6,062	7.36
DoS	12,264	6.99	4,089	4.97
Reconnaissance	10,491	5.98	3,496	4.25
Analysis	2,000	1.14	677	0.82

Class	Training records	Training (%)	Testing records	Testing (%)
Backdoor	1,746	1.00	583	0.71
Shellcode	1,133	0.65	378	0.46
Worms	130	0.07	44	0.05
Total	**175,341**	**100**	**82,332**	**100**

TABLE 3. Feature groups of UNSW-NB15 and the pre-processing applied.

Group	Example features	Pre-processing
Basic flow	dur, sbytes, dbytes, sttl, dttl, sload, dload, spkts, dpkts	log(1+x), min–max
Content	swin, dwin, smean, dmean, trans_depth, response_body_len	log(1+x), min–max
Time	sjit, djit, sinpkt, dinpkt, tcprtt, synack, ackdat	log(1+x), min–max
Connection	ct_srv_src, ct_dst_ltm, ct_src_dport_ltm, ct_dst_src_ltm	log(1+x), min–max
Nominal	proto (133 values), service (13), state (9)	One-hot, rare levels merged
Flag	is_sm_ips_ports	Kept as 0/1

C. DATA PRE-PROCESSING

The record identifier, attack type and label fields are removed from the dataset. Nominal predictor fields are recoded as binary variables. To simplify the dataset, we limit protocols to the five most common and services to the 12 most common; less common protocols and services are grouped into an "other" category. There are 29 binary predictors. Continuous predictors have heavy-tailed distributions and are transformed and standardized.

$$x'_{ij} = \frac{\log(1 + x_{ij}) - \min_i \log(1 + x_{ij})}{\max_i \log(1 + x_{ij}) - \min_i \log(1 + x_{ij})} \qquad (8)$$

The minimum and maximum values are estimated from the training set. Test set values are clipped between −0.5 and 1.5 to limit the effect of unseen extremes. Adding the extra binary variable results in m=68 predictors for each observation.

D. NORMAL-TRAFFIC PROFILING AND OUT-OF-FOLD SCORING

The four profiles in Section III are applied to normal data only. PCA keeps q=12 principal components, which explain 95% of the variance in the data. The MCD estimator is fit to a random sample of 20,000 normal data points with a support fraction of 0.9. The isolation forest model has 200 trees and a sub-sample size of ψ=256.

A bias is introduced by training the profiles and random forest on the same data. Normal observations used to train the profiles and random forest will have low anomaly scores and the classifier will learn a misleading rule. Out-of-fold scoring can reduce this bias. The normal data are split into five folds. For each fold k, the profiles $\Phi^{(-k)}$ are fitted on the normal records of the other four folds and then score fold k:

$$\mathbf{s}(\mathbf{x}_i) = \Phi^{(-k(i))}(\mathbf{x}_i) = [s_z, s_M, s_Q, s_{IF}] \qquad (9)$$

where $k(i)$ is the fold that contains record i. For test records, the profiles are refitted on all normal training records.

E. FEATURE SELECTION AND FUSION

Mutual information (7) (computed for the label and all 68 features on a random sample of 40,000 training examples) is used to select the top K=30 features. The classifier takes as input a combination of the selected input features and

$$\mathbf{z} = [\mathbf{x}_{\text{sel}} \,\|\, s_z, s_M, s_Q, s_{IF}] \in \mathbb{R}^{34} \qquad (10)$$

F. RANDOM FOREST CLASSIFICATION

A random forest of T=200 trees is fitted to z. Each tree is built at each node by bootstrapping the data and selecting a random set of √34 features. The split that maximizes the reduction in Gini impurity is chosen at each node:

$$G(S) = 1 - \sum_{c \in \{0,1\}} p_c^2 \qquad (11)$$

where p_c is the share of class c in node S. The forest output and the final decision are

$$P(y = 1 \mid \mathbf{z}) = \frac{1}{T} \sum_{t=1}^{T} p_t (y = 1 \mid \mathbf{z}) \qquad (12)$$

$$\hat{y} = \begin{cases} 1, & P(y = 1 \mid \mathbf{z}) \geq \tau \\ 0, & \text{otherwise} \end{cases} \qquad (13)$$

Here p_t is the class probability given by tree t. The default threshold is $\tau = 0.5$. The training and detection steps are given in Algorithms 1 and 2.

Algorithm 1. Training of the proposed framework.

Input: training records X, labels y, $K = 30$, $T = 200$

Output: profiles Φ, selected index set F, forest RF

Pre-process X with one-hot encoding and Eq. (8)
Split X into five folds
for $k = 1$ to 5 **do**
Fit $\Phi^{(-k)}$ on normal records outside fold k
Compute $\mathbf{s}_i = \Phi^{(-k)}(\mathbf{x}_i)$ for records in fold k
end for
Fit Φ on all normal training records
Rank features by $I(x_j; y)$; keep top K as F
Build $\mathbf{z}_i = [\mathbf{x}_{i,F} \| \mathbf{s}_i]$ for all i
Train RF with T trees on $\{(\mathbf{z}_i, y_i)\}$
return Φ, F, RF

Algorithm 2. Detection of anomalous flows.

Input: new flow x, Φ, F, RF, threshold τ

Output: label $\hat{y}$ (0 = normal, 1 = anomaly)

Pre-process **x** with training statistics
$\mathbf{s} \leftarrow \Phi(\mathbf{x})$ using Eqs. (2), (4), (5), (6)
$\mathbf{z} \leftarrow [\mathbf{x}_F \| \mathbf{s}]$
$p \leftarrow P(y = 1 \mid \mathbf{z})$ by Eq. (12)
$\hat{y} \leftarrow$ Eq. (13); **if** $\hat{y} = 1$ **then** raise alert
return $\hat{y}$

G. EVALUATION PROCEDURE AND METRICS

All models are trained on the full training set and tested once on the test set. The test set is not used in any other way. Fivefold stratified cross-validation is applied to the training set to measure stability. Other metrics derived from TP, TN, FP, and FN values are reported, assuming the attack class is the positive class:

$$\mathbf{ACC} = \frac{TP + TN}{TP + TN + FP + FN} \qquad (14)$$

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN} \tag{15}$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R}, \qquad \text{FAR} = \frac{FP}{FP + TN} \tag{16}$$

Recall R is sometimes called the detection rate elsewhere. Other metrics reported include AUC, an ROC curve metric typical of continuous model outputs. Reported training and average record detection times are also provided.

V. EXPERIMENTS AND RESULTS

A. IMPLEMENTATION AND BASELINES

The framework was built with Python 3 and libraries scikit-learn, NumPy, and pandas. Data visualizations were created with Matplotlib. Experiments were run on a Linux system with two 2.1 GHz Intel Xeon CPU cores and 8 GB of memory. All experiments were repeated using random seed 42 for reproducibility. Seven supervised learning methods were trained on all 68 pre-processed predictors to establish baseline results. Four other statistical and unsupervised methods were also run as single models; these methods identify anomalies in test data by comparing out-of-fold scores for training data to the 95th percentile. Table 4 lists the tested methods.

TABLE 4. Model settings used in the experiments.

Model	Main settings
Logistic regression (LR)	L2 penalty, C = 1, L-BFGS solver
Gaussian naive Bayes (GNB)	Default variance smoothing
Decision tree (DT)	Gini, minimum leaf size 2
Linear SVM (L-SVM)	C = 1, squared hinge loss
k-nearest neighbours (KNN)	k = 5, Euclidean distance
Multilayer perceptron (MLP)	64–32 ReLU units, Adam, early stopping
Random forest (RF)	200 trees, √m features per split, leaf size 2
Standalone detectors	Robust z, Mahalanobis, PCA-SPE, iForest; 95th-percentile threshold
Proposed	K = 30, four scores, RF with 200 trees, τ = 0.5

B. COMPARATIVE RESULTS ON THE TEST PARTITION

Table 5 presents results for 82,332 test flow records. The new framework achieves an accuracy of 86.87% and an F1 score of 0.892. The full-feature random forest model has an accuracy of 87.10% and an F1 score of 0.894, higher by only 0.23 percentage points than the new framework. The new framework uses fewer input features than the full-feature random forest model. Both random forest models outperform LR, GNB, L-SVM, and MLP in accuracy and F1 score. Decision tree and KNN models have similar accuracy levels to the new framework but lower AUCs. KNN is 40 times slower than the new framework in flow detection because every training record needs to be compared with a test record.

TABLE 5. Binary detection results on the UNSW-NB15 test partition (82,332 records).

Model	Inputs	ACC (%)	P	R	F1	FAR (%)	AUC	Train (s)	Detect (µs/rec)
LR	68	83.60	0.779	0.981	0.868	34.16	0.965	1.6	0.2
GNB	68	77.75	0.872	0.699	0.776	12.61	0.855	0.2	1.1
DT	68	86.72	0.830	0.954	0.888	23.94	0.868	2.6	0.5
L-SVM	68	83.32	0.773	0.988	0.867	35.65	0.966	1.8	0.2
KNN	68	86.31	0.818	0.967	0.886	26.44	0.932	0.0	986.1

Model	Inputs	ACC (%)	P	R	F1	FAR (%)	AUC	Train (s)	Detect (µs/rec)
MLP	68	84.97	0.792	0.986	0.878	31.73	0.980	28.1	4.0
RF	68	**87.10**	0.816	0.988	**0.894**	27.28	**0.983**	22.8	15.4
Robust z	38	47.14	0.603	0.117	0.195	9.39	0.797	–	–
Mahalanobis	68	43.83	0.000	0.000	0.000	**2.46**	0.778	–	–
PCA-SPE	68	59.67	0.900	0.301	0.451	4.12	0.826	–	–
iForest	68	62.91	0.839	0.404	0.545	9.49	0.808	–	–
Proposed	**34**	86.87	0.813	**0.989**	0.892	27.86	0.979	56.5	24.3

The traditional statistical approaches show reduced false and true positive detection rates (2.5%-9.5% and 0%-40%, respectively) compared with other approaches. The robust Mahalanobis method does not detect any test attacks at the 95th percentile, although its area under the curve is 0.778. Generic and DoS attacks are farther from the center of the normal class than normal flows, with median detection scores of 7.0 versus 2.4. Enterprise production flows contain many normal flows with anomalous features. The 95th percentile of normal training scores (7.79) is higher than the maximum test attack score of 7.17. Stand-alone statistical detection methods may not be suitable for enterprise detection but can be used for ranking. Total training time for the new approach is 56.5 s, including 25.4 s for the five-fold profiling step. Per-record detection time of 24.3 µs enables detection of about 41,000 flows per second on a two-core system.

The confusion matrices for the best-performing models are provided in Table 6. The proposed technique missed 503 attacks, and the full-model random forest technique missed 531 attacks. The proposed technique issued 10,307 false alarms, and the full-model random forest technique issued 10,093 false alarms.

TABLE 6. Confusion matrices on the test partition (rows: actual, columns: predicted).

Model	Actual	Pred. normal	Pred. attack	Total
Proposed	Normal	26,693	10,307	37,000
	Attack	503	44,829	45,332
RF (68 inputs)	Normal	26,907	10,093	37,000
	Attack	531	44,801	45,332

Fig. 2 shows ROC curves. Figure 2a shows little separation and close groupings around the top-left corner among the proposed method, RF, and MLP. Figure 2b again shows relatively flat ROC curves for statistical methods. The best individual detector overall, PCA residual, has an AUC of 0.826. The isolation forest detector has an AUC of 0.808.

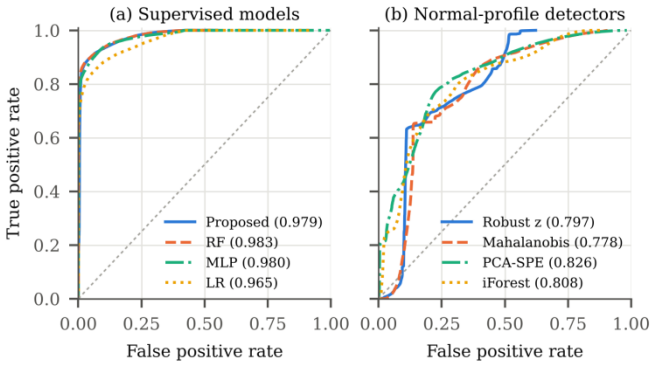


FIGURE 2. ROC curves on the test partition: (a) supervised models; (b) standalone normal-profile detectors. AUC values are given in brackets.

C. DETECTION RATE PER ATTACK CATEGORY

Table 7 lists the proportion of true positives for each attack type in the test data. All attack types are detected with more than 92% accuracy. Worms and Backdoors are detected with 100% accuracy. Generic, Reconnaissance, and DoS attacks are detected with more than 99.8% accuracy. Fuzzers have the lowest detection rate of any attack type in the test data, at 92.77%. Fuzzers create large amounts of traffic with odd payloads but otherwise similar characteristics to normal traffic. There is an improvement of 0.8 percentage points in model accuracy compared to RF for Fuzzers. Both models have low accuracy for classifying normal traffic, with 72% of normal traffic correctly classified.

TABLE 7. Share of correctly classified test records per category (%).

Category	Records	Proposed	RF (68)
Normal	37,000	72.14	72.72
Generic	18,871	99.99	100.00
Exploits	11,132	99.57	99.66
Fuzzers	6,062	**92.77**	91.97
DoS	4,089	99.88	99.93
Reconnaissance	3,496	99.94	99.97
Analysis	677	99.11	100.00
Backdoor	583	100.00	100.00
Shellcode	378	99.47	99.47
Worms	44	100.00	100.00

D. ABLATION STUDY AND FEATURE COUNT

Table 8 presents some feature contributions. Classification accuracy using only four anomaly scores is 60.76%. This measures the same performance as that reported simply using the anomaly scores. Classification accuracy using only the 30 selected traffic features is 86.93%. Adding the statistical scores, the isolation forest score, or all four scores changes classification accuracy by less than 0.15 percentage points. Classification accuracy using all anomaly scores along with all 68 features is 86.73%. Most of the information needed for classification can be found in the selected traffic features. The advantages of using anomaly scores in this experiment include improved interpretability and robustness, not improved classification accuracy.

TABLE 8. Ablation results on the test partition.

Input configuration	Dim.	ACC (%)	F1	FAR (%)	AUC
Four anomaly scores only	4	60.76	0.599	30.03	0.759
MI top-30 features only	30	86.93	0.893	27.55	0.980
MI-30 + three statistical scores	33	86.81	0.892	27.80	0.980
MI-30 + iForest score	31	86.93	0.893	27.55	0.981
All 68 features + four scores	72	86.73	0.891	28.21	0.980
Proposed (MI-30 + four scores)	**34**	86.87	0.892	27.86	0.979

The effect of number of chosen features, K, on results was examined, holding the four scores constant. Accuracy using 10, 20, 30, 40, and 60 features with four scores was 86.65%, 86.49%, 86.87%, 86.86%, and 86.84%, respectively. Accuracy did not vary significantly with K values of 30 or more. Accuracy dropped just 0.22 percentage points when only

10 features and four scores were used. This suggests that the technique could be applied to other uses with memory or other feature limitation concerns.

E. FEATURE IMPORTANCE

Fig. 3. Twelve most important input variables in the trained forest model, measured by average decrease in Gini impurity. The most important input variable is source time-to-live (sttl). Other high-importance input variables include robust z-scores s_z and ct_state_ttl. Four of the seven highest-importance input variables are anomaly scores. Altogether, the four anomaly scores account for 25.1% of the total importance of input variables. Performance of the model without these anomaly scores is similar to performance of the full model (Table 8). These results suggest overlap between information in the anomaly scores and other flow variables. Benefits of the anomaly scores include more intuitive explanations for analysts reviewing alerts, such as “this flow deviates strongly from normal byte and timing statistics”.

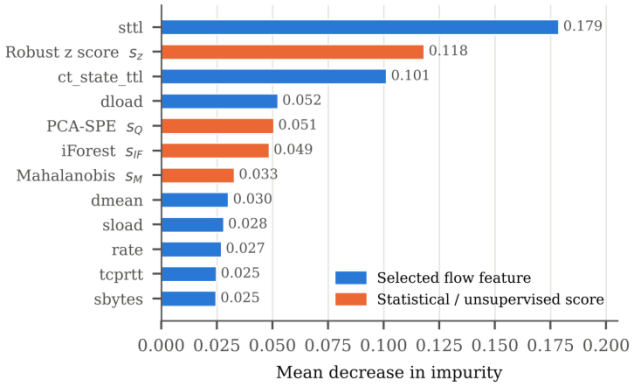


FIGURE 3. Twelve most important inputs of the proposed random forest. Orange bars mark the statistical and unsupervised anomaly scores.

F. CROSS-VALIDATION AND PARTITION SHIFT

Table 9 displays the results of fivefold cross-validation using the training dataset. The proposed method and RF have an accuracy of about 95.98% and a standard deviation of about 0.1%. The false alarm rate is about 8.6%. The cross-validation results shown in Table 9 are much higher than those reported for the test dataset in Table 5. The difference of nine percentage points or so is probably due to differences between the training and test datasets. In particular, the fraction of normal traffic observations in the test dataset is higher (44.9%) than in the training dataset (31.9%), and the distributions of normal traffic observations in the two datasets are different. Cross-validation results alone would have overstated expected performance in this setting. Stand-alone test data, as used in [12], are recommended for enterprise applications.

TABLE 9. Five-fold cross-validation on the training partition (mean ± standard deviation).

Metric	Proposed	RF (68)
ACC (%)	95.98 ± 0.11	95.98 ± 0.12
Precision	0.961 ± 0.001	0.961 ± 0.001
Recall	0.981 ± 0.001	0.981 ± 0.001
F1-score	0.971 ± 0.001	0.971 ± 0.001
FAR (%)	8.56 ± 0.14	8.55 ± 0.11
AUC	0.994 ± 0.000	0.994 ± 0.000

G. DECISION THRESHOLD ANALYSIS

The high FAR at τ=0.5 is a practical concern for SOC operations. Figures 4 and 10 show the impacts of different τ values on other detection statistics. These impacts were assessed in a sensitivity analysis using test data and were not a factor in selecting the model in Table 5. Recall at τ values up to 0.7 is greater than 95%, and FAR declines as τ increases. Other

statistics at τ=0.8 are 92.26% accuracy, 94.1% precision, 92.9% F1 score, and 7.04% FAR. The number of false positives declines from 10,307 to 2,605, and detection rate declines slightly to 91.7%. FAR declines to less than 2% at τ=0.9. Companies using this approach should use their own network data to determine an appropriate τ value based on the number of alerts analysts can respond to.

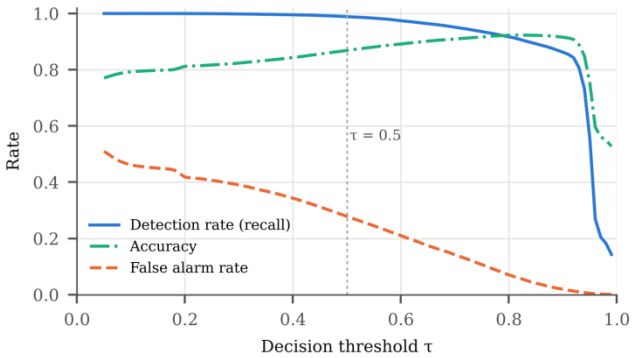


FIGURE 4. Effect of the decision threshold τ on detection rate, accuracy and false alarm rate of the proposed framework (test partition).

TABLE 10. Operating points of the proposed framework for different thresholds (test partition).

τ	ACC (%)	P	R	F1	FAR (%)	False alarms
0.3	82.38	0.758	0.998	0.862	39.01	14,432
0.5	86.87	0.813	0.989	0.892	27.86	10,307
0.6	89.15	0.850	0.974	0.908	20.99	7,766
0.7	90.95	0.892	0.951	0.920	14.18	5,246
0.8	**92.26**	0.941	0.917	**0.929**	7.04	2,605
0.9	91.56	0.982	0.863	0.918	1.94	717

VI. DISCUSSION

The experiments have four main findings. First, combining machine learning techniques with statistical techniques improves an intrusion detection model compared to using only statistical techniques. Intrusion detection performance improved from a high of 40% with only statistical techniques to almost 99% with the combined techniques. Second, the combined approach has about the same intrusion detection performance as a random forest model with all features, but with about half as many features. The model also had close to the same performance with just ten flow features. Cutting the number of features required to achieve good detection performance reduces costs for gathering information on enterprise networks. Third, anomaly scores are a critical input to the combined model and can aid human analysts in assessing intrusion alerts. This addresses a common concern about the usefulness of machine learning models to analysts [1]. Fourth, the out-of-fold modeling approach helps prevent overfitting of the combined techniques. Modeling approaches without the out-of-fold technique may give overly positive assessments to training data that will not hold up with new data.

The study has multiple limitations. Statistical features did not help on the UNSW-NB15 test set after controlling for other traffic features. Statistical features may help on other types of traffic and other datasets with limited or no labeled attacks, but this was not explored in the study. Like other supervised approaches [6], [7], the proposed approach requires knowing attack types for training data. Large discrepancies between cross-validation and test results indicate changes in normal traffic over time [17], [12]. We recommend periodically updating normal traffic profiles with recently observed normal traffic. The study was limited to binary detection methods and binary threshold selection on test data. A separate validation set should be used to select binary thresholds for deployed detection systems.

VII. CONCLUSION

This paper describes an intelligent system for detecting anomalies in enterprise networks that draws on both statistical and machine learning methods. Four normal profiles (a profile based on a robust z-score, a profile based on a robust Mahalanobis distance, a profile based on principal components residuals, and an isolation forest profile) were created and used as anomaly scores. Thirty inputs were selected using mutual information and combined with the four anomaly scores. A random forest classifier was used to classify the results. Out-of-fold scoring was used to avoid contamination between the profiling and classification stages. On the official UNSW-NB15 data set test data, the new system achieved an accuracy rate of 86.87%, a detection rate of 98.89%, an F1-score of 0.892, and an AUC of 0.979 using 34 inputs. Results are similar to those for a 68-input random forest model and better than those for the four stand-alone statistical models. Setting the anomaly score threshold to 0.8 reduced false positives to 7.04% and the attack detection rate to 91.70%. Planned future work includes deploying the system in enterprise network environments and adding mechanisms for updating normal profiles in real time [16]. Planned future projects also include extending the system to detect multiclass attacks and testing other approaches to selecting thresholds with validation data.

REFERENCES

[1] C. A. Catania and C. Garcia Garino, "Automatic network intrusion detection: Current techniques and open issues," *Comput. Electr. Eng.*, vol. 38, no. 5, pp. 1062–1072, 2012, doi: 10.1016/j.compeleceng.2012.05.013.

[2] M. Thottan, G. Liu, and C. Ji, "Anomaly detection approaches for communication networks," in *Algorithms for Next Generation Networks* (Computer Communications and Networks). London, U.K.: Springer, 2010, pp. 239–261, doi: 10.1007/978-1-84882-765-3_11.

[3] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput.*, vol. 22, pp. 949–961, 2019, doi: 10.1007/s10586-017-1117-8.

[4] S. S. Kim and A. L. N. Reddy, "Statistical techniques for detecting traffic anomalies through packet header data," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 562–575, 2008, doi: 10.1109/TNET.2007.902685.

[5] D. J. Weller-Fahy, B. J. Borghetti, and A. A. Sodemann, "A survey of distance and similarity measures used within network intrusion anomaly detection," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 70–91, 2015, doi: 10.1109/COMST.2014.2336610.

[6] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, 2018, doi: 10.1109/TETCI.2017.2772792.

[7] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019, doi: 10.1109/ACCESS.2019.2899721.

[8] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4815–4830, 2019, doi: 10.1109/JIOT.2018.2871719.

[9] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Comput. Secur.*, vol. 70, pp. 255–277, 2017, doi: 10.1016/j.cose.2017.06.005.

[10] A. J. Malik, W. Shahzad, and F. A. Khan, "Network intrusion detection using hybrid binary PSO and random forests algorithm," *Secur. Commun. Netw.*, vol. 8, no. 16, pp. 2646–2660, 2015, doi: 10.1002/sec.508.

[11] Y. Meng and L.-F. Kwok, "Enhancing false alarm reduction using voted ensemble selection in intrusion detection," *Int. J. Comput. Intell. Syst.*, vol. 6, no. 4, pp. 626–638, 2013, doi: 10.1080/18756891.2013.802114.

[12] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments," *Comput. Netw.*, vol. 127, pp. 200–216, 2017, doi: 10.1016/j.comnet.2017.08.013.

[13] S. Rastegari, P. Hingston, and C.-P. Lam, "Evolving statistical rulesets for network intrusion detection," *Appl. Soft Comput.*, vol. 33, pp. 348–359, 2015, doi: 10.1016/j.asoc.2015.04.041.

[14] A. R. Vasudevan and S. Selvakumar, "Local outlier factor and stronger one class classifier based hierarchical model for detection of attacks in network intrusion detection dataset," *Front. Comput. Sci.*, vol. 10, no. 4, pp. 755–766, 2016, doi: 10.1007/s11704-015-5116-8.

[15] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Inf. Sci.*, vol. 177, no. 18, pp. 3799–3821, 2007, doi: 10.1016/j.ins.2007.03.025.

[16] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 34–47, 2017, doi: 10.1109/TNSM.2016.2627340.

[17] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, 2013, doi: 10.1016/j.neucom.2012.11.050.

[18] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018, doi: 10.1109/ACCESS.2018.2869577.

[19] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *J. Supercomput.*, vol. 75, no. 9, pp. 5597–5621, 2019, doi: 10.1007/s11227-019-02805-w.

[20] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 2019, doi: 10.1109/COMST.2019.2896380.