Monshizadeh, Mehrnoosh; Khatri, Vikramajeet; Atli, Buse; Kantola, Raimo; Yan, Zheng

Performance Evaluation of a Combined Anomaly Detection Platform

Published in:
IEEE Access

DOI:
10.1109/ACCESS.2019.2930832

Published: 24/07/2019

Document Version
Publisher's PDF, also known as Version of record

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ARTIFICIAL INTELLIGENCE IN CYBERSECURITY

Received June 14, 2019, accepted July 15, 2019, date of publication July 24, 2019, date of current version August 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930832

Performance Evaluation of a Combined Anomaly Detection Platform

MEHRNOOSH MONSHIZADEH[1,2], VIKRAMAJEET KHATRI[3], BUSE GUL ATLI[4], RAIMO KANTOLA[2], AND ZHENG YAN[2,5]

[1]Nokia Bell Labs, 91620 Nozay, France
[2]Department of Comnet, Aalto University, 02150 Espoo, Finland
[3]Nokia Bell Labs, 02610 Espoo, Finland
[4]Department of Computer Science, Aalto University, 02150 Espoo, Finland
[5]The State Key Lab of ISN, Xidian University, Xi'an 710071, China

Corresponding author: Mehrnoosh Monshizadeh (mehrnoosh.monshizadeh@nokia-bell-labs.com)

ABSTRACT Hybrid Anomaly Detection Model (HADM) is a platform that filters network traffic and identifies malicious activities on the network. The platform applies data mining techniques to tackle effectively the security issues in high load communication networks. The platform uses a combination of linear and learning algorithms combined with protocol analyzer. The linear algorithms filter and extract distinctive attributes and features of the cyber-attacks while the learning algorithms use these attributes and features to identify new types of cyber-attacks. The protocol analyzer in this platform classifies and filters vulnerable protocols to avoid unnecessary computation load. The use of linear algorithms in conjunction with learning algorithms and protocol analyzer allows the HADM to achieve improved efficiency in terms of accuracy and computation time to detect cyber-attacks over existing solutions. While authors' previous paper evaluated HADM efficiency (accuracy and computation time) against related studies, this paper, concentrates on HADM robustness and scalability. For this purpose, five datasets, including ISCX-2012, UNSW-NB15 Jan, UNSW-NB15 Feb, ISCX-2017, and MAWILab-2018, with various size and diverse attacks have been used. Different feature selection methods are applied to find the best features. The feature selection methods are selected based on the algorithms' computation time and detection rate. The best algorithms are then selected through a benchmark on applied datasets and based on the metrics such as cross-entropy loss, precision, recall, and computation time. The result of HADM platform shows robustness and scalability against datasets with different size and diverse attacks.

INDEX TERMS Anomaly detection, data mining, feature selection, machine learning, security.

I. INTRODUCTION

Intrusion Detection Systems (IDSs) are considered well-known tools for monitoring and detection of malicious traffic in communication networks. However, IDS is a technology that uses highly developed and complex algorithms for processing large volumes of data [1]. The complexity of the algorithms results in long computation time. IDS captures network traffic in real time and compares the received packet patterns with known patterns to detect anomalies in network. Yet the cost and high processing time to handle traffic load is a challenge in IDS.

To solve this problem, network traffic flow control in combination of Data Mining (DM) techniques are proposed in Hybrid Anomaly Detection Model. HADM platform consists of two main parts where each part independently increases the efficiency of attack detection based on the factors such as precision, recall, accuracy and computation time. While part 1 of the model utilizes some algorithms and the protocol analyzer for traffic filtering, reducing the processing time and increasing the accuracy, the part 2 applies a dynamic feature selection with a genetic algorithm to classify unknown attacks and increase the accuracy as well. HADM model comprises a protocol analyzer, linear and learning algorithms as well as other modules. Since some protocols such as streaming protocols are not vulnerable, and attackers usually target specific

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Quan.

protocols, protocol analyzer in this platform classifies and filters vulnerable protocols to avoid unnecessary computation load. Protocol analyzer forwards the filtered traffic either to a linear algorithm only for Denial of Service (DoS) detection or to a combination of a linear and a learning algorithm for other types of attacks. The linear algorithm initially defines if the traffic is secure or unsecure regardless of the attack type. In addition, it extracts the proper features in order to provide them to a learning algorithm in order to classify already known attacks and detect unknown attacks. Another counter measure located after the learning algorithm extracts information about known attacks (against which network is already protected) from other deployed security mechanisms in the network e.g., firewall, IDS, DPI etc. It compares the extracted information with the attack received from the linear algorithm and drops similar attack flows. In each step, feedback is sent to a database for next level detection. In the learning algorithm, the received attack is assigned to one of the attack clusters. In addition, algorithm changes its structure and input weights dynamically based on the received feedback. If the attack does not belong to any of the mentioned clusters, it is assigned to a totally new cluster. This novel mechanism dynamically defines new features in order to detect new types of attacks.

The protocol analyzer in HADM platform classifies and filters vulnerable protocols to avoid unnecessary computation load. On the other hand, each data set includes hundreds of features that may cause performance degradation in the detection process. To overcome this problem, feature selection methods are used to select a smaller number of features and reduce the dimensions of the dataset [1]. In addition, the use of linear algorithms in conjunction with learning algorithms improves accuracy and reduces computation time. Linear algorithms will detect the attack in general level regardless of their types while learning algorithms cluster the attacks in different categories. This mechanism decreases the load of input data for the learning algorithm that are the most time-consuming part because of their complexity [2]. The major differences and novelty of this paper against related works are as follow:

- Most surveys that cover security mechanisms such as existing IDS (i.e., signature, statistical, supervised, unsupervised, etc.) have very limited focus on hybrid IDS techniques, while the paper [7] not only provides a comprehensive review of a hybrid platform but also it applies a protocol analyzer, taking intrusion detection to the next level.
- We are the first to discuss and apply the concept of protocol analyzer to reduce the IDS load.
- Feature extraction and selection techniques play important roles in intrusion detection as they influence their learning processes. Unlike other studies that apply specific feature selection method with a particular algorithm, this paper discusses the topic thoroughly and applies various feature selection methods along with different algorithms to find the best combination.
- None of the related studies have evaluated the IDS robustness and scalabilities against various datasets.

Overall, the contribution of this paper is in introducing an efficient (considering computation time and accuracy) anomaly detection platform with several characteristics:

- We propose a novel platform that comprises several modules such as protocol analyzer and combined algorithms
- We take advantage of the various feature selection methods to select the best features from data and apply different combination of learning and linear algorithms to identify attacks efficiently.
- We integrate our algorithms to protocol analyzer that filters vulnerable protocols reducing the platform load
- We extensively evaluate our system robustness and scalability over several metrics, different datasets and various attacks to deliver a proof of concept, which is supported by experimental results. And for this purpose, we:
  - ➢ Survey different datasets and their pros and cons to select more prominent datasets for our testing.
  - ➢ Evaluate the platform efficiency based on uniform evaluation metrics including computation time, precision and recall.
  - ➢ Evaluate scalability and robustness by applying five different datasets, ISCX-2012, UNSW-NB15 Jan, UNSW-NB15 Feb, ISCX-2017 and MAWILab-2018 that contains diverse attacks.

The rest of the paper is organized as follows. Section 2 provides a brief background in intrusion detection. Section 3 gives an introduction about the platform, its algorithms, and metrics. In Section 4, HADM implementation along with datasets and data preprocessing are discussed. Section 5 discusses the experimental results. Finally, in Section 6, we draw conclusion along with scope of future research.

II. RELATED WORK

Di Pietro *et al.* [3] applies machine learning algorithms such as k-nearest neighbor (k-NN) and Support Vector Machine (SVM) to detect anomalies. Furthermore, a Deep Packet Inspection (DPI) mechanism is utilized to define rules for capturing packets. However, the rules, protocols and details of the process is not explained. In addition, authors have not discussed their model scalability and robustness neither any experimental result is presented in this study.

Vasseur *et al.* [4] propose a supervised learning classifier to detect DDoS attack. This study applies Deep Neural Networks (DNN) classifier, which mainly concentrates on optimizing training process in order to provide labeled data. However, not only this study introduces a combined method, but also mentioned algorithm is utilized only for DDoS detection rather than other types of attacks. In addition, authors have not discussed their model scalability and robustness neither any experimental result is presented.

Pietro *et al.* [5], apply a machine learning based model comprises of ANN to compare received traffic with

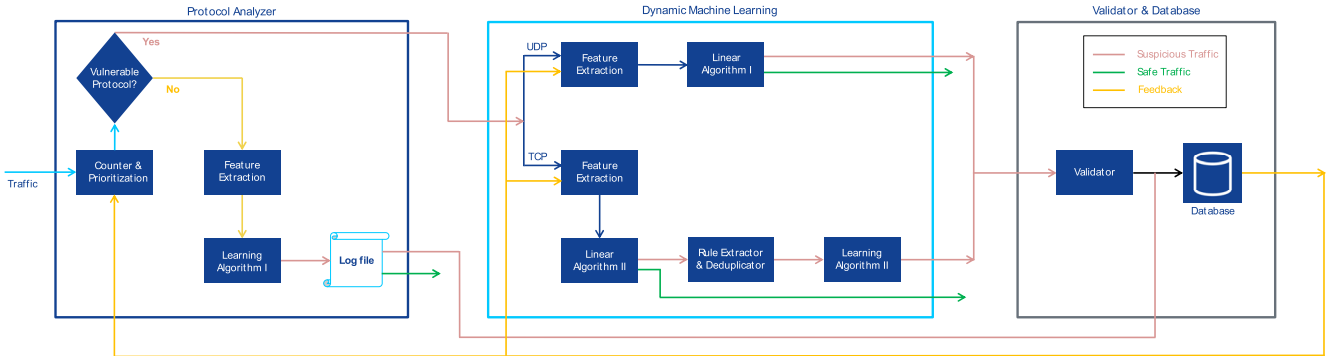


FIGURE 1. Hybrid anomaly detection model (HADM).

expected traffic. Presented model is trained with expected traffic and upon receiving the input data, the signature of data is compared with the expected traffic. If they are different, a signature for the attack class will be generated and model will be trained with new information. This study doesn't discuss any types of attack neither the implementation result is presented.

Yadav *et al.* [6], propose a Virtual Machine (VM) based analytic model to detect anomalies within the network traffic based on the dynamic modeling of network behavior. They have applied honeypot to collect malicious traffic. Though, the model comprises of unsupervised and supervised machine learning algorithms, the honeypot relies only on received attacks and not the other attacks. In this study, applied algorithms is not disclosed and experimental result is not presented. In addition, authors have not discussed the scalability and robustness of their model.

III. HADM PLATFORM

The HADM comprises a protocol analyzer, linear and learning algorithms, validator and database as shown in Fig. 1.

The mentioned components are deployed in conjunction with one another to filter packets on the communication networks, such as mobile networks and for certain network protocols that are known or considered to be vulnerable to or used in cyber-attacks. This allows the HADM to expend a smaller amount of processing resource on other network protocols, such as streaming protocols that are not normally vulnerable and thus not typically targeted by cyber-attackers. The ability of the HADM to focus on vulnerable network protocols helps to avoid burdening network servers with unnecessary computational load. The protocol analyzer filters the network packets and identifies vulnerable protocols. The non-vulnerable protocols are forwarded to the feature extraction module for further processing. The feature extraction module extracts features from the incoming packets and provides these features to the learning algorithm I for the analysis. If the output from learning algorithm I is suspicious, it is recorded into log file. If traffic is carried on vulnerable protocol, the counter and prioritization module forwards the suspicious traffic to next level based on the occurrence of protocol against a defined threshold.

For certain suspected attacks, such as Denial of Service (DoS) attacks carried on User Datagram Protocol (UDP), the protocol analyzer forwards the filtered packets to feature extraction module and linear algorithm I. For other suspected attacks carried over Transmission Control Protocol (TCP), the protocol analyzer forwards the filtered packets to the feature extraction module and linear algorithm II. The linear algorithm II initially defines whether the packets are safe or unsafe regardless of the suspected attack type, then extracts the features of the suspected attack and provides them to the learning algorithm II. The learning algorithm II compares the extracted features against known attack features and classifies the suspected attack as either known or unknown. In case of unknown attack, they are labeled. The information about attack is then shared to the validator and database component.

The validator and database component validate the output of the linear and learning algorithms. If the actual output (e.g., from the learning algorithm) differs from the expected output, then the actual output is considered as an error. The expected output refers to numeric values that are predefined by a user and represent safe traffic. The actual output contains numeric values assigned to the features and attributes from the output of the learning algorithm. The comparison is done based on the values of these traffic features. The validator output is stored into database component, the attack features are provided as feedback to the protocol analyzer and the linear and learning algorithms for use in subsequent detections. Such an arrangement allows the HADM to define dynamically new attack features in order to better identify new types of cyber-attacks.

A. APPLIED ALGORITHMS

For performance testing, the selected features are applied to six different algorithms including Extreme Learning Machine (ELM), Multi-Layer Perceptron (MLP), SVM, k-Nearest Neighbor (k-NN), Decision Tree (DT) and Logistic Regression (LR). The best algorithms were selected through a benchmark on applied datasets and comparing the results using metrics like accuracy, False Positives (FPs), False Negatives (FNs), training and testing time. The applied algorithms are described below.

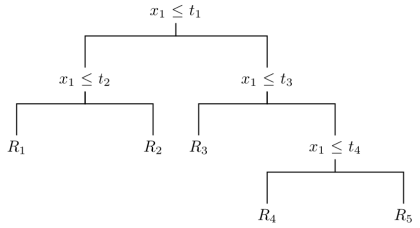


FIGURE 2. Simple decision tree with 5 regions.

LR: LR is a supervised learning method, which maps input $x \in \mathbb{R}^d$ to the output $y \in \{0, 1\}$ by constructing a linear function with weight vector $w = [w_1, w_2, \ldots.., w_d]$ and calculates the probability of the output given the input as $P(y|x)$. The predicted class maximizes this parametric form. This method finds strong relationships in input features. However, model may not converge when the training data is small or the decision boundary between two classes are highly non-linear [8].

DT: DT is a non-parametric machine learning method, which is based on binary splitting features. Each leaf node in a DT represents a specific region or class with different data characteristics, as illustrated in Fig. 2. DT is easily interpretable but might be unstable, since it is affected by the variance in the training set [8].

k-NN: KNN is an unsupervised machine learning method that requires no knowledge about the prior distribution of the data and true class labels [8]. It first selects some initial seeds (initial training samples) and then groups the data by comparing the distance of other training samples to these seeds by using a similarity measure. In each iteration, the class of a new training sample is decided by the class of its k nearest neighbors. In experiments, 5 was selected as k and weight of each the nearest neighbor to the final decision was equally distributed. Euclidean distance was selected as the similarity measure.

SVM: SVM constructs a hyperplane in a high-dimensional space by mapping input data to a higher dimensional space using kernel methods in order to create a non-linear decision boundary. SVM has a high accuracy in many applications but the time complexity of it is quite high [8].

MLP: An MLP is a deep, feed-forward, artificial neural network including more than one perceptron and different layers. It includes an input layer to receive the signal (input data), an output layer to give a probability vector for predictions or only one prediction and a different number of hidden layers in order to represent the input vector in a more abstract form. A single perceptron in each layer calculates a weighted sum of the input and applies a non-linear activation function to this weighted sum. The output of one perceptron is fed as an input to the perceptron of the next layer.

During training, MLP accepts the input x, forwards the information from layer to layer using its parameters θ (weights and biases) and produces an output y' as well as a scalar cost $J(x; y; \theta)$ between the original class y and the predicted class y'. With a back-propagation algorithm,

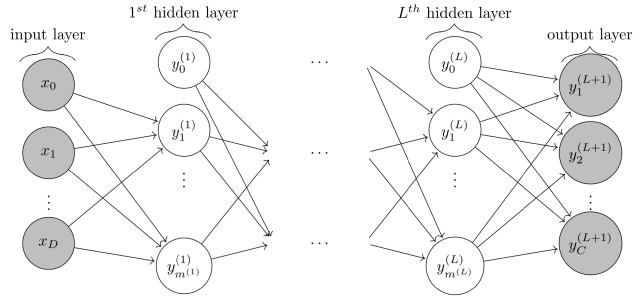


FIGURE 3. Multi-layer perceptron with input layer, output layer and hidden layers.

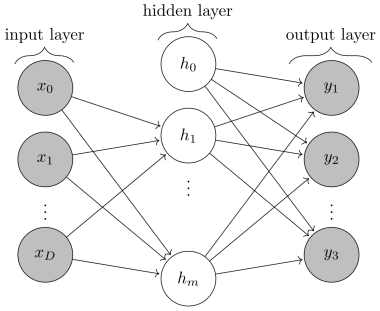


FIGURE 4. Structure of an extreme learning machine.

it calculates the partial derivative of the cost function (gradient) with respect to its parameters. It updates weights and biases using gradient values. The back-propagation is applied in each iteration (epoch) until the convergence of the parameters or the convergence of test error. A more detailed information about back-propagation is given in [9]. For parameter update, different gradient optimization techniques can be used such as Stochastic Gradient Descent (SGD), Momentum, RMSProp or Adam [9].

In experiments, 2 different MLP architectures are constructed, one with 10 (MLP10) and other with 50 (MLP50) perceptron in one hidden layer. Each perceptron had a logistic activation function, cross-entropy loss was selected as the cost function and Adam optimization technique was used in back-propagation.

ELM: It is a type of single hidden-layer feedforward neural network proposed in [10]. Fig. 4 shows that ELM has a similar architecture to MLP. However, ELM does not use back-propagation to update its parameters. ELM randomly initializes its input weights and updates only the parameters connecting the hidden layer and the output layer in order to reduce the computational time while ensuring the robustness [8].

Consider a training data with N samples $\{(x_i, y_i)\}_{1 \leq i \leq N}$ where $x_i \in \mathbb{R}^d$ is input data and $y_i \in \mathbb{R}^c$ is the true output. Then, ELM can approximate the input to the true output as:

$$\sum_{j=1}^{M} \beta_j \phi(w_j^T x_i + b_j) = y_i \tag{1}$$

where w_j is the weight vector connecting input layer to the hidden layer, b_j is the bias of the j^{th} hidden node, β_j is the

output weight vector connecting the j^{th} hidden node to nodes in the outer layer and $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is the activation function. The equation can be represented in a matrix form $H\beta = Y$, and H is a $N \times M$ matrix, where N denotes the number of training samples and M denotes the number of nodes in the hidden layer.

ELM uses Least-Squares Method (LSM) and tries to minimize the squared Euclidean norm of the error matrix $||H\beta - Y||^2$ in order to update the β matrix. The matrix β can be calculated easily if H was singular: $\beta = H^{-1}Y$. However, H is not singular in most cases, since the number of hidden nodes is less than the number of training samples [8]. Therefore, ELM finds a pseudoinverse of the solution of the system as:

$$\beta^{\wedge} = \left(H^T H\right)^{-1} HY \tag{2}$$

Once an approximate solution $\beta^{\wedge}$ is found, the weights connecting the hidden and output layer are updated and the training procedure is finished.

In experiments, we used 10 and 50 neurons in hidden layers (ELM10 and ELM50) and sigmoid function as an activation function.

B. FEATURE SELECTION

The datasets involve different features that are often classified into below groups:

1) Flow features: this group includes the identifier attributes between hosts, such as client-to-server or server-to-client.
2) Basic features: this category involves the attributes that represent protocols connections.
3) Content features: this group encapsulates the attributes of TCP/IP; also, they contain some attributes of HTTP services.
4) Time features: this category contains the attributes of time, for example, arrival time between packets, start or end packet time and round-trip time of TCP protocol.
5) Additional generated features: this category can be further divided into two groups:
   a. General purpose features where each feature has its own purpose, in order to protect the service of protocols.
   b. Connection features are built from the flow of 100 record connections based on the sequential order of the last time feature.
6) Labelled Features: this group represents the label of each record [11].

However, network packets also carry a wide variety of irrelevant or redundant features. In this section, feature characteristics of our datasets are examined to remove the unwanted features that affect the efficiency and detection rate of our algorithms. For this purpose, we apply different feature selection methods such as Chi2, F-Score, SVMonline and RFE to find the best features from the datasets. The feature selection methods are chosen based on the achieved efficiency considering testing time and detection rates. The utilized feature selection algorithms are described below.

1) **Chi2:** Chi square measures the dependency between a feature and a class by counting the occurrence of the feature with respect to occurrence of the class. Chi2 is simple but effective if a feature with a certain distribution can be differentiated easily in normal and attack packets [12]. In this method, features with highest scores are selected.
2) **F-Score:** The best feature subset includes features having a high linear relationship with a class. Although the calculated correlation value captures strong relationship between features and labels, it fails to detect non-linear relationships [12].
3) **SVMonline:** Incremental SVM calculates the loss and retrains linear SVM in every batch using stochastic gradient descent. It assigns SVM weights to each feature and selects those with highest absolute value as best discriminative features. Although SVMonline relies on linear dependency of features and labels as in F-Score, it is more robust than F-Score, since it splits the dataset into small batches and calculates the average of model coefficients that further increases the robustness [13].
4) **RFE:** This method first calculates the importance of each feature from a full features list based on a trained estimator, which can be a simple machine learning algorithm. Then, RFE removes features having the least importance value from the subset recursively until a desired length of feature list is reached. RFE was tested by selecting logistic regression as the estimator [14].

C. EVALUATION METRICS

To evaluate HADM detection rate, applied metrics such as cross-entropy loss, accuracy score, precision, recall and F1 score are briefly explained. We consider four classes: normal, unknown, other attacks and DoS (−1, 0, 1, 2) for SVM, k-NN, DT and LR. On the other hand, we consider 3 classes: normal, unknown and other attacks (0, 1, 2) for ELM and MLP.

1) **Cross-entropy loss:** Entry i and j in a confusion matrix are the number of observations actually in group i, but predicted to be in group j. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.
   If the actual probability is p_i but the predicted probability is q_i, each event will occur with the probability of p_i but surprisal will be given by q_i in its formula. The weighted average surprisal, in this case, is cross-entropy (c) loss and it is calculated as:

$$c = \sum_{i=0}^{n} p_i log\left(\frac{1}{q_i}\right) \tag{3}$$

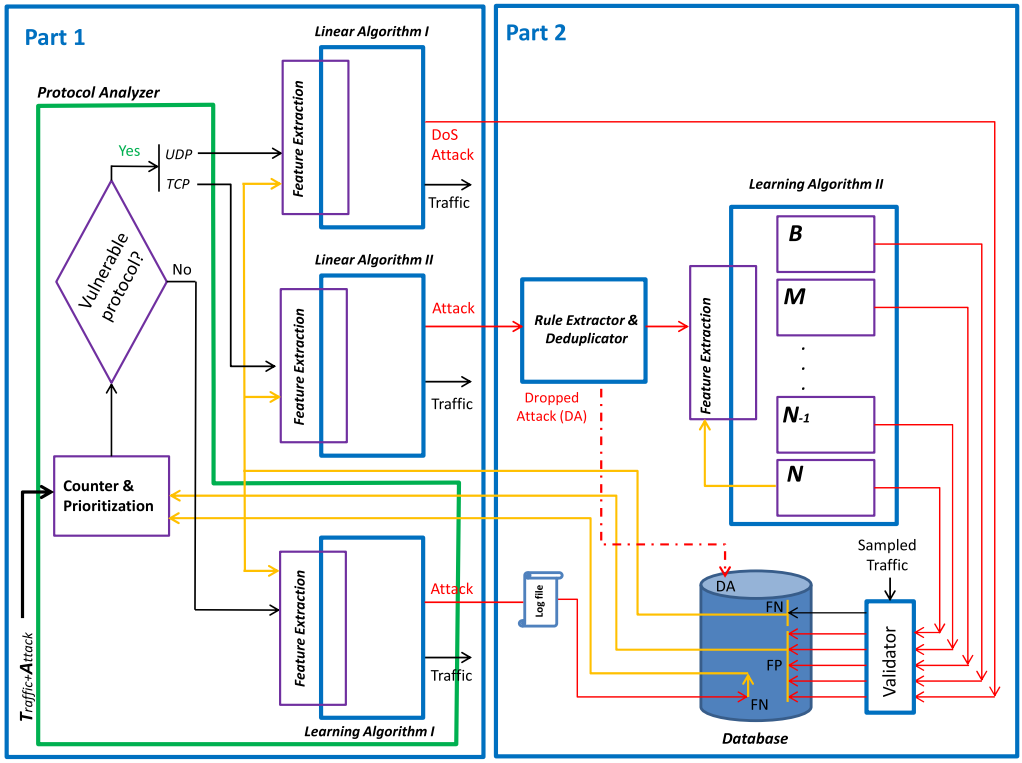


FIGURE 5. Two parts of hybrid anomaly detection model (HADM).

In the case of binary classification where we have only two classes, we name it as binary cross-entropy loss and the above formula becomes:

$$c = \sum_{0}^{1} p_i log\left(\frac{1}{q_i}\right) = p_0 log\left(\frac{1}{q_0}\right) + p_1 log\left(\frac{1}{q_1}\right)$$
$$= p_0 \log\left(\frac{1}{q_0}\right) + (1 - p_0)\left(\frac{1}{1 - q_0}\right) \quad (4)$$

2) **Accuracy score:** It computes the count of correct predictions:

$$Accuracy\left(y, y'\right) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1\left(y_i' = y_i\right) \quad (5)$$

In (5), $y_i^{'}$ refers to the predicted value of i$^{\text{th}}$ sample, y_i refers to the corresponding true value and $1\,(x)$ is the indicator function.

3) **Precision:** It is the ability of a classifier not to wrongly label a negative sample as positive. In other words, how many of the selected objects were correct. Precision is calculated with:

$$Precision = \frac{TP_i}{TP_i + FP_i} \quad (6)$$

where,

- TP_i or True Positive: Is the number of instances with actual class other than the *i-th,* and correctly predicted to belong in the *i-th* class. This metric represents the malicious traffic that correctly identified as attack.
- FP_i or False Positive: Is the number of instances with actual class other than the *i-th*, but wrongly predicted to belong in the *i-th* class. This metric represents the safe traffic incorrectly identified as attack.

4) **Recall:** It refers to ability of a classifier to find all positive samples. In other words, how many of the objects that should have been selected were actually selected. Recall is calculated with:

$$Recall = \frac{TP_i}{TP_i + FN_i} \quad (7)$$

where,

- FN_i or False Negative: Is the number of instances with *i-th* being the actual class, but falsely predicted to belong to another class. This metric represents the malicious traffic that incorrectly identified as safe traffic.

5) **F1 score:** It is the weighted average of the precision and recall and is calculated with:

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (8)$$

IV. IMPLEMENTATION PHASES

As Fig. 5 shows, the implementation of HADM platform is divided into two parts. This paper concentrates on the part 1 to detect all attacks on a general level and DoS attack in particular. The part 2 will be introduced in a separate paper as an innovative method to label specific types of attacks applying a dynamic feature selection mechanism.

TABLE 1. Comparison between different publicly available datasets.

Dataset	Year	Labelled	Payload	Protocol	Attacks	Comments
DARPA		Yes	No	HTTP, SSH, FTP, email (browsing, telnet, IRC[a], SNMP[b])	DoS, Scan, Brute force, others (guess password, buffer overflow, remote FTP, SYN[c] flood)	not real-world traffic, lack of FP, out of date
KDD'99	1999	Yes	No	HTTP, SSH, FTP, email (tcpdump)	DoS, Scan, Brute force, others (Neptune DoS, pod-DoS, Smurf-DoS, buffer-overflow)	Redundant records
DEFCON		No	No	HTTP, SSH	Scan, Backdoor, others (port scanning, buffer-overflow, seeps, administrative privilege, telnet protocol attack)	not real-world network traffic
CAIDA		No	Yes		DoS, Scan, DNS[d], others	Limited to specific attacks, anonymized payload, IP[f] and protocols
LBNL		No	Yes	HTTP, SSH,	Scan	Medium size data, heavily anonymized
CDX		No	No	HTTP, SSH, FTP, email	DoS, Scan, DNS (Nikto, Nessus, WebScarab)	Lack of traffic diversity and volume
KYOTO		Yes	No	HTTP, HTTPS[e], SSH, FTP, email	DoS, Scan, Backdoor, DNS, Brute force, Browser, others	Limited to traffic collected by honeypot, lack of FP
TWENTE		Yes	Yes	HTTP, SSH, FTP (open SSH, apache and webserver)	Scan, Brute force, others	Lack of volume and attack diversity
UMASS		Yes	No	HTTP	others	Lack of traffic and attack diversity
ADFA2013	2013	Yes	Yes	HTTP, SSH, FTP, email	Backdoor, Brute force, Browser, others (Java based meterpreter, Linux Meterpreter, Add new Superuser, WebShell attack)	Lack of attack and features diversity
TUIDS	2018	Yes	No	TCP, UDP, ICMP	SYN, RST[f], UDP, Ping, Fraggle and Smurf flood attacks.	Anonymized, focused only on DoS flooding attacks.
MAWILab	2018	Yes	Yes	HTTP, HTTPS, SSH, FTP, DNS, email, BGP[g]	DoS, Scan, Worm, Browser, Brute force, DNS, others (SMB[h], RPC[i], RST, Sasser, Netbios, SYN, FIN[j], Ping flood, FTP, SSH)	Labelled using several anomaly detection classifiers, real world traffic with diverse attacks.
ISCX-2012	2012	Yes	No	HTTP, SSH, FTP, email	DoS, Scan, Backdoor, Brute force, Browser	Lack of HTTPS while most of the today's traffic is HTTPS
ISCX-2017	2017	Yes	Yes	HTTP, HTTPS, FTP, SSH, email	Web based, Brute force, DoS, DDoS[k], Infiltration, Heart-bleed, Bot, Scan	It is metadata with diversity in features, attack and traffic, real world traffic
UNSW-NB15 Jan	2015	Yes		TCP, UDP, ICMP[l]	Fuzzer, analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode, Worm	diversity in features
UNSW-NB15 Feb	2015	Yes		TCP, UDP, ICMP	Fuzzer, analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode, Worm	diversity in features

[a]IRC Internet Relay Chat
[b]SNMP Simple Network Management Protocol
[c]SYN A TCP flag to synchronize sequence numbers
[d]DNS Domain Name System
[e]HTTPS HyperText Transfer Protocol Secure
[f]RST A TCP flag to reset the connection
[g]BGP Border Gateway Protocol
[h]SMB Server Message Block
[i]RPC Remote Procedure Call
[j]FIN A TCP flag to indicate no more data from sender
[k]DDOS Distributed Denial of Service
[l]ICMP Internet Control Message Protocol

A. DATASET

Reliable and publicly available datasets are important for evaluating IDSs. Here we briefly compare 16 datasets that have been publicly available since 1998. Majority of mentioned datasets are small, they do not have attack or traffic diversity and are usually anonymized. Therefore, we selected

TABLE 2. Distribution of packets in datasets.

Dataset	DoS	Other attacks	Unknown	Normal	Total
ISCX-2012	67697	197569	3067797	1664006	4997069
UNSW-NB15 Jan	746	9680	739108	748272	1497806
UNSW-NB15 Feb	3100	35276	471484	465158	975018
ISCX-2017	1427966	282398	15096260	38161934	54968558
MAWILab-2018	0	15832866	18061933	36672050	70566849

TABLE 3. Distribution of Distribution of training and testing packets in datasets.

Dataset	Algorithm	Data	DoS	Other attacks	Unknown	Normal	Total
ISCX-2012	DT / LR	Training (2/3)	45131	131712	131712	131712	440267
		Testing (1/3)	22566	65857	65857	65857	220137
	ELM / MLP	Training (2/3)		37794	37794	37794	113382
		Testing (1/3)		18897	18897	18897	56691
UNSW-NB15 Jan	k-NN / SVM	Training (2/3)	497	497	497	497	1988
		Testing (1/3)	249	249	249	249	996
	ELM / MLP	Training (2/3)		1822	1822	1822	5466
		Testing (1/3)		912	912	912	2736
UNSW-NB15 Feb	k-NN / SVM	Training (2/3)	2066	2066	2066	2066	8264
		Testing (1/3)	1034	1034	1034	1034	4136
	ELM / MLP	Training (2/3)		6205	6205	6205	18615
		Testing (1/3)		3103	3103	3103	9309
ISCX-2017	DT / LR	Training (2/3)	188265	188265	188265	188265	753060
		Testing (1/3)	94133	94133	94133	94133	376532
	ELM / MLP	Training (2/3)		161730	161730	161730	485190
		Testing (1/3)		80866	80866	80866	242598
MAWILab-2018	DT / LR	Training (2/3)	0	9906430	9906430	9906430	29719290
		Testing (1/3)	0	4953216	4953216	4953216	14859648
	ELM / MLP	Training (2/3)		8658718	8658718	8658718	25976154
		Testing (1/3)		4329360	4329360	4329360	12988080

and applied five recent datasets that have less of mentioned limitations and meet the real traffic criterions in some level. Table 1 shows a benchmark on mentioned datasets.

In order to evaluate HADM efficiency, five different datasets ISCX-2012, UNSW-NB15 Jan, UNSW-NB15 Feb, ISCX-2017 and MAWILab 2018 with diverse attacks are used. Datasets are classified in four categories: normal, DoS attack, other attacks and unknown.

The ISCX-2012 dataset exhibits realistic network behavior and contain traffic on protocols: HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), Secure SHell (SSH), Internet Message Access Protocol (IMAP), Post Office Protocol version 3 (POP3) and File Transfer Protocol (FTP). The dataset is labeled and includes full packet payload together with diverse intrusion scenarios such as FTP and SSH password brute force, Java based Meterpreter, Superuser, Linux Meterpreter payload and C100Webshel attacks [15].

The UNSW-NB15 (January, February) data set contains 100 GB of raw network traffic with the class label, 49 features and nine different modern attack types. The involved attacks of the UNSW-NB15 data set were categorized into Fuzzers, Analysis, Backdoor, DoS, Exploit, Generic, Reconnaissance, Shellcode and Worms. Analysis category of attack represents different attacks of port scan, spam and penetrations of HyperText Markup Language (HTML) files. Generic category of attack represents cryptographic generic attacks that works against all block-ciphers with a given block and key size, without considering the structure of the block-cipher [16].

The ISCX 2017 dataset consists of 51G network traffic metadata that is labeled including 80 features and full packet payload. The network traffic is provided on protocols, such as HTTP, HTTPS, FTP, SSH and email. The dataset includes the most common attacks based on the 2016 McAfee report, such as Web based, Brute force, DoS,

TABLE 4. Features in datasets.

No.	Feature	No.	Feature
1	Ethernet size	21	UDP length
2	Ethernet destination	22	ICMP type
3	Ethernet source	23	ICMP code
4	IP header length	24	Duration of connection
5	IP type of service	25	Connection starting time
6	IP length	26	IP fragmentation flag
7	IP time to live	27	IP fragmentation overlap
8	IP protocol	28	TCP ACK flag
9	IP source (divided into 4 features)	29	TCP retransmission
13	IP destination (divided into 4 features)	30	TCP push flag
17	TCP source port	31	TCP SYN flag
18	TCP destination port	32	TCP FIN flag
19	UDP source port	33	TCP urgent flag
20	UDP destination port		

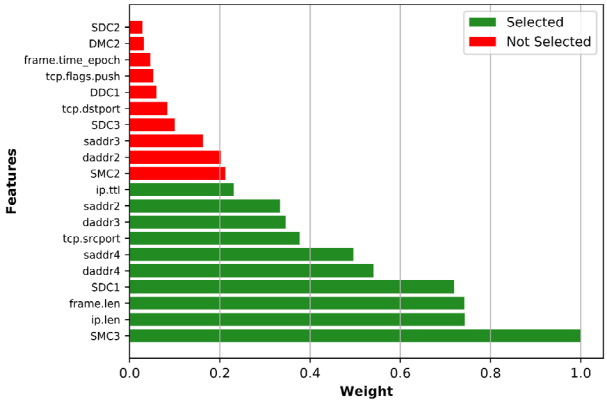


FIGURE 6. Selected features in ISCX-2012 for DT/LR with F-Score.

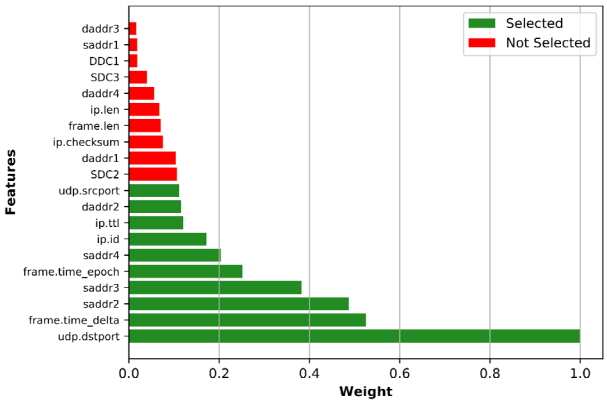


FIGURE 7. Selected features in UNSW-NB15 Jan for k-NN with SVMonline.

DDoS, Infiltration, Heart-bleed, Bot and Scan covered in this dataset [17].

The MAWILab-2018 dataset is captured at a trans-Pacific internet backbone link in Japan. The traffic is captured everyday only for 15 minutes, payload contents are removed and then captured traffic is made available in PCAP format. In addition, the captured data is labeled using several anomaly detection classifiers. The dataset contains Sasser worm,

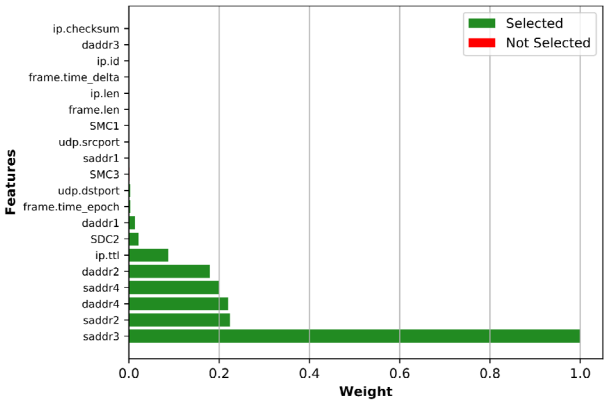


FIGURE 8. Selected features in UNSW-NB15 Jan for SVM with F-Score.

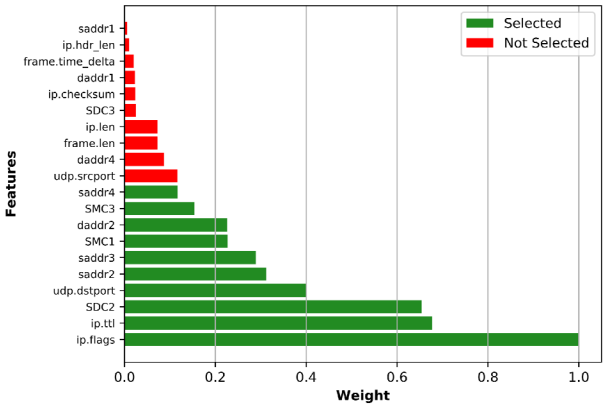


FIGURE 9. Selected features in UNSW-NB15 Feb for k-NN/SVM with Chi2.

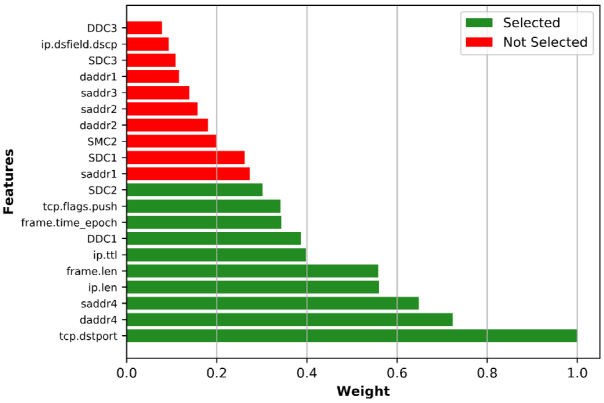


FIGURE 10. Selected features in ISCX-2017 for DT/LR with SVMonline.

Netbios, RPC, SMB, SYN, RST, FIN, Ping flood, FTP, SSH, HTTP and HTTPS attacks. The dataset also labels network scans, port scans and DoS attacks. We have analyzed the traffic for 28th August 2018 [18].

B. DATA PREPROCESSING

The algorithms need data normalization where numeric attributes are transformed into nominal attributes to improve the performance of the algorithms. The IP address and hexadecimal Medium Access Control (MAC) address of the applied datasets are transformed into separate numeric attributes. Each numeric attribute is normalized between

TABLE 5. Selected features for each algorithm.

Dataset	Algorithm	Feature Selection Method	Selected features
ISCX-2012	DT / LR	Chi2	frame.len, ip.len, SMC3, tcp.srcport, saddr2, daddr2, SMC2, tcp.dstport, SDC1, daddr4
		F-Score	SMC3, ip.len, frame.len, SDC1, daddr4, saddr4, tcp.srcport, daddr3, saddr2, ip.ttl
		SVMonline	daddr3, daddr4, SDC1, frame.time_epoch, tcp.dstport, SMC2, DDC1, DDC3, tcp.srcport, saddr2
		RFE	saddr2, DDC3, SMC2, DDC1, DMC2, DMC3, tcp.flags.urg, ip.dsfield.dscp, daddr3, ip.ttl
UNSW-NB15 Jan	k-NN / SVM	Chi2	SDC2, ip.ttl, udp.dstport, saddr2, frame.time_epoch, saddr3, SMC1, daddr2, SMC3, udp.srcport
		F-Score	saddr3, saddr2, daddr4, saddr4, daddr2, ip.ttl, SDC2, daddr1, frame.time_epoch, udp.dstport
		SVMonline	udp.dstport, frame.time_delta, saddr2, saddr3, frame.time_epoch, saddr4, ip.id, ip.ttl, daddr2, udp.srcport
		RFE	daddr2, saddr2, saddr3, udp.dstport, SDC2, frame.time_delta, ip.ttl, daddr4, ip.id, saddr4
UNSW-NB15 Feb	k-NN / SVM	Chi2	ip.flags, ip.ttl, SDC2, udp.dstport, saddr2, saddr3, SMC1, daddr2, SMC3, saddr4
		F-Score	saddr3, ip.ttl, ip.flags, daddr4, saddr2, daddr2, saddr4, SDC2, daddr1, udp.dstport
		SVMonline	udp.dstport, ip.flags, frame.time_delta, saddr2, saddr3, ip.hdr_len, saddr4, daddr2, daddr3, frame.len
		RFE	saddr3, daddr2, udp.dstport, saddr2, ip.flags, daddr3, daddr4, saddr4, frame.time_epoch, SDC2
ISCX-2017	DT / LR	Chi2	DMC2, DDC1, tcp.dstport, tcp.flags.syn, DDC3, saddr3, tcp.srcport, SMC1, SDC1, DMC1
		F-Score	saddr3, tcp.srcport, DDC3, tcp.dstport, DMC3, DMC2, DDC1, frame.len, ip.len, saddr4
		SVMonline	tcp.dstport, daddr4, saddr4, ip.len, frame.len, ip.ttl, DDC1, frame.time_epoch, tcp.flags.push, SDC2
		RFE	tcp.dstport, ip.len, daddr4, frame.time_delta, ip.dsfield.dscp, saddr3, saddr4, saddr1, ip.flags, DDC1
MAWILab-2018	DT / LR	Chi2	tcp.flags.syn, tcp.srcport, saddr1, SMC3, SDC1, SDC3, DMC2, DDC2, SMC1, saddr2
		F-Score	saddr1, saddr2, ip.len, frame.len, tcp.srcport, tcp.dstport, saddr3, tcp.flags.syn, tcp.flags.ack, ip.flags
		SVMonline	ip.dsfield.dscp, ip.len, frame.len, daddr3, daddr4, saddr3, tcp.srcport, daddr2, saddr1, ip.flags
		RFE	ip.dsfield.dscp, ip.len, frame.len, frame.time_delta, saddr1, daddr4, daddr3, saddr2, SDC3, tcp.srcport

0 and 1 by calculating batch mean and standard deviation, unless there is an already defined range (e.g., IP address range).

Though we have also trained and tested our model with small batches of data, where each batch has same amount of packets from each dataset, our aim was applying the subset of entire data in order to verify the model scalability. Therefore, all the analysis in this paper refers to training and testing with the entire data.

Distribution of packets in datasets is shown in Table 2; whereas distribution of packets for testing and training is shown in Table 3.

While, the 2/3 of data is used for training the algorithms, the 1/3 is used for testing. In addition, to solve the class imbalance problem, down sampling is applied for effective class distribution of data to train and test the algorithms. The class imbalance problem is a quite common issue with real life traffic and causes performance degradation of conventional machine learning algorithms.

For UNSW-NB15 Jan and UNSW-NB15 Feb dataset, it was realized that "DoS attacks" only comprise 0.15% (3846/2472824) and "Other attacks" is 1.8% (44956/2472824) of the total UDP packets. Since the cost of miss-predicting "DoS attacks" and "Other attacks" is equally important, to solve this problem, the cost is kept the same but two different sampling strategies were implemented: under-sampling and over-sampling. Under-sampling randomly eliminates some data from majority classes, whereas over-sampling adds duplicated or artificially generated samples to the minority classes [19].

TABLE 6. Learning algorithm I performance evaluation.

Dataset	Algorithm	Accuracy score	Cross-entropy loss	FN score	Testing time (s)
ISCX-2012	ELM10	0.53051	0.96743	0.51352	0.08776
	ELM50	0.58761	0.90109	0.51352	0.14861
	MLP10	0.86024	0.55731	0.14431	0.02163
	MLP50	0.87229	0.58157	0.14590	0.06401
UNSW-NB15 Jan	ELM10	0.99452	0.55983	0	0.01031
	ELM50	0.99452	0.56187	0	0.00842
	MLP10	0.99452	0.03504	0	0.00099
	MLP50	0.99452	0.03392	0	0.00302
UNSW-NB15 Feb	ELM10	0.95639	0.61755	0	0.03299
	ELM50	0.95639	0.61819	0	0.04620
	MLP10	0.95521	0.17596	0	0.00251
	MLP50	0.95628	0.17583	0	0.00884
ISCX-2017	ELM10	0.70817	0.90051	0.10640	0.57638
	ELM50	0.76518	0.80292	0.10640	1.02763
	MLP10	0.78525	0.44047	0.10640	0.09491
	MLP50	0.80387	0.41068	0.06230	0.12003
MAWILab-2018	ELM10	0.66998	0.89468	0.32096	23.2952
	ELM50	0.81291	0.78605	0.14894	43.9939
	MLP10	0.88745	0.32922	0.10091	4.56287
	MLP50	0.94309	0.29415	0.08888	15.9275

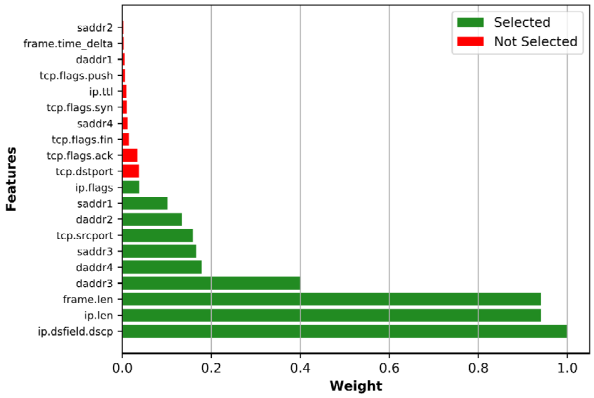


FIGURE 11. Selected features in MAWILab-2018 for DT with SVMonline.

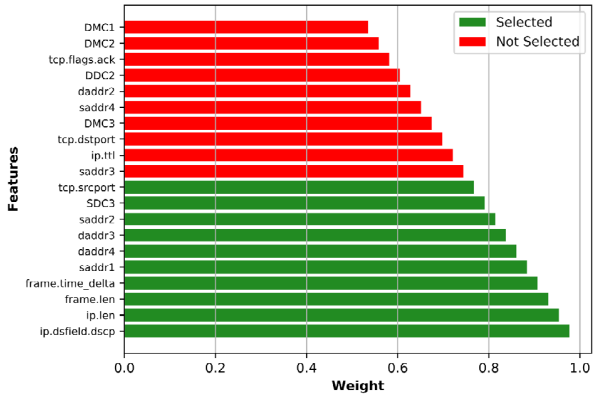


FIGURE 12. Selected features in MAWILab-2018 for LR with RFE.

For under-sampling, a subset of normal, unknown and other type of attacks' classes were randomly and independently selected. This method reduces the number of samples in each of the classes and combines the subsets that contain the DoS attack as a new dataset. For over-sampling, random samples from the DoS attacks were duplicated and added to the new dataset.

Training and testing time in the experimental results support that over-sampling has longer training time and can lead to over-fitting. On the other hand, under-sampling provides better DoS attack prediction than over-sampling, as mentioned in the paper [19]. We had tested on smaller dataset and our findings also concluded that under-sampling performs better DoS attack prediction than over-sampling. Therefore, we applied under-sampling to the datasets.

V. EXPERIMENTAL RESULTS

All the experiments are carried out on a server with Intel®Xeon® 2 x Gold 6130 CPU @2.1GHz (16 cores in each processor), 125 GB RAM, 1.6 TB HDD. The scripts were developed in Python in a Linux environment (Ubuntu 17.10) and utilized scikit-learn library [20]. All applied algorithms in the evaluation process of HADM are trained once and saved for the future tests. Currently the platform is tested on a single workstation, but the whole functionality can be installed on several Virtual Machines (VMs) for load balancing and decentralized monitoring purposes in order to handle large amount of traffic at a core network. The mechanism has been explained in other papers presented by authors on SDN security [21].

The proposed approach is tested, and performance is evaluated with combination of below algorithms and feature selection methods:

1- ELM10, ELM50, MLP10 and MLP50.
2- k-NN with Chi2, RFE, SVMonline and F-Score.
3- SVM with Chi2, RFE, SVMonline and F-Score.
4- DT with Chi2, RFE, SVMonline and F-Score.
5- LR with Chi2, RFE, SVMonline and F-Score.

TABLE 7. Linear algorithm I performance evaluation.

Dataset	Algorithm	FS method	FN score	Precision	Recall	F1 Score	Testing time (s)
UNSW-NB15 Jan	SVM	Chi2	0.991967871	0.2000	0.00803	0.015444018	0.00061
		F-Score	0	0.5000	1	0.666666667	0.00056
		SVMonline	0.136546185	0.85317	0.86345	0.858283434	0.00056
		RFE	0.084337349	0.78621	0.91566	0.846011133	0.00056
	k-NN	Chi2	0.120481928	0.96053	0.87952	0.918238994	0.00623
		F-Score	0.108433735	0.94872	0.89157	0.919254662	0.00570
		SVMonline	0.092369478	0.93004	0.90763	0.918699185	0.00877
		RFE	0.156626506	0.94595	0.84337	0.891719745	0.00470
UNSW-NB15 Feb	SVM	Chi2	0.039651838	0.60549	0.96035	0.74270755	0.00081
		F-Score	0.091876209	0.94183	0.90812	0.924667653	0.00074
		SVMonline	0.076402321	0.88100	0.92360	0.901794146	0.00071
		RFE	0.097678917	0.94721	0.90232	0.924219908	0.00090
	k-NN	Chi2	0.032882012	0.74683	0.96712	0.842815003	0.06114
		F-Score	0.044487427	0.87202	0.95551	0.911859711	0.04990
		SVMonline	0.062862669	0.95187	0.93714	0.944444442	0.04841
		RFE	0.091876209	0.94277	0.90812	0.92512315	0.03443

TABLE 8. Linear algorithm II performance evaluation.

Dataset	Algorithm	FS method	FN score	Precision	Recall	F1 Score	Testing time (s)
ISCX-2012	DT	Chi2	0.032054299	0.97104	0.96795	0.969491423	0.01783
		F-Score	0.005405652	0.88277	0.99460	0.93535443	0.01167
		SVMonline	0.015229968	0.96026	0.98477	0.972360283	0.01282
		RFE	0.072885191	0.91735	0.92711	0.922206698	0.01417
	LR	Chi2	0.082421003	0.82100	0.91758	0.866607872	0.03399
		F-Score	0.063577144	0.88415	0.93642	0.909533363	0.03408
		SVMonline	0.067054375	0.89881	0.93295	0.915560854	0.03416
		RFE	0.067008822	0.88434	0.93299	0.90801481	0.03398
ISCX-2017	DT	Chi2	0.031274898	0.99361	0.96873	0.981012113	0.02499
		F-Score	0.031274898	0.99361	0.96873	0.981012113	0.02470
		SVMonline	0.022266368	0.99964	0.97773	0.988566243	0.02485
		RFE	0.022361977	0.99978	0.97764	0.988586373	0.02921
	LR	Chi2	0.426959727	0.84096	0.57304	0.681619448	0.05647
		F-Score	0.420320185	0.73730	0.57968	0.649058531	0.04722
		SVMonline	0.208470993	0.90802	0.79153	0.845780125	0.04623
		RFE	0.434353521	0.91362	0.56565	0.698706803	0.04710
MAWILab-2018	DT	Chi2	0.180783959	0.88037	0.81922	0.848691847	1.91455
		F-Score	0.077401228	0.66860	0.92260	0.775328147	1.90881
		SVMonline	0.052822045	0.70316	0.94718	0.807131788	1.89239
		RFE	0.130722747	0.75884	0.86928	0.810313293	1.37810
	LR	Chi2	0.213921622	0.61454	0.78608	0.689802611	1.32465
		F-Score	0.249397967	0.64651	0.75060	0.694677142	1.35961
		SVMonline	0.261111771	0.70346	0.73889	0.720740275	2.13941
		RFE	0.150309011	0.66467	0.84969	0.745877048	1.60010

Since in previous paper [22], authors already compared HADM efficiency with a scenario where protocol analyzer has not been used (algorithms were used standalone for attack detection) and revealed that protocol analyzer scenario performed better in terms of detection. Therefore, this paper only concentrates on the scenario where protocol analyzer is used. The traffic carried by a vulnerable protocol is directed to k-NN/SVM, DT/LR and the rest of the traffic to ELM/MLP. All UDP traffic is forwarded to k-NN/SVM algorithm and all TCP traffic to DT/LR algorithm.

There are not many variations on features related to protocol, for learning algorithm I in protocol analyzer module, therefore, we considered 9 fixed features including source IP address (saddr1, saddr2, saddr3, saddr4), destination IP address (daddr1, daddr2, daddr3, daddr4) and time to live (ip.ttl). The feature selection methods mentioned in Section II, have been applied on linear algorithm I and II and the best combinations for both feature selection methods and algorithms have been selected based on the achieved efficiency. The original dataset contains 33 features as shown in Table 4.

Out of 33 features, each feature selection method gives us 10 selected features. As it is shown in Table 5, all algorithms (SVM, k-NN, DT and LR) are tested with four feature selection methods and then based on the achieved best performance, one feature selection method is selected for each

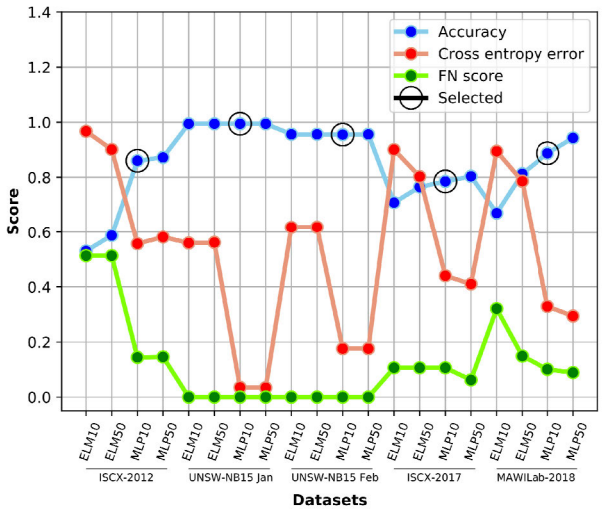


FIGURE 13. Performance evaluation of learning algorithm I.

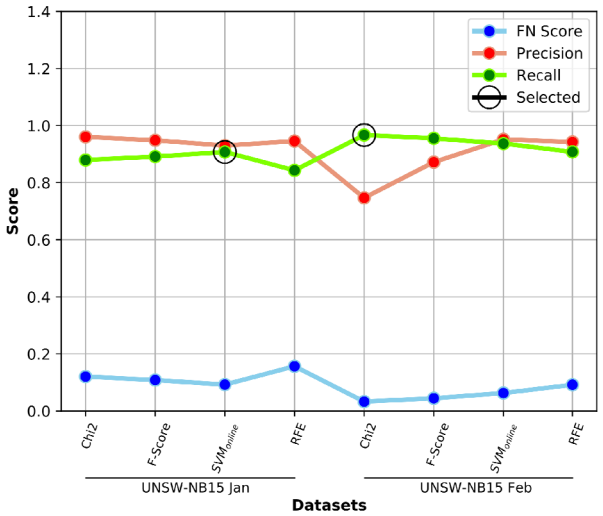


FIGURE 14. Performance evaluation of k-NN.

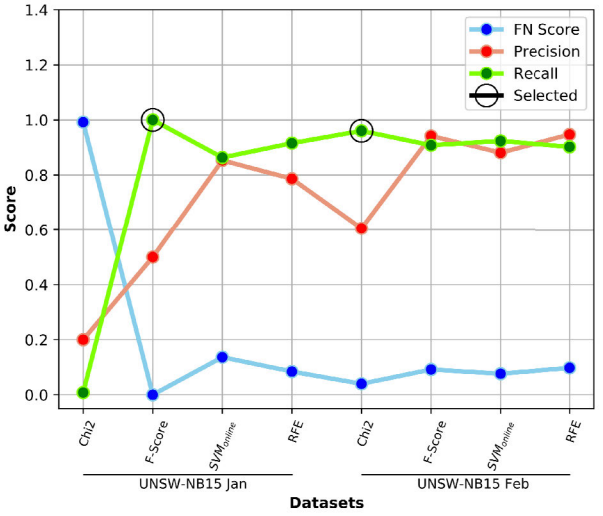


FIGURE 15. Performance evaluation of SVM.

algorithm. As it is shown in Fig. 6-12, the proposed approach has selected 10 features out of 33 features for each algorithm that means 69.69% dimensionality reduction. This reduction will be beneficial in situations with a greater number of features such as extracting payload features.

Presented results in Table 6 and Fig. 13, for testing time, total accuracy score, binary cross-entropy loss and false negative score shows MLP algorithm outperforms the ELM algorithm. The binary cross-entropy loss and the false negative score of ELM is quite high, which means that in most cases, it fails to give alarm when an intrusion occurs. If MLP with 10 and 50 hidden layer neurons are compared, it can be seen that MLP with 10 hidden layer neurons performs slightly better than the other algorithms in terms of differentiating the normal and attack traffic, also the testing time is smaller with MLP with 10 hidden layer neurons. Therefore, considering the fact the overall architecture has a high time and model complexity with many pre-processing (protocol analyzer) and post-processing (learning algorithm II) steps, we have chosen MLP with 10 hidden layer neurons for this module to reduce the overall processing time for labeling an incoming network packet and the model complexity.

Table 6 shows that every learning algorithm tested on UNSW-NB15 dataset gives the exact same performance. When checked in detail, it was found that differentiating attack traffic from normal traffic is quite simple. Therefore, even ELM10 can reach the highest performance possible. However, the cross-entropy loss of ELM is still higher than MLP algorithms. This can be explained by checking the confidence values of ELM and MLP. We observed that, even the accuracy of ELM and MLP is the same, the confidence (the probability of the predicted class) of ELM is generally lower than MLP. For example, a packet can be correctly identified as an attack by both MLP and ELM methods, but ELM gives a probability of 70%, whereas MLP gives a probability of 90%. Therefore, cross-entropy loss of ELM might be higher than MLP methods even if their accuracy is the same.

As discussed, attacks in UNSW-NB15 dataset have a distinct source and destination IP address that differentiate it clearly from normal and unknown classes. Since, we take fixed features that are majorly source and destination IP address, this causes 0 FN score. UNSW-NB15 has been generated in laboratory environment and this explains the reason for only one IP address for all attacks. To tackle this issue, we utilized 4 feature selection methods (Chi2, F-Score, RFE and SVMonline) to extract 10 best features from UNSW-NB15 datasets and then applied ELM and MLP algorithms. However, still feature selection methods selected majorly source and destination IP addresses as best features; and as a result, again attacks were not misclassified, and FN score was again 0 which confirms UNSW-NB15 dataset is not diverse from the features perspective. The performance evaluation of each algorithm can be seen in the Fig. 13. Table 7 and Table 8 also show the performance evaluation for testing HADM model based on five metrics, FN score, precision, recall, F1 score and testing time. The precision

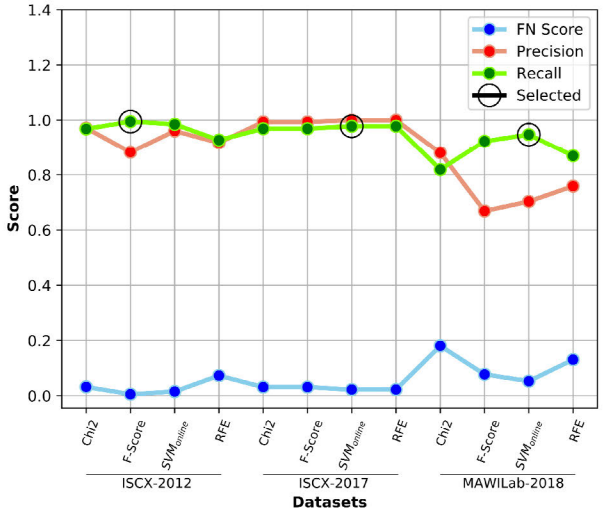


FIGURE 16. Performance evaluation of DT.

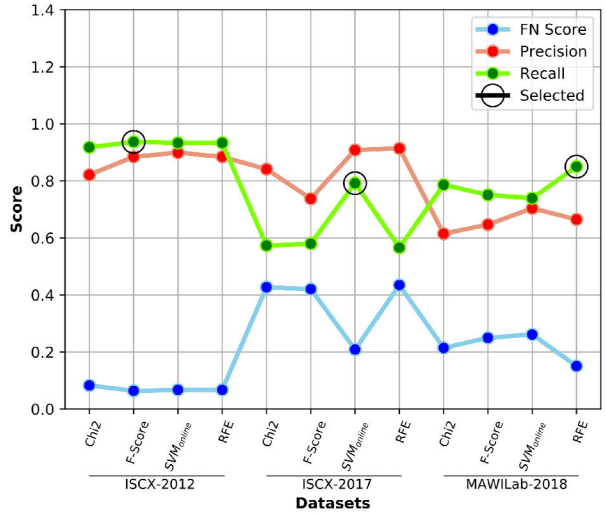


FIGURE 17. Performance evaluation of LR.

and recall values are measured for DoS and other attack classes.

As it is shown in both tables, for UDP DoS detection, though the detection rate in some results is a bit higher for k-NN still SVM algorithm is selected considering lower computation time. And for other attacks, the best performance is achieved with DT algorithm. It appears from the results that HADM did not have tremendous increase in computation time neither considerable decrease in detection factors while various datasets with different size and diverse attacks have been used. This means that the proposed model is scalable and robust.

Fig. 13-17 show the performance evaluation of each algorithm for each dataset, applying the best feature selection methods. The selected points in Fig. 14-17 (linear algorithm I and II), are considered based on the best recall achieved since the aim is to detect the majority of attack packets.

VI. CONCLUSION AND FUTURE WORK

In previous paper [22] Hybrid Anomaly Detection Model (HADM) was proposed as an intelligent platform and its efficiency was evaluated against available methods and algorithms to detect network traffic intrusion. The proposed model compromises of two main parts where each part independently increases the efficiency of attack detection based on the metrics such as precision, recall, accuracy and computation time. Overall, the proposed model utilizes the protocol analyzer and a combination of learning and linear algorithms for network traffic filtering, reducing the processing time and increasing the detection rate.

In this paper, various feature selection methods have been applied together with several algorithms to achieve the highest efficiency. Even though it has been a challenge to find reliable and publicly available datasets, to measure the model robustness and scalability over the previous study, model has been tested with various datasets. For this purpose, 16 datasets that were publicly available starting from 1998 are introduced and compared. Majority of mentioned datasets are small, and they do not have attack or traffic diversity and are usually anonymized. Therefore, the five recent datasets that have less of mentioned limitations and meet the real traffic criterions were selected for testing. From the experimental results it can be concluded that the Support Vector Machine (SVM) algorithm together with SVMonline feature selection improves User Datagram Protocol (UDP) Denial of Service (DoS) detection accuracy along with reduced computation time. Similarly, Decision Tree (DT) algorithm with SVMonline feature selection method gives higher efficiency for other attacks. It appears from the results that HADM did not have tremendous increase in computation time neither considerable decrease in detection factors while various datasets with different size and diverse attacks have been used. This shows that the proposed model is scalable and robust.

The future work of this paper would concentrate on second part of the model and is ongoing. The study will apply a dynamic feature weight selection method together with a deep learning algorithm to dynamically label and cluster unknown and known attacks.

REFERENCES

[1] K. S. Desale and R. Ade, "Genetic algorithm based feature selection approach for effective intrusion detection system," in *Proc. Int. Conf. Comput. Commun. Inform. (ICCCI)*, Coimbatore, India, Jan. 2015, pp. 1–6.

[2] M. Monshizadeh and Z. Yan, "Security related data mining," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.*, Xi'an, China, Sep. 2014, pp. 775–782.

[3] A. Di Pietro, J.-P. Vasseur, and J. C. Mota, "Dynamic deep packet inspection for anomaly detection," U.S. Patent 2017 099 310 (A1), Apr. 6, 2017.

[4] J. Vasseur, F. Flacher, and G. Mermoud, "Anomaly detection in a network coupling state information with machine learning outputs," U.S. Patent 2017 0 104 774 (A1), Apr. 13, 2017.

[5] A. D. Pietro, J.-P. Vasseur, and J. C. Mota, "Signature creation for unknown attacks," U.S. Patent 2016 0 028 750 (A1), Jan. 28, 2016.

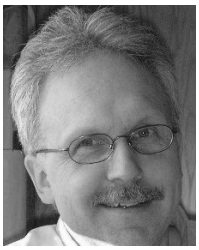
[6] N. Yadav, E. Scheib, and R. Agasthy, "Network behavior data collection and analytics for anomaly detection," U.S. Patent 2016 0 359 695 (A1), Dec. 8, 2016.

[7] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3369–3388, 4th Quart., 2018.
[8] B. G. Atli, "Anomaly-based intrusion detection by modeling probability distributions of flow characteristics," Dept. Comput. Sci., Aalto Univ., Espoo, Finland, Tech. Rep., 2017.
[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[10] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.
[11] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, 2016.
[12] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Lang. Eng.*, vol. 16, no. 1, pp. 100–103, 2010.
[13] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.
[14] P. Laskov, C. Gehl, S. Krüger, and K. Müller, "Incremental support vector learning: Analysis, implementation and applications," *J. Mach. Learn. Res.*, vol. 7, pp. 1909–1936, Sep. 2006.
[15] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.
[16] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, 2018, pp. 108–116.
[18] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. ACM CoNEXT*, Philadelphia, PA, USA, 2010, Art. no. 8.
[19] M. M. Rahman and D. N. Davis, "Addressing the class imbalance problem in medical datasets," *Int. J. Mach. Learn. Comput.*, vol. 3, no. 2, pp. 224–228, Apr. 2011.
[20] *Scikit-Learn Machine Learning in Python*. Accessed: Jan. 22, 2019. [Online]. Available: https://scikit-learn.org/stable/
[21] M. Monshizadeh, V. Khatri, and R. Kantola, "Detection as a service: An SDN application," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, Bongpyeong, South Korea, Feb. 2017, pp. 285–290.
[22] M. Monshizadeh, V. Khatri, B. Atli, and R. Kantola, "An intelligent defense and filtration platform for network traffic," in *Proc. Wired/Wireless Internet Communications* (Lecture Notes in Computer Science), vol. 10866, K. Chowdhury, M. D. Felice, I. Matta, and B. Sheng, Eds. Boston, MA, USA: Springer, 2018, pp. 107–118.

MEHRNOOSH MONSHIZADEH is currently pursuing the Ph.D. degree with the Electrical School, Aalto University, Finland. She is currently a Security Research Specialist with Nokia Bell Labs. Her research interests include cloud security, mobile network security, the IoT security, and data analytics.

VIKRAMAJEET KHATRI received the M.Sc. degree in information technology from the Tampere University of Technology, Finland. He is currently a Security Specialist with Nokia Bell Labs. His research interests include intrusion detection, malware detection, the IoT security, and cloud security.

BUSE GUL ATLI received the M.Sc. degree in signal, speech and language processing from Aalto University, where she is currently pursuing the Ph.D. degree with the Secure Systems Group at Computer Science. Her research interests include adversarial machine learning and security of artificial intelligence.



RAIMO KANTOLA received the D.Tech. degree in computer science from the Helsinki University of Technology, Finland. He is currently a Professor in networking technology with the Department of Comnet, Aalto University, Finland. His research interests include SDN, customer edge switching, trust in networks, and cloud security.

ZHENG YAN received the D.Sc. degree in technology in electrical engineering from the Helsinki University of Technology. She is currently a Professor with the Xidian University, and also a Visiting Professor with Aalto University. Her research interests are in trust, security, privacy, and security-related data analytics. She serves as a General or Program Chair for over 30 international conferences and workshops. She is a Steering Committee Co-Chair of the IEEE Blockchain International Conference. She is also an Associate/Area Editor of many reputable journals, including the IEEE Internet of Things Journal, the *Information Sciences*, the *Information Fusion*, JNCA, the IEEE Access, SCN, and so on.