Data Randomization and Cluster-Based Partitioning for Botnet Intrusion Detection

Omar Y. Al-Jarrah, Omar Alhussein, Paul D. Yoo, *Senior Member, IEEE*, Sami Muhaidat, *Senior Member, IEEE*, Kamal Taha, *Senior Member, IEEE*, and Kwangjo Kim, *Member, IEEE*

***Abstract*—Botnets, which consist of remotely controlled compromised machines called bots, provide a distributed platform for several threats against cyber world entities and enterprises. Intrusion detection system (IDS) provides an efficient countermeasure against botnets. It continually monitors and analyzes network traffic for potential vulnerabilities and possible existence of active attacks. A payload-inspection-based IDS (PI-IDS) identifies active intrusion attempts by inspecting transmission control protocol and user datagram protocol packet's payload and comparing it with previously seen attacks signatures. However, the PI-IDS abilities to detect intrusions might be incapacitated by packet encryption. Traffic-based IDS (T-IDS) alleviates the shortcomings of PI-IDS, as it does not inspect packet payload; however, it analyzes packet header to identify intrusions. As the network's traffic grows rapidly, not only the detection-rate is critical, but also the efficiency and the scalability of IDS become more significant. In this paper, we propose a state-of-the-art T-IDS built on a novel randomized data partitioned learning model (RDPLM), relying on a compact network feature set and feature selection techniques, simplified subspacing and a multiple randomized meta-learning technique. The proposed model has achieved 99.984% accuracy and 21.38 s training time on a well-known benchmark botnet dataset. Experiment results demonstrate that the proposed methodology outperforms other well-known machine-learning models used in the same detection task, namely, sequential minimal optimization, deep neural network, C4.5, reduced error pruning tree, and randomTree.**

***Index Terms*—Botnet intrusion detection, efficient learning, ensembles, feature selection, machine-learning (ML).**

Manuscript received November 13, 2014; revised May 8, 2015 and July 3, 2015; accepted October 2, 2015. Date of publication October 30, 2015; date of current version July 15, 2016. This work was supported in part by the Khalifa University of Science, Technology and Research-Korea Institute of Science and Technology (KAIST) Institute, in part by the KAIST, Korea, and in part by the National Research Foundation of Korea through the Korea government (MSIP) under Grant NRF-2015R1A2A2A01006812. This paper was recommended by Associate Editor L. D. Xu.

O. Y. Al-Jarrah and K. Taha are with the Electrical and Computer Engineering Department, Khalifa University of Science, Technology and Research, Abu Dhabi, UAE (e-mail: omar.aljarrah@kustar.ac.ae; kamal.taha@kustar.ac.ae).

O. Alhussein is with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A1S6, Canada (e-mail: oalhusse@sfu.ca).

P. D. Yoo is with the Department of Computing and Informatics, Bournemouth University, Poole, U.K. (e-mail: paul.d.yoo@ieee.org).

S. Muhaidat is with the Electrical and Computer Engineering Department, Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, and also with the University of Surrey, Guldford GU2 7XH, U.K. (e-mail: muhaidat@ieee.org).

K. Kim is with the School of Computing, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Korea (e-mail: kkj@kaist.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2490802

I. Introduction

NOWADAYS, despite the use of different security measures, such as firewalls and encryption algorithms, many organizations and enterprises fall victims of different cyber-attacks. Attackers might launch different types of attacks, which vary on their effect, over the organizations and enterprises, from low to high significance. Attackers may deliberately exploit systems vulnerabilities in order to sneak into the system and ultimately control it.

Botnet attack is known as one of the most severe cyber-threats, where a collection of computers are compromised and remotely controlled by a controlling computer called botmaster [1]–[3]. Botnet often involves thousands of bots that are spread over the Internet. Botmaster utilizes botnet by keeping it under its control as long as possible. Botmaster might use botnet for different purposes, such as launching distributed cyber-attacks, e.g., distributed denial of service attack, and performing distributed computation tasks. Intrusion detection system (IDS) is considered as a reactive measure that enhances existing network security by detecting, identifying, and tracking the intruders [4]. Many existing IDSs built for botnets are rule-based, whose performances depend upon rule-sets defined by experts [5], [6]. A rule-based botnet IDS identifies botnets by examining network traffic and comparing it with a known or previously seen botnet signatures, which normally encoded by security expert [7]. However, due to the substantial increase in network traffic, keeping these rules updated becomes more and more difficult, tedious, and time-consuming [8]. Such dependency makes it inefficient toward novel forms of botnets [9].

Machine-learning (ML) techniques could be used to automate botnet detection process. Knowledge-based intrusion detection by ML techniques, where a learning system build a model from previously known attack signatures, has recently been an area of concentration in ML society [10]. The adaptability, flexibility, and automated-learning ability of ML models show significant advantages over rule-based IDSs. However, the majority of ML models are performance-driven, meaning that most nonparametric and model-free properties of ML-based approaches require high computational cost to find an optimal model (i.e., the global optima). Thus, designing a more accurate ML-based intrusion detection model will hence lead to a higher likelihood of slow model update, and thus, preventing real-time detection, and efficient use of energy, due to the increasing computational cost.

An ideal IDS shall provide a full protection to users and networks with zero type II error rate [i.e., false-negative (FN)], which refers to a failure of an IDS to detect an actual attack [11], and adapt to the frequently changing network environments [8], [12]. With the recent emergence of large-scale digitized network-traffic data (i.e., big data), the efficiency and scalability of ML-based IDS is of critical importance [13]. Not only the model accuracy and the detection-rate (DR) must be considered when evaluating the performance of an IDS, but also the computational complexity and efficiency.

In this paper, we examine network flow within a time interval and extract feature sets that characterize the network flow within that time interval. Feature sets are then used to distinguish between botnet attack and normal network traffic.

This paper presents the design and development of botnet detection methodology and solution to deal with large volumes of network data. The contributions of this paper include.

1) Design of novel algorithms that detect botnet attacks based on network-traffic flow characteristics at the transport layer level, regardless of the underlying layers contents or payload. The proposed model outperforms payload-inspection approach in terms of computational time and immunity to encryption algorithms [14].
2) A novel data condensation method design and development. Toward this, we propose the following.
    a) A modified forward selection ranking technique to eliminate redundant and irrelevant features.
    b) A data reduction technique that utilizes Voronoi-based data partitioning to deal with the large-scale botnet dataset.
3) Evaluation and comparisons of the proposed method. We compare the proposed methods with several popular ML models on well-known Information Security and Object Technology (ISOT) botnet dataset theoretically and experimentally, and they are as follows. ML algorithms: multilayered deep neural network (DNN), reduced error pruning tree (REPTree), randomTree (RTree), C4.5 decision tree (DT), and sequential minimal optimization (SMO). Feature selection algorithms such as correlation feature selection (CFS) and WrapperSubsetEval. These are evaluated on Botnet benchmark-11Fs dataset and their performance are measured by seven different standard metrics.

The organization of the paper is as follows. Section II gives the theoretical background of botnet and botnet detection techniques. Section III proposes novel randomized data partitioned learning model (RDPLM). In Section IV, the performance of the proposed model is compared with other existing detection models in terms of detection accuracy (Acc), detection rate (DR), false alarm rate (FAR), Mathew's correlation coefficient (Mcc), standard deviation of F-measure (σ), time to build model (TBM), and testing time (TT).

II. Botnet Detection Techniques

Generally, botnet detection techniques can be classified into four categories: 1) rule-based; 2) anomaly-based; 3) domain name server (DNS)-based; and 4) data-mining-based [15]. Snort, which is used by BotHunter [16], is a well-known open source rule-based IDS [6]. Such method inspects the payload of the transmission control protocol (TCP) or user datagram protocol packets and compare it with known intrusion signatures. Although rule-based methods could be accurate and efficient in identifying malicious activities, as the volume of data grows, encoding attack's rules becomes tedious and time-consuming.

Anomaly-based methods define anomaly behavior as a deviation from normal network behavior. Unlike rule-based, it can detect novel attacks. However, this method requires modeling of pure normal network profile, which is not always easy to achieve. Karasaridis *et al.* [17] proposed a novel methodology, which employs scalable nonintrusive algorithm that analyzes huge amount of summary traffic data of transport layer data to detect, track, and characterize botnet on a large Tier-1 ISP network. Gu *et al.* [18] proposed an algorithm called Botsniffer based on the observation that bots, within the same botnet, show a very strong synchronization and spatial-temporal correlations in their responses and activities. Arshad *et al.* [19] proposed a fully anomaly-based approach that requires no *a priori* knowledge of bot signatures, command and control (C&C) servers' addresses or botnet C&C protocols. This method clusters bots with similar net flows and attacks in different time windows and performs correlation to identify infected hosts successfully.

DNS-based methods are built on the fact that bots need to establish a connection with C&C server in order to join the botnet channel and interact with botmaster or other bots. This method is similar to anomaly-based, in a sense that it detects botnet by monitoring the network traffic on DNS server side and comparing it with normal network profile [20]. Villamarin-Salmon and Brustoloni [21] developed botnet intrusion detection model that analyzes and correlates anomalous DDNS. Choi *et al.*'s [22] system monitors group activities in DNS traffic.

A number of recent approaches utilize data-mining or ML techniques have gained popularity. Model-free property of ML algorithms seems to be suitable for rapidly changing environments and brings a few important advantages to botnet intrusion detection tasks, such as model adaptability, learnability, and flexibility [14]. Han and Cho [23] proposed a novel intrusion-detection technique based on evolutionary neural networks. The proposed technique discovers the structures and weights of the neural networks, simultaneously. Strayer *et al.* [24] examined flow characteristics, such as bandwidth, packet timing, and burst duration, for evidence of Internet-relay-chat-based (IRC) C&C activities, and successfully distinguished IRC flow by eliminating traffics that are likely to be normal IRC. Then, they classified the remaining traffic into groups that is likely to be a part of botnet. Masud *et al.* [25] proposed a flow-based botnet detection system, which does not impose any restriction on botnet communication protocol. Hence, it could be used for detecting non-IRC botnets. Saad *et al.* [26] proposed a P2P botnet detection system using traffic analysis. The system considers both flow-based features and host-based features. They tested five different ML algorithms including nearest neighbor (NN), naive

Bayesian, support vector machine (SVM), artificial neural networks, and Gaussian-based classifier. Their proposed method utilized with a SVM achieved a detection rate of about 97%.

Rasheed and Alhajj [27] presented a time efficient suffix tree-based algorithm to detect the periodicity of outlier patterns in a time series by giving more significance to less frequent yet periodic patterns. Bilge *et al.* [28] also used a ML-based method to classify wide-scale NetFlow traces as normal or malicious. Their method uses the NetFlow traces that are usually sampled over the observed traffic, which might causes losses of information about fine pattern of malicious traffic. Among four different ML algorithms, namely, SVM, C4.5, DT, and RandomForest (RF), RF obtained the best ratio between DR and false positive rate. In the study of Stevanovic and Pedersen [29], how much traffic, that is needed to be observed per flow in order to capture the pattern of malicious traffic, is evaluated. Their proposed method successfully detected botnet traffics using purely flow-based traffic analysis. Most botnet IDSs reported in the literature are payload-inspection-based that captures and analyzes header and payload for every packet in network traffic, which is computationally demanding [30]. Moreover, such method could be very inefficient when packets are encrypted. Traffic-based botnet IDS examines the complete traffic between two hosts during the botnet life time [14]. Thus, it is not preferable when considering real-time botnet detection since hosts might run and communicate for few seconds or days.

With the explosive growth of distributed and heterogeneous network data, not only the accuracy, but also model efficiency and scalability have become a matter of great interest among cyber-security experts. Distributing workload among multiple workstations/computing clusters are of great interest as well. However, existing ML algorithms are not scalable and flexible enough to deal with such massive amount of network data.

III. Methodologies

Our methodologies consist of four consecutive phases. First phase involves collecting and generating botnet dataset that represents real-world traffic, to evaluate the performances of our proposed method. Second, to develop an efficient and scalable ML algorithm that can deal with the large-scale network traffic and provide real-time detection within acceptable time-margin, selecting right features is of critical importance. In this phase, we develop a novel feature selection technique that eliminates redundant and irrelevant data and creates a subset of relevant features. Third, in addition to above, we introduce novel data reduction technique that can significantly reduce the number of data samples used during learning process. Fourth, meta-learning model of multiple randomized trees, which consider a randomly selected features subset with no pruning, is constructed to detect botnet attacks. The predictive performance of the proposed methods are compared against those of SMO algorithm for training a support vector classifier, multilayered perceptron DNN, C4.5 pruned DT, REPTree, and RTree for accuracy, DR, false alarm rate, Mcc, F-measure, which represents model stability and generalization ability, time to build model, and TT.

TABLE I
Features Description

No	Name	Description	Type
1	SrcPort	Flow source port.	Disc.
2	DstPort	Flow destination port.	Disc.
3	Proto	Transport layer protocol or mixed.	Disc.
4	APL	Average payload packet length for time interval.	Cont.
5	PV	Variance of payload packet length for time interval.	Cont.
6	PX	# of packets exchanged for time interval.	Disc.
7	PPS	# of packets exchanged per second in time interval T.	Cont.
8	FPS	The size of the first packet in the flow.	Disc.
9	TBP	The average time between packets in time interval.	Cont.
10	NR	# of reconnects for a flow.	Disc.
11	FPH	# of flows from this address over the total number of flows generated per hour.	Cont.

The final dataset consists of 11 features selected from the original 41 features in interval of 300s, based on the behavior on well-known protocols as well as well-known botnet behavior [14].

A. Data Collection: Botnet Benchmark-11Fs Dataset

Building a ML-based botnet IDS requires a dataset, where the model can learn from. The dataset shall also represent real networks traffic by having all the necessary features. In our experiments, the selected models are evaluated on the well-referenced ISOT benchmark dataset [26]. ISOT contains both malicious P2P botnet and nonmalicious traffic. It has the traffics of two notorious P2P botnets from 2007 to 2009, namely Waledac and Storm. ISOT dataset obtained its malicious traffic from the French Chapter of Honeynet Project [31]. The nonmalicious traffic was collected jointly by the TrafficLab at Ericsson Research in Hungary [32] and Lawrence Berkeley National Laboratory (LBNL) in 2005. These two nonmalicious datasets have a good number of everyday traffic from variety of applications, including HTTP Web browsing, gaming, and bit torrent clients, as well as traffic from mid-sized enterprises, which is provided by LBNL.

To generate an experimental dataset that represents real world traffic, these three datasets have been merged into single trace file by mapping the IP addresses of the infected machines to two of the machines that provide background traffic. TCPReplay tool that replays trace files on the network interface device is then used to replay all the trace files in order to homogenize the network behavior of the three datasets. Finally, the replayed data is captured via Wireshark.

In this paper, we examine network traffic characteristics that define network flow, a collection of network packets that are exchanged between two IP addresses within time period T using a pair of ports utilizing different layer 4 protocols [33]. The selection of the length of T is a tradeoff between accuracy and speed that could influence the utility and usefulness of the detection system. Our ultimate aim is to detect any intrusions as quickly as possible while maintaining acceptable accuracy. If the value of T is too small, we may fail to capture certain attack characteristics that might be apparent over a long period of time. In such case, the detection system will suffer from high error rate. If the value of T is too large, the detection system would not be able to detect attacks until the end of T, and it eventually increases the detection time and prevents real-time detection. We thus examine the flow characteristics within a time interval T of 300 s, the optimal time interval suggested by Zhao *et al.* [14]. Table I shows the 11 features that encode the flow characteristics within the given time interval.

In order to include universal network independent features, the source and destination port numbers are not included in the experiments of this paper. That because the use of these features became in efficient when data comes from different network [34].

B. Randomized Data Partitioned Learning Model (RDPLM)

This section describes the building blocks of the proposed RDPLM. The proposed model consists of three consecutive steps as follows.

1) Step I (Modified Forward Selection Ranking): Selecting right features of botnet is of critical importance as it could not only reduce the computational cost but also improve the generalization capabilities of the detection system and give better understanding of the problem [12]. Feature selection methods can be widely categorized into two main categories, individual evaluation (also known as feature ranking) and subset evaluation. Feature ranking methods assess features and assign them weights according to their degrees of importance [35]. In contrast, subset evaluation methods select candidate feature subset based on a certain search method. Feature selection methods might be classified based on incorporating classification model in feature set selection process into filter and wrapper methods [36]. Filter methods depend on training data characteristics to select features with independence of the classification model, whereas, wrapper methods optimize classifier as a part of the feature subsets evaluation and selection processes.

In Step I, a modified forward selection ranking technique is used to eliminate redundant and irrelevant features, which might contain false correlations that obstruct the learning process of the classifiers [37], [38]. As depicted in Fig. 1, we first rank the features/attributes based on their utility and significance on detecting a class type. Information gain ratio with respect to the class is used to evaluate the utility of each feature/attribute in the features/attributes subsets. This is given by

$$\mathrm{IR}(C, A_j) = \big(H(C) - H(C|A_j)\big)\big/H(A_j)$$

where C is the class, and A_j is the jth attribute. Here, $H(\cdot)$ is the entropy function given by

$$H(\cdot) = -\sum_i P_i(\cdot)\log_2[P_i(\cdot)]$$

where, $P(\cdot)$ is the probability operator and i is the number of classes in a given dataset. Second, features are sorted in descending order based on its utility in feature set S_1. The first feature with the highest rank form a new feature subset S_2 from feature set S_1. Then, similarly to forward selection ranking [38], the algorithm adds new feature from S_1 to S_2 and calculates the performances (detection accuracy and training time/time-to-build model) of the classifier when feature set S_2 is used. Performance measure ω, which captures the time complexity and detection accuracy of a classifier, is calculated as follows:

$$\omega(S_i) = \mathrm{TBM}_{S_i} \times \big(100 - \mathrm{Acc}_{S_i}\big)$$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$

where $x_i \in X, y_i \in Y = \{normal, anomaly\}$

- For $j = 1, \ldots, n$:
  Calculate $IR(C, A_j) = (H(C) - H(C \mid A_j))/H(A_j)$;
- Rank A_j based on its $IR(C, A_j)$ to obtain ranked feature set (S_1).
- Choose the first feature from S_1 to form feature set S_2.
- Intialize $S_3 = S_2$.
- **For $k = 2, \ldots, n$:**
  $S_3 = S_2$;
  Add next feature(k) from S_1 to S_3;
  Calcualte $\omega(S_3)$; \\the performance measure of S_3
    If ($\omega(S_3) < \omega(S_2)$)
      $S_2 = S_3$;
      $\omega(S_2) = \omega(S_3)$;
    End If
  End For

End

Fig. 1. Feature selection pseudo code. IR(C, A_j) denotes the information gain ratio of attribute j. $\omega(S_i) = \mathrm{TBM}_{S_i} \times (100 - \mathrm{Acc}_{S_i})$, where Acc_{S_i} and TBM_{S_i} are the detection accuracy and training time of classifier, respectively, when feature set S_i is used.

where, TBM_{S_i} and Acc_{S_i} denote time to build model and detection accuracy of a classifier when feature set S_i is used. We aim to minimize ω, where feature set S_i is said to have better performance than feature set S_j if $\omega(S_i) < \omega(S_j)$. The feature that reduces ω is kept in S_2; otherwise, it is discarded. The algorithm proceeds till all features in S_1 are tested. The resultant features set S_2 includes the selected features set. Fig. 1 shows the pseudo code of Step I.

2) Step II (Simplified Subspacing): To deal with the large-scale botnet dataset in an efficient manner, a data reduction technique that utilizes Voronoi-based data partitioning and clustering techniques is proposed.

Several instance reduction methods have been reported in literature. Some of which rely on selecting what so-called border instances to be the newly reduced dataset. An instance is called a border instance of some class A, if this instance is the nearest neighbor of an instance that belongs to another class, that is not in A. Examples of such algorithms are patterns by ordered projections [39] and pairwise opposite class-NN (POC-NN) [40]. Some other methods rely on clustering, in which the dataset is split into n clusters, and the centroids of the clusters are selected to represent the newly reduced dataset [41]–[43]. For instance, in [42], the generalized-modified Chang algorithm merges the clusters that contain similar classes and then fetches their centroids. Although the previously mentioned methods do not assume any subsequent classifier model to be used, they still assume the existence of fully labeled datasets. In other words, they utilize the knowledge of the labels for the reduction. However, in botnet detection domain, the generation, collection, and labeling of the datasets are both very costly and time-consuming. Therefore, for future research convenience, we do not constrain our reduction technique to fully

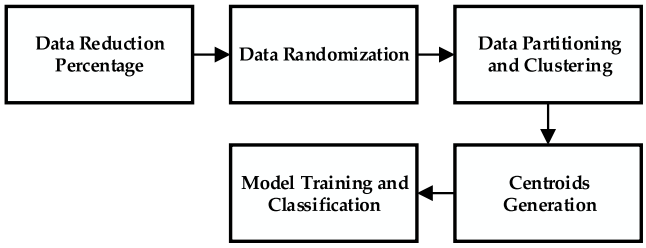

Fig. 2. Stepwise procedure of Step II. SRS is performed based on the specified sampling percentage. Voronoi and K-means algorithms are applied to reduce the dataset size. Then, Step III uses the newly reduced dataset for classification.

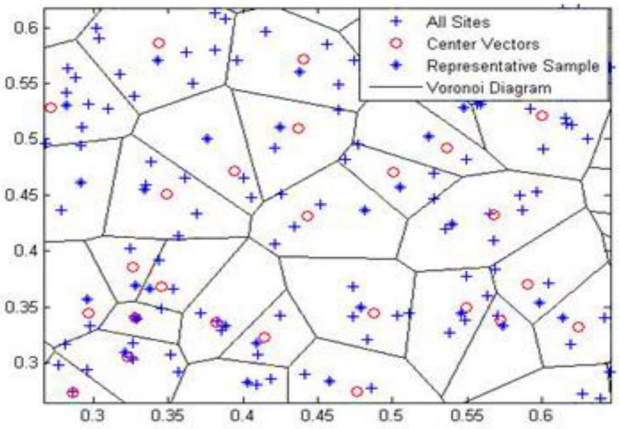

Fig. 3. Voronoi regions.

labeled datasets. In what follows, we introduce simplified subspacing data reduction algorithm based on simple Voronoi data partitioning and clustering techniques.

Given a dataset $X \in \mathcal{R}^{N\times D}$, with N instances and D features, we would like to reduce the dataset to obtain $C \in \mathcal{R}^{n\times D}$, which has $n \ll N$ instances. Our proposed data reduction algorithm consists of two main stages. In the first stage, a Voronoi-partitioning technique clusters the whole input vector space into smaller Voronoi regions based on some randomly selected representative samples. Our motivation beyond this stage is to reduce the complexity of the next stage. We assume that in large scale data, very far away points should not have a big influence on each other. In addition, this stage would allow for parallel computing techniques. In the second stage, we utilize the K-means algorithm to fetch the centroids of each Voronoi region to represent all original input vectors. The collection of all centroids constitutes the newly reduced dataset, C. Thereby, the time to build the subsequent classifier models could be reduced significantly [44]–[46]. In addition, combining with a parametric partitioning model introduces semi-parametric properties to the pure nonparametric learning model so that the final model could be more stable and robust. Fig. 2 illustrates the stepwise procedure of Step II.

In the first stage of the algorithm, simple random sampling (SRS) is performed on the original dataset to select n representative samples out of a total of N samples. SRS selects n equiprobable random samples from the original dataset with replacement following a binomial distribution [47].

Next, we partition the dataset space into n Voronoi regions, based on the selected representative samples. In the case of semi-labeled datasets, the selected representative set in the first stage is assumed to be labeled. There exist many efficient implementation algorithms for the construction of the Voronoi diagram (VD). Out of which, the divide-and-conquer and fortune's line sweep algorithms are most efficient. According to [48, Ths. 3.3 and 3.4], the divide-and-conquer and Fortune's line sweep constructs a VD of N points within time of $O(N\log N)$ and linear space, in the worst case, where both bounds are optimal [48].

Let the representative subset, $S = \{p_1, p_2, \ldots, p_n\} \in \mathcal{R}^D$, be used to partition the dataset space into n Voronoi regions, where D denotes the dimensionality of the input vectors, and (n/N) is the sampling percentage, then the Voronoi region

Given: Dataset S of n points

- Sp = Specify the sampling percentage
- Desired No. of clusters = percentage of data included in sampling

Output: Classification results

Functions and variables used:

- *Sd*: Randomly sampled representative subset based on *Sp*
- *Neighbor(p)*: all neighbors of point *p*
- *Vor(Sd)*: Construct the Voronoi diagram of *Sd*
- *v(p)*: Voronoi region of point *p*
- *Rd*: Reduced dataset

Start

- Generate *Sd* via *SRS*
- Call *Vor(Sd)*
- Select random point *p* from *Sd*
- Call *Neighbor(p)*
- Get centroid of each cluster via *K*-Means algorithm
- Extract the reduced dataset (*Rd*)

Stop

Fig. 4. Step II pseudo code.

centered by a representative instance p_i, is defined by

$$\text{Vor}(p_i) = \left\{X \in \mathcal{R}^D \;\middle|\; \|x - p_i\| \le \|x - p_j\| \quad \forall j \in S\right\}$$

where $\|\cdot\|$ represents the L_2 norm operator.

In the second stage, we overlay the entire (original) dataset into the constructed Voronoi regions. Then, the centroid of each Voronoi region is fetched using the K-means algorithm. The label of the centroids followed the labels of the selected representative subset. Thereby, the data size is reduced to the previously specified sampling percentage (n/N). As an illustrative example, we perform the instance reduction algorithm on some uniformly random 2-D dataset. Fig. 3 illustrates the two-stage reduction algorithm, where we firstly select random representative subset and construct a VD. Then, for each Voronoi region, we perform K-means clustering. Fig. 4. depicts the pseudo code of Step II.

3) Step III (Multiple Randomized Trees): Due to the heterogeneity observed in ISOT dataset, different data types from different resources (e.g., HTTP browsing and gaming), using a single classifier is not a wise choice. We thus construct

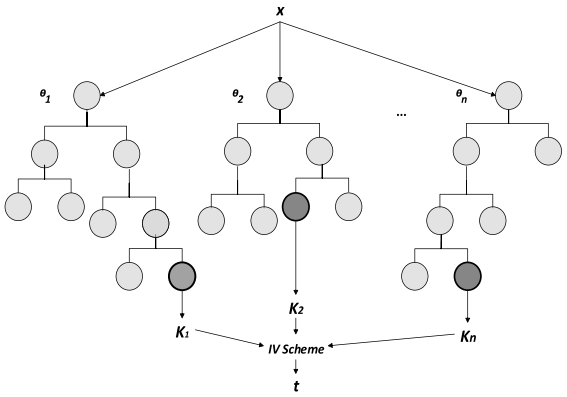

Fig. 5. General architecture of Step III ensemble. The collection of trees $\{h(x, \theta k), k = 1, 2, \ldots, n\}$, where the θk are independently, identically distributed RTrees, and each tree casts "a unit vote" for the final classification of input x.

ensemble model that combines a number of methods and procedures to exploit network traffic characteristics effectively.

Typical to RF, Step III utilizes that if we take a large collection of weak learners, each performing only better than chance, then by putting them together, it is possible to make an ensemble learner that can perform arbitrarily well. However, base classifiers of ensemble model should be built on nonidentical training datasets in order to reduce variance error and provide diversity among them. Randomness is introduced by bootstrap resampling to grow each tree in the ensemble learner, and also by finding the best splitter at each node within a randomly selected subset of inputs. Step III grows many DTs [49] as in Fig. 5. To classify a new input vector x, put the input vector down each of the DTs in the ensemble classifier. Each DT is trained on a bootstrap sample of the training data [50]. The DTs are used in this step and the entropy was used to rank attributes in terms of the reduction in the uncertainty of the class label. The algorithm finds the attribute which has the highest information gain, which represents the expected reduction of entropy by partitioning examples based on certain attribute, and places it on the root node of the tree. Then, creates the branches which represent the possible values of parent node.

Step III performs a kind of cross-validation by using out-of-bag (OOB) data. Since each tree in the ensemble grows on a bootstrap sample of the data, the sequences left out of the bootstrap sample, the OOB data, can be used as legitimate test set for that tree. On average 1/3 of the training data will be OOB for a given tree. Consequently, each data in the training dataset will be left out of 1/3 of the trees in the ensemble, and use these OOB predictions to estimate the error rate of the full ensemble.

Like classification and regression tree [51], Step III uses the gini impurity for determining the final class in each DT. The gini impurity of node impurity is the measure most commonly chosen for classification-type problems. If a dataset T contains examples from n classes, gini impurity $G(T)$ is defined as

$$G_{\text{split}}(T) = 1 - \sum_{j=1}^{n} (P_j)^2$$

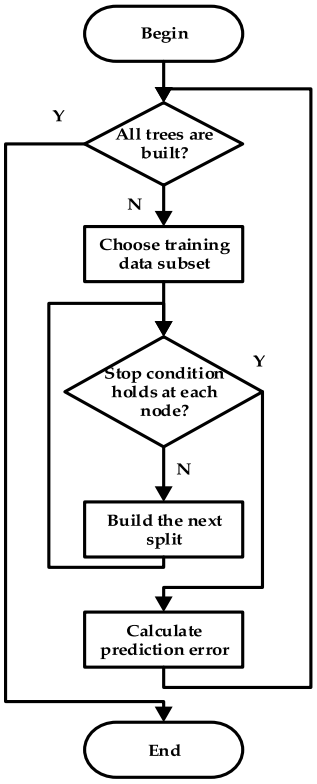

Fig. 6. Flowchart of Step III ensemble.

where P_j is the relative frequency of class j in T. If a data set T is split into two subsets $T1$ and $T2$ with sizes $N1$ and $N2$, respectively, the gini impurity of the split data contains examples from n classes, the $G_{\text{split}}(T)$ is defined as

$$G_{\text{split}}(T) = \frac{N_1}{N}\, G(T_1) + \frac{N_2}{N}\, G(T_2)$$

the attribute value that provides the smallest $G_{\text{split}}(T)$ is chosen to split the node.

Fig. 6 illustrates the flowchart of the Step III ensemble. First, a random collection of samples from the training dataset is selected while maintaining the class distribution. Second, with this selected dataset, a random set of attributes from the original dataset is chosen based on user defined values. All the input variables are not considered because of enormous computation and high chances of overfitting. In a dataset where M is the total number of input attributes in the dataset, only R attributes are chosen at random for each tree where $R = \log_2(M) + 1$, $R < M$. Third, the attributes from this set create the best possible split to develop a DT model. The process repeats for each of the branches until the termination condition stating that leaves are the nodes that are too small to split or only have data of the same class. Step III builds randomized tree in each iteration of bagging algorithm, and often produces excellent classifier. Each tree votes for class type and the final decision is the majority votes of all trees.

IV. IDS Evaluation and Analysis

The most common and well-accepted statistical methods to evaluate the performance of a classifier are resubstuituation,

TABLE II
FEATURE SETS

No	Method	Features
1	CFS	PV, PX, FPS, FPH
2	Wrapper	Proto, FPS, TBP, FPH
3	RDPLM–S I	Proto, APL, FPS, NR, FPH

The parameters of Wrapper method were given the following values: classifier=J48, seed=1, threshold=0.01. Best first search was used to generate the best discriminatory features by both methods (CFS and Wrapper). RDPLM–S1 denotes features obtained through Step I.

cross-validation, and bootstrapping. In this paper, we use cross-validation to assess the detective performance of the proposed model. In our evaluation, the original dataset is divided into ten random subsets where nine subsets are used for training and the other one subset is used for testing. This process is repeated, recursively, until all subsets are tested. The classification errors of every subset is accumulated then the mean absolute error is computed. This method gives us better understanding of how our proposed method performs in real time situations.

Different measures are used to evaluate the performance of each classifier, namely, classification accuracy (Acc), DR (also known as, sensitivity), false alarm rate (FAR), Mcc, standard deviation of F-measure ($\boldsymbol{\sigma}$), time has been taken to build model (TBM) and (TT). Acc reveals classifier's ability to correctly classify normal and botnet traffic; DR is the number of intrusion traffic detected by the model divided by the total number of intrusions in the test set; FAR refers to the percentage of normal traffic classified as intrusion; and Mcc is a correlation coefficient between the observed and detected binary classification that has a value between -1 and $+1$, a coefficient of $+1$ represents a perfect detection, 0 means no better than random detection and -1 indicates total disagreement between detection and observation. A high value of Mcc means more robust detection model. $\boldsymbol{\sigma}$ provides a good idea about model's generalization ability, and stability. We aim for a high Acc, DR, and Mcc, and low $\boldsymbol{\sigma}$, TBM, FAR, and TT. The above measures can be defined as follows:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{DR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FAR} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

$$\text{Mcc} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

where true-positive (TP) is the number of correctly classified intrusions, true-negative (TN) is the number of correctly classified normal traffic, FN is the number of incorrectly classified intrusions as normal traffic, and false-positive (FP) is the number of incorrectly classified normal traffic as intrusions. Our dataset contains network flow attributes that describe flow information within a 300 s window. For each flow, there is at least one packet that was exchanged. We generated 689 205 instances, where 7394 are malicious and 681 811 are nonmalicious instances.

TABLE III
PERFORMANCE OF RDPLM-S III ON 11FS

Model	Acc	DR	FAR	Mcc	σ	TBM(S)	TT(S)
S III	**99.9845**	**0.9942**	0.00009	**0.9927**	0.0031	36.933	0.104
REPTree	99.9819	0.9930	0.00011	0.9915	0.0028	10.454	0.044
RTree	99.9816	0.9905	**0.00008**	0.9913	0.0032	**5.857**	**0.042**
C4.5	99.9811	0.9930	0.00011	0.9911	**0.0020**	28.917	0.044
DNN	99.9575	0.9869	0.00029	0.9801	0.0041	889.380	0.169
SMO	99.9491	0.9662	0.00015	0.9758	0.0033	7.033	0.090

We compared the performance of S III with Reduced Error Pruning Tree (REPTree), Random Tree (RTree), C4.5 decision tree, multilayered Deep Neural Network (DNN) and Sequential Minimal Optimization (SMO). $\boldsymbol{\sigma}$ and TT denote StdDev F-measure and testing time, respectively. Each model has been tested on the 11 feature set described in Table 1 except SrcPort and DstPort, with flow characteristics within interval of 300s. The parameters of Step III (S III) were given the following values: numIterations=10, weightThreshold=100, numTrees=10, seed=1.

TABLE IV
MODELS PERFORMANCES ON CFS FEATURE SET

Model	Acc	DR	FAR	Mcc	σ	TBM(S)	TT(S)
SMO	99.9362	0.9563	0.00017	0.9696	0.0050	**2.9852**	0.065
DNN	99.9527	0.9843	0.00031	0.9779	0.0037	363.5053	0.096
C4.5	**99.9793**	**0.9950**	0.00016	**0.9903**	0.0028	15.0080	0.042
REPTree	99.9784	0.9949	0.00016	0.9899	0.0026	4.8827	**0.040**
RTree	99.9739	0.9870	**0.00012**	0.9877	0.0026	3.4433	0.043
S III	99.9752	0.9888	0.00013	0.9883	**0.0024**	23.9680	0.090

Each model used the feature set selected by CFS method. The parameters of Step III (S III) were given the following values: numIterations=10, weightThreshold=100, numTrees=10, seed=1.

To show the effectiveness of the proposed methods, their performance is compared with those of other well-known ML algorithms on feature set derived through methods reported in literature like CFS [52], which is used in [12], WrapperSubsetEval [53], and Step I. Table II shows feature sets that are produced by the different feature selection methods.

Table III shows the performances of each algorithm on the Botnet benchmark-11Fs dataset. Experiment results show that RDPLM-S III (Step III) outperforms other ML algorithms used in the paper. RDPLM-S III has achieved the highest Acc (99.9845), DR (0.9942), and Mcc (0.9927), which are better than those of SMO, DNN, C4.5, REPTree, and RTree, with very low FAR (0.00009). C4.5 achieved the highest stability and generalization capacity $\boldsymbol{\sigma}$ (0.0020).

Table IV shows models performances on feature set that is produced by CFS feature selection method. C4.5 achieved the highest Acc, DR, and Mcc, while Step III obtained the highest generalization and stability denoted by $\boldsymbol{\sigma}$. It is noticeable that, generally, models performances in term of Acc, DR, FAR, and Mcc degrade when these models are build on feature set produced by CFS method. As expected, models' TBM and TT reduced when less number of features is used to build classifier.

Table V shows models performances when models are built on feature set that is generated by wrapper method. As expected, C4.5 has achieved the best performances in terms of Acc (99.9842), DR (0.9977), and Mcc (0.9926), that because wrapper method optimized C4.5 classifier in the feature subset evaluation and selection process. Although, feature set, that is produced by wrapper method, enhanced the performances of C4.5 classifier and achieved better detection accuracy than using the full feature set, it does not enhance the performances of other models, which means that the selected features are not

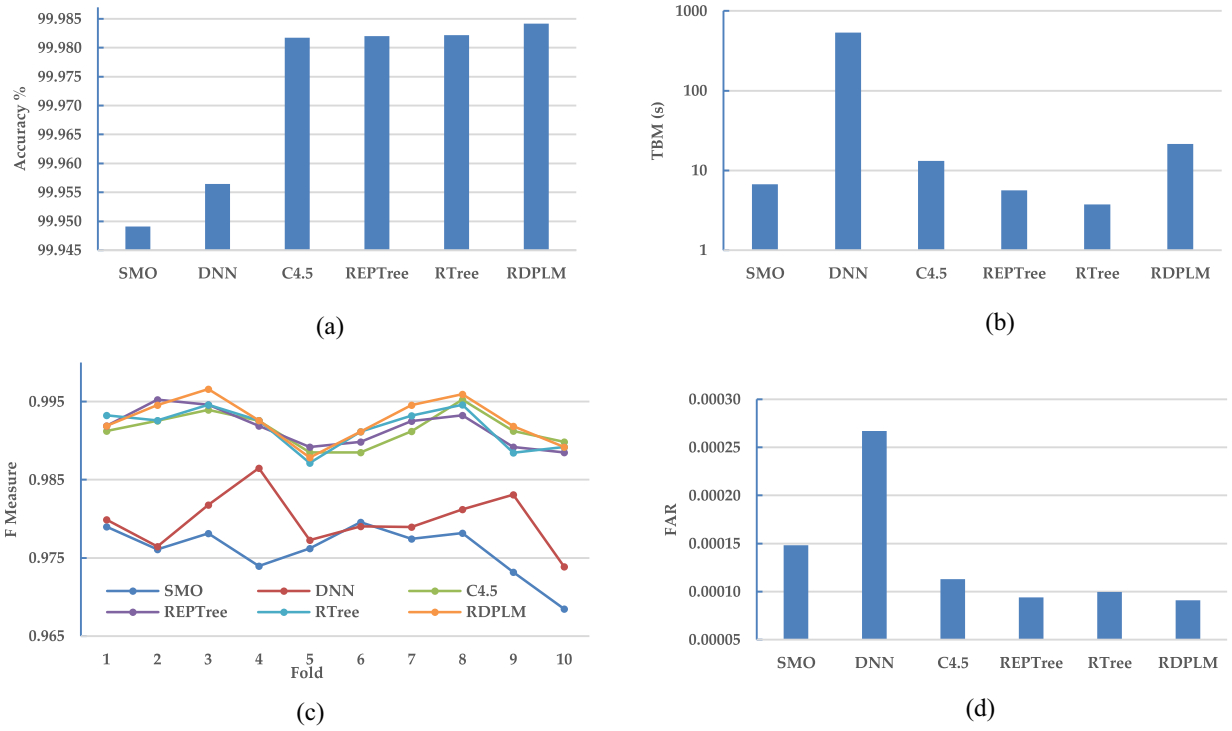

Fig. 7. Models performance comparisons. (a) Accuracy of SMO, DNN, C4.5, REPTree, RTree, and RDPLM. (b) Time to build model of SMO, DNN, C4.5, REPTree, RTree, and RDPLM. (c) F-Measure of SMO, DNN, C4.5, REPTree, RTree, and RDPLM. (d) False Alarm Rate of SMO, DNN, C4.5, REPTree, RTree, and RDPLM.

TABLE V
MODELS PERFORMANCE ON WRAPPER FEATURE SET

Model	Acc	DR	FAR	Mcc	σ	TBM(S)	TT(S)
SMO	99.9486	0.9656	0.00015	0.9756	0.0035	6.2095	0.078
DNN	99.9544	0.9827	0.00027	0.9786	0.0037	503.4210	0.121
C4.5	**99.9842**	**0.9977**	0.00013	**0.9926**	0.0023	11.2350	0.044
REPTree	99.9804	0.9966	0.00016	0.9908	**0.0022**	4.5717	**0.040**
RTree	99.9756	0.9888	**0.00012**	0.9885	0.0025	**3.3538**	0.043
S III	99.9779	0.9904	**0.00012**	0.9896	0.0027	20.1872	0.088

Each model used the feature set selected by Wrapper method where C4.5, a.k.a J48, classifier is used in subset evaluation. The parameters of Step III (S III) were given the following values: numIterations=10, weightThreshold=100, numTrees=10, seed=1.

TABLE VI
MODELS PERFORMANCES ON RDPLM-SI FEATURE SET

Model	Acc	DR	FAR	Mcc	σ	TBM(S)	TT(S)
SMO	99.9491	0.9662	0.00015	0.9758	0.0034	6.721	0.078
DNN	99.9565	0.9840	0.00027	0.9796	0.0036	536.032	0.128
C4.5	99.9817	0.9934	0.00011	0.9914	**0.0022**	13.132	0.044
REPTree	99.9820	0.9919	**0.00009**	0.9915	0.0024	5.617	**0.040**
RTree	99.9822	0.9926	0.00010	0.9916	0.0026	**3.729**	0.042
S III	**99.9842**	**0.9936**	**0.00009**	**0.9926**	0.0028	21.387	0.092

Each model used the 5 feature set selected by Step I. The parameters of Step III (S III) were given the following values: numIterations=10, weightThreshold=100, numTrees=10, seed=1.

TABLE VII
PERFORMANCE OF RDPLM-S II AND III ENSEMBLE

%	Acc	DR	FAR	Mcc	σ	TBM(S)	TT(S)
50	99.9648	0.9889	0.00021	0.9860	0.0039	12.8544	0.041
60	99.9730	0.9915	0.00016	0.9897	0.0033	13.9871	0.046
70	99.9769	0.9932	0.00014	0.9914	0.0032	18.3516	0.056
80	99.9756	0.9923	0.00014	0.9907	0.0015	22.1049	0.063
90	99.9798	0.9949	0.00013	0.9925	0.0016	26.0828	0.071

Different scales of reduced datasets have been created by Step II (SII). The table depicts the experimental results of Step III on the different scales. The scale of original dataset is denoted as 100.

the optimum ones. Step III and RTree obtained the lowest false alarm rate.

Table VI shows models performances on Step I feature set, Step III outperforms other models in terms of Acc (99.9842), DR (0.9936), FAR (0.00009), and Mcc (0.9926). It is noteworthy that feature set that is selected through Step I improves the detection accuracy of C4.5, REPTree, RTree, maintains the detection accuracy of SMO, and slightly effects the detection accuracy of DNN. In term of DR, Step I feature set maintains the performance of SMO, and enhances C4.5 and RTree and slightly effects DNN and Step III. In term of FAR Step I does not effect SMO, C4.5, and Step III, while it enhances DNN and REPTree and slightly increases RTree's FAR.

As seen in Table VI, our feature selection algorithm described in Step I enhances the performance of Step III by reducing its TBM, the time complexity of the model. Noticeably, unlike CFS and Wrapper feature sets, the selected features through Step I enhanced the performance of most of the other models. The best performances were achieved by Step III except in model stability and TBM, where C4.5 and RTree excelled, respectively. Step III effectively reduced its TBM from 36.93 s to 21.38 s, which means fast model update. Moreover, the throughput of Step III increases since its TT is reduced from 0.104 s to 0.092 s. This makes the proposed model, RDPLM, a good candidate especially for large-scale botnet detection tasks. All in all, the novel features selection technique of Step I used along with Step III could lead to the best detective performance by achieving Acc = 99.9842, DR = 0.9936, FAR = 0.00009, and Mcc = 0.9926.

We can notice from Table VII that the Acc of Step III is improving as a lower reduction rate is used. Acc and TBM

of Step III were 99.9648 and 12.8544 s, respectively, when the volume of the original dataset has been reduced to 50% using Step II. However, the Acc and TBM reach 99.9798 and 26.0828 s when the volume of the data has been reduced to 90%. When the volume of the data is reduced to 70% and 80%, no significant improvement has been observed in terms of Acc, FAR, and Mcc. However, system stability and generalization capabilities improved. We can notice from Table VII that the proposed model, RDPLM, obtained better performances than DNN and SMO even when the dataset is reduced by 50%. Moreover, its TT is less than other models TT, which means fast classification and high system throughput. That makes it a preferred model to provide real time detection.

Fig. 7(a) depicts the performance of different models in term of Acc. As shown, RDPLM outperforms other models in term of Acc. As in Fig. 7(b), RDPLM-S III TBM is much lower than DNN model. Fig. 7(c) shows that RDPLM-S III shows a stable performance on different folds, while DNN and SMO accuracy noticeably varies on different folds. Fig. 7(d) shows models FAR. RDPLM-S III achieved the lowest FAR than all other models.

V. Conclusion

The concept of IDS has provided us an opportunity to enhance existing network security by detecting, identifying and tracking the attackers. ML algorithms are used in many existing IDSs. However, due to its inherent nature, model complexity and computational requirements could grow exponentially when dealing with large datasets. In this paper, we presented three steps to construct efficient botnet IDS for large-scale networks. To eliminate redundant features and reduce the volume of dataset, we use a novel feature ranking and Voronoi-clustering-based data partitioning techniques. In addition, we use network flow characteristics to detect botnet intrusions regardless of packet payload content, which make it immune to packet encryption. Experimental results show that the proposed methods not only achieved very high detection accuracy (99.984%) but also successfully reduced the model computational cost. This can be seen as a significant contribution as it could prevent the potential data processing delays, a critical requirement of ML-based intrusion detection model when dealing with large-scale networks. As the number of objects connected to networks increases, the security systems face a critical challenge due to the global connectivity and accessibility of Internet of things (IoT). Unlike business computers, which for decades have been sheltered behind corporate firewalls and IDSs, the products now being linked to the Internet are frequently on their own. IoT devices in general have limited computing power and memory. In our future work, we aim to achieve an order of magnitude reduction to the usage of memories during the implementation of IoT devices by using large-scale online-learning techniques. Locally, in online learning model, the memory needed to store the function remains constant even with added datapoints, since the solution computed at one step is updated when a new datapoint becomes available, after which that datapoint can then be discarded. Thus, the solution takes less time to compute with the addition of a new datapoint, as compared to batch learning techniques. Globally, a new large-scale (distributed) learning scheme allocates the computation of learning and data processing among several computing devices. We believe it is a natural way of scaling up learning algorithms to distribute the analytic computations and solve their large and complex problems.

References

[1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proc. 6th ACM SIGCOMM Conf. Internet Meas.*, New York, NY, USA, 2006. pp. 41–52.

[2] N. S. Raghava, D. Sahgal, and S. Chandna, "Classification of botnet detection based on botnet architechture," in *Proc. Int. Conf. Commun. Syst. Netw. Technol. (CSNT)*, Rajkot, India, 2012, pp. 569–572.

[3] B. Al-Duwairi and L. Al-Ebbini, "BotDigger: A fuzzy inference system for botnet detection," in *Proc. 5th Int. Conf. Internet Monitor. Prot. (ICIMP)*, Barcelona, Spain, 2010, pp. 16–21.

[4] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.,* vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[5] J. Zhang and M. Zulkernine, "Network intrusion detection using random forests," in *Proc. PST*, St. Andrews, NB, Canada, 2005, pp. 53–61.

[6] M. Roesch, "Snort—Lightweight intrusion detection for networks," in *Proc. USENIX LISA*, Nov. 1999.

[7] Z. Yu, J. J. P. Tsai, and T. Weigert, "An automatically tuning intrusion detection system," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 2, pp. 373–384, Apr. 2007.

[8] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, "Online adaboost-based parameterized methods for dynamic distributed network intrusion detection," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 66–82, Jan. 2014.

[9] S. T. Sarasamma and Q. A. Zhu, "Min–max hyperellipsoidal clustering for anomaly detection in network security," *IEEE Trans. Syst., Man, Cybern. B, Cybern.,* vol. 36, no. 4, pp. 887–901, Aug. 2006.

[10] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.,* vol. 36, no. 10, pp. 11994–12000, 2009.

[11] M. E. Whitman and H. Mattord, *Principles of Information Security*. Clifton Park, NY, USA: Cengage Learning, 2011.

[12] F. Zhang, P. P. K. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Trans. Cybern.,* to be published.

[13] M.-H. Tsai, K.-C. Chang, C.-C. Lin, C.-H. Mao, and H.-M. Lee, "C&C tracer: Botnet command and control behavior tracing," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC),* Anchorage, AK, USA, 2011, pp. 1859–1864.

[14] D. Zhao *et al.*, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Security,* vol. 39, pp. 2–16, Nov. 2013.

[15] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *Proc. 3rd Int. Conf. Emerg. Security Inf., Syst. Technol., SECURWARE*, Athens, Greece, 2009, pp. 268–273.

[16] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, and W. Lee, "BotHunter: Detecting malware infection through IDS-driven dialog correlation," in *Proc. USENIX Security*, Boston, MA, USA, 2007, pp. 167–182.

[17] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proc. 1st Conf. First Workshop Hot Topics Und. Botnets*, Cambridge, MA, USA, 2007, p. 7.

[18] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," 2008.

[19] S. Arshad, M. Abbaspour, M. Kharrazi, and H. Sanatkar, "An anomaly-based botnet detection approach for identifying stealthy botnets," in *Proc. IEEE Int. Conf. Comput. Appl. Ind. Electron. (ICCAIE)*, Penang, Malaysia, 2011, pp. 564–569.

[20] L. Cao and X. Qiu, "Defence against botnets: A formal definition and a general framework," in *Proc. IEEE 8th Int. Conf. Netw., Archit. Stor. (NAS),* Xi'an, China, 2013, pp. 237–241.

[21] R. Villamarin-Salomon and J. C. Brustoloni, "Identifying botnets using anomaly detection techniques applied to DNS traffic," in *Proc. IEEE 5th Consum. Commun. Netw. Conf. CCNC*, Las Vegas, NV, USA, 2008, pp. 476–481.

[22] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in DNS traffic," in *Proc. 7th IEEE Int. Conf. Comput. Inf. Technol., CIT*, Aizuwakamatsu, Japan, 2007, pp. 715–720.

[23] S.-J. Han and S.-B. Cho, "Evolutionary neural networks for anomaly detection based on the behavior of a program," *IEEE Trans. Syst., Man, Cybern. B, Cybern.,* vol. 36, no. 3, pp. 559–570, Jun. 2005.
[24] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," in *Botnet Detection*. Medford, MA, USA: Springer, 2008, pp. 1–24.
[25] M. M. Masud, T. Al-Khateeb, L. Khan, B. Thuraisingham, and K. W. Hamlen, "Flow-based identification of botnet traffic by mining multiple log files," in *Proc. 1st Int. Conf. Distrib. Framework Appl., DFmA*, Penang, Malaysia, 2008, pp. 200–206.
[26] S. Saad *et al.*, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. 9th Annu. Int. Conf. Privacy, Security Trust (PST),* Montreal, QC, Canada, 2011, pp. 174–180.
[27] F. Rasheed and R. Alhajj, "A framework for periodic outlier pattern detection in time-series sequences," *IEEE Trans. Cybern.,* vol. 44, no. 5, pp. 569–582, May 2014.
[28] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: Detecting botnet command and control servers through large-scale netflow analysis," in *Proc. 28th Annu. Comput. Security Appl. Conf.*, Orlando, FL, USA, 2012, pp. 129–138.
[29] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Honolulu, HI, USA, 2014, pp. 797–801.
[30] L. Braun, G. Munz, and G. Carle, "Packet sampling for worm and botnet detection in TCP connections," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, Osaka, Japan, 2010, pp. 264–271.
[31] (Jan. 2015). *French Chapter of Honenynet*. [Online]. Available: http://www.honeynet.org/chapters/france
[32] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó, "On the validation of traffic classification algorithms," in *Passive and Active Network Measurement*. Berlin, Germany: Springer, 2008, pp. 72–81.
[33] A. Sperotto *et al.*, "An overview of IP flow-based intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 343–356, Jul. 2010.
[34] H. Hang, X. Wei, M. Faloutsos, and T. Eliassi-Rad, "Entelecheia: Detecting P2P botnets in their waiting stage," in *Proc. IFIP Netw. Conf.*, Brooklyn, NY, USA, 2013, pp. 1–9.
[35] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.,* vol. 3, pp. 1157–1182, Mar. 2003.
[36] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowl. Inf. Syst.,* vol. 34, no. 3, pp. 483–519, 2013.
[37] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336. Feb. 2014.
[38] S. Zaman and F. Karray, "Features selection for intrusion detection systems based on support vector machines," in *Proc. 6th IEEE Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, 2009, pp. 1–8.
[39] J. C. Riquelme, J. S. Aguilar-Ruiz, and M. Toro, "Finding representative patterns with ordered projections," *Pattern Recognit.,* vol. 36, no. 4, pp. 1009–1018, 2003.
[40] T. Raicharoen and C. Lursinsap, "A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm," *Pattern Recognit. Lett.,* vol. 26, no. 10, pp. 1554–1567, 2005.
[41] J. C. Bezdek and L. I. Kuncheva, "Nearest prototype classifier designs: An experimental study," *Int. J. Intell. Syst.,* vol. 16, no. 12, pp. 1445–1473, 2001.
[42] R. A. Mollineda, F. J. Ferri, and E. Vidal, "An efficient prototype merging strategy for the condensed 1-NN rule through class-conditional hierarchical clustering," *Pattern Recognit.,* vol. 35, no. 12, pp. 2771–2782, 2002.
[43] C. J. Veenman and M. J. T. Reinders, "The nearest subclass classifier: A compromise between the nearest mean and nearest neighbor classifier," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 27, no. 9, pp. 1417–1429, Sep. 2005.
[44] K. Josien, G. Wang, T. W. Liao, E. Triantaphyllou, and M. C. Liu, "An evaluation of sampling methods for data mining with fuzzy c-means," in *Data Mining for Design and Manufacturing*. New York, NY, USA: Springer, 2001, pp. 355–369.
[45] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[46] M. P. S. Bhatia and D. Khurana, "Experimental study of data clustering using k-means and modified algorithms," *Int. J. Data Mining Knowl. Manage. Process (IJDKP)*, vol. 3, no. 3, pp. 17–30, 2013.
[47] W. N. H. W. Mohamed, M. N. M. Salleh, and A. H. Omar, "A comparative study of reduced error pruning method in decision tree algorithms," in *Proc. IEEE Int. Conf. Control Syst. Comput. Eng. (ICCSCE)*, Penang, Malaysia, 2012, pp. 392–397.
[48] F. Aurenhammer and R. Klein, "Voronoi diagrams," *Handbook of Computational Geometry*, vol. 5. Amsterdam, The Netherlands: Elsevier North Holland, 2000, pp. 201–290.
[49] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.,* vol. 1, no. 1, pp. 81–106, 1986.
[50] L. Breiman, "Bagging predictors," *Mach. Learn.,* vol. 24, no. 2, pp. 123–140, 1996.
[51] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA, USA: Wadsworth and Brooks, 1984.
[52] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, Dept. Comput. Sci., Waikato Univ., Hamilton, New Zealand, 1999.
[53] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Amsterdam, The Netherlands: Morgan Kaufmann, 2005.

Omar Y. Al-Jarrah received the B.S. degree in computer engineering from Yarmouk University, Irbid, Jordan, in 2005, and the M.S. degree in computer engineering from the University of Sydney, Sydney, NSW, Australia, in 2008. He is currently pursuing the Ph.D. degree with Khalifa University, Abu Dhabi, UAE.

His current research interests include machine learning, intrusion detection, big data analytics, and knowledge discovery in various applications.

Omar Alhussein received the B.Sc. degree in communications engineering from Khalifa University, Abu Dhabi, UAE, in 2013, and the M.A.Sc. degree in engineering science from Simon Fraser University, Burnaby, BC, Canada, in 2015.

In 2014, he was a Research Assistant with the Etisalat BT Innovation Centre, Khalifa University, for five months. Since 2014, he has been with the Multimedia Communications Laboratory, Simon Fraser University. His current research interests include spanning signal processing, wireless communications, and machine learning.

Mr. Alhussein currently serves as a Reviewer for the IEEE COMMUNICATIONS LETTERS and other flagship conferences.

Paul D. Yoo (SM'13) received the Ph.D. degree in engineering and information technology from the University of Sydney (USyd), Sydney, NSW, Australia, in 2009.

He was a Post-Doctoral Scientist with the Centre for Distributed and High Performance Computing, USyd, from 2008 to 2009, and a Doctoral Researcher (Quantitative Analysis) with the Capital Markets CRC, Sydney, administered by the Australia Federal Department for Education, Science and Training from 2004 to 2008. He was an Assistant Professor of Data Science with the ATIC-Khalifa Semiconductor Research Centre and the Electrical and Computer Engineering Department, Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, from 2009 to 2014. He is currently a Lecturer (equivalent Assistant Professor) with the Department of Computing and Informatics, Bournemouth University, Poole, U.K. He is also affiliated with the USyd, the Aristotle University of Thessaloniki, Thessaloniki, Greece, and the Korea Advanced Institute of Science and Technology, Daejeon, Korea, as a Senior Fellowship Scientist. He holds over 50 prestigious journal and conference publications.

Dr. Yoo serves as an Editor of the IEEE COMMUNICATIONS LETTERS and *Elsevier JBDR* journals.

Sami Muhaidat (SM'11) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2006.

From 2007 to 2008, he was an NSERC Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. From 2008 to 2012, he was an Assistant Professor with the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada. He is currently an Associate Professor with Khalifa University, Abu Dhabi, UAE, and a Visiting Reader with the Faculty of Engineering, University of Surrey, Guildford, U.K. His current research interests include advanced digital signal processing techniques for image processing and communications, machine learning, cooperative communications, vehicular communications, multiple-input-multiple-output, and space time coding. He has authored over 100 journal and conference papers in the above topics.

Dr. Muhaidat was a recipient of several scholarships during his undergraduate and graduate studies and the 2006 Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship Competition. He currently serves as a Senior Editor for the IEEE COMMUNICATIONS LETTERS, and an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Kamal Taha (SM'14) received the Ph.D. degree in computer science from the University of Texas at Arlington, Arlington, TX, USA, in 2010.

He has been an Assistant Professor with the Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, UAE, since 2010. He was an Instructor of Computer Science with the University of Texas at Arlington, from 2008 to 2010. He was an Engineering Specialist with Seagate Technology, Cupertino, CA, USA, from 1996 to 2005. He has over 60 refereed publications that have appeared in prestigious top ranked journals, conference proceedings, and book chapters. Fifteen of his publications have appeared (or are forthcoming) in the IEEE TRANSACTIONS journals. His current research interests include information forensics and security, bioinformatics, information retrieval, data mining, and databases, with an emphasis on making data retrieval and exploration in emerging applications more effective, efficient, and robust.

Dr. Taha serves as a member of the Program Committee, Editorial Board, and Review Panel for a number of international conferences and journals, some of which are the IEEE and Association for Computing Machinery (ACM) journals.

Kwangjo Kim (M'12) received the B.Sc. and M.Sc. degrees in electronic engineering from Yonsei University, Seoul, Korea, in 1980 and 1883, respectively, and the Ph.D. degree from the Division of Electrical and Computer Engineering, Yokohama National University, Yokohama, Japan, in 1991.

He was a Visiting Professor with the Massachusetts Institute of Technology, Cambridge, MA, USA, and the University of California at San Diego, La Jolla, CA, USA, in 2005, and the Khalifa University of Science, Technology and Research, Abu Dhabi, UAE, in 2012, and an Education Specialist with the Bandung Institute of Technology, Bandung, Indonesia, in 2013. He is currently a Full Professor with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, Korea, the Korea representative to IFIP TC-11, and the honorable President of the Korea Institute of Information Security and Cryptography (KIISC). His current research interests include theory of cryptology, information security, and its applications.

Prof. Kim had served as a Board Member of the International Association for Cryptologic Research (IACR) from 2000 to 2004, the Chairperson of Asiacrypt Steering Committee from 2005 to 2008, and the President of the KIISC in 2009. He is a member of Institute of Electronics, Information and Communication Engineers, IACR, and ACM.