Received May 14, 2019, accepted May 26, 2019, date of publication May 29, 2019, date of current version June 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919680

Collaborative Framework for Early Detection of RAT-Bots Attacks

AHMED A. AWAD [1], **SAMIR G. SAYED** [1,2,3], **AND SAMEH A. SALEM** [1,3]

[1]Department of Electronics, Communications, and Computer Engineering, Faculty of Engineering, Helwan University, Cairo 11792, Egypt
[2]Department of Electronic and Electrical Engineering, University College London (UCL), London WC1E 6BT, U.K.
[3]Egyptian Computer Emergency Readiness Team (EG-CERT), National Telecom Regulatory Authority (NTRA), Cairo 12577, Egypt

Corresponding author: Sameh A. Salem (sameh_salem@h-eng.helwan.edu.eg)

This work was supported by the Egyptian Computer Emergency Readiness Team (EG-CERT) at the National Telecommunications Regulatory Authority (NTRA), Egypt.

ABSTRACT Attackers tend to use Remote Access Trojans (RATs) to compromise and control a targeted computer, which makes the RAT detection as an active research field. This paper introduces a machine learning-based framework for detecting compromised hosts and networks that are infected by the RAT-Bots. The proposed framework consists of two agents that are integrated to achieve reliable early detection of the RAT-bots. The first agent, the host agent, is responsible for monitoring the system behavior of the running host and raising an alarm for any anomalies. The second agent, the network agent, monitors the network traffic to extract any malicious patterns. The integrated approach improves both the detection ratio and accuracy. However, each approach cannot separately achieve the same performance as the proposed RAT-Bots detection framework. The performance of the introduced framework is evaluated by using real-world benchmark datasets. The experimental results show that the proposed approach can achieve an accuracy of 98.83% with 1.45% false positive rate.

INDEX TERMS Bots, botnets, host-based detection, network-based detection, machine learning algorithms, rootkit behavior.

I. INTRODUCTION

The advances in technology made it possible to do whatever activities from home such as online transactions with banks, shopping from online sites and online video games. This kind of advance aims to comfort the internet users. However, these technological improvements are encountered with a real threat appeared from criminals in the cyber world, who exploit vulnerabilities for their malicious intents. Malwares and Malicious programs are a kind of application that is designed and deployed for the aim of helping the intruder in his/her malicious objectives. Worms, Trojans, viruses, backdoors, keyloggers, botnets and ransomwares are examples of malwares. Attackers use botnets to infect machines then control it. Such approach for controlling an infected machine is achieved by a communication scheme named Command & Control (C&C) channels. The famous protocols used in the C&C communication are IRC and HTTP protocols [1]–[3] to launch synchronized attacks such as spam attacks, distributed denial of service malicious behavior [4], phishing [5] and scanning attacks [6].

Remote Access Trojans (RATs) are an Advanced Persistent software Threat (APT) version of botnet malwares. These are designed for attacks with specific aim such as compromising personal data, deleting important information, damaging a specific machine and controlling the victim machine for malicious intentions such as using the camera or the microphone to capture the personal life of the victims. RAT's primary role is to provide control over a victim machine, which can be achieved by injecting itself into legitimate programs for hiding their malicious activities [7], [8].

RAT software is designed into two parts one installed on the victim host and called RAT server, while the intruder executes the second part (named RAT client). Intruders use social engineering methods such as Drive-by-download to download RAT server on the victim machine. In this context, the gained root privileges at the setup process are preserved by RAT server for launching the required attacks and disabling any functions intimidate its functionality [8]. The communication between RAT master (client) and its

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

slave (server) is either direct which initiated by RAT client or reverse which started by RAT server. Most RATs are conventionally using reverse connections as network security policies prevent external connections [9]. Nevertheless, RAT bots can cause severe damages to infected machines without being detected [10]–[12]. In addition, intruders can control more than one machine independently from the host's location.

The rest of the paper is organized as follows. Section II presents the related work, while the proposed collaborative framework is demonstrated in section III. Sections IV and V provide the experimental results and discussions respectively. Finally, Conclusions are given in section VI.

II. RELATED WORK

Researches have focused on the detection of botnets using network analysis techniques. BotHunder is a passive network monitoring tool developed by Gu *et al.* [13]; for detecting bots through capturing a communication sequence that occurred during the infection cycle. Such approach fails to detect bots with different sequence.

In [14], an anomaly detection technique called botminer has been presented. This technique is used for botnet detection without any dependency on the structure or the communication protocol. It detects botnets by clustering machines that have similar network activity patterns in the monitored network. In this method, more than one infected machine is needed in order to cluster their traffic. Jiang and Omote [9] monitored network activities using machine learning algorithms for early stage RAT detection.

Yamada *et al.* [15] presented an approach for raising alarms for hosts with reconnaissance in their network pattern activities, which compromise other hosts using administrative network protocols. Wu et al. [16] introduced a framework for detecting RAT bots at the network gateway. This framework detects the remotely controlled machines through human operators who raise an alarm on these machines that have a high infection probability. This is achieved by determining the packet directions of IP flows of the monitored machines. The captured IP flows are sliced according to packet arrival time. After that, Naive Bayes classifier is used to decide the infection of a given machine according to the packet direction sequence in its flow fragments. Farinholt *et al.* [17] provided a case study of the behavior of the darkcomets operators. A statistical analysis of the operator's actions is introduced after the execution of several darkcomet bots in a given honeypot.

Other researches focused on the detection of botnets using host-based detection techniques. Stinson and Mitchell [18] showed how to detect botnets by keep track of outsider parameters such as parameters coming from the network. Such approach can be easily defeated by applying any hiding policy like the injection policy, where this policy hides the existence of botnets. Brumley *et al.* [19] developed a Minesweeper which is a RAT detection mechanism. This mechanism captures triggered-based malicious software by finding hidden behaviors and activating it to decide whether its behavior is malicious or not. Nonetheless, automating such mechanism is a difficult task. Cui *et al.* [20] detected malwares by observing internet connected processes autonomously. Nonetheless, malwares can easily hide from such monitoring by running their code inside the body of other benign processes. Law *et al.* [21] decides whether a machine is compromised or not after correlating and analyzing its collected network and memory events.

Liang *et al.* [11] presented an approach for detecting if the host is infected or not, such approach checks the running process and based on the network properties it can differentiate between a RAT and a benign process. Mimura *et al.* [22] developed an extraction mechanism to extract executable codes that are embedded inside document files without running the investigated files. Evasion techniques could be used to evade this mechanism such as obfuscating the executable code using encoding mechanisms; consequently, decryption operations are required before running the extraction algorithms. This method cannot be evaluated against the new RAT bots.

Muthumanickam, and Ilavarasan [23] presented a method for detecting peer-to-peer (P2P) botnets using host-based analysis by monitoring the host and network behavior of a machine. For host analysis, system file, registry, and blacklisted IPs connected to the host machine are monitored. Network analysis is implemented by clustering similar machines using the number of connections of each node with other nodes in the same network, intra/inter-communication degree, and with other nodes in other networks. Next, the suspicious clusters are detected by determining similar behaviors among hosts. The correlator stage decides whether a host is a part of a botnet or not. Then, the host analyzer raises an alarm for the hosts in the suspicious cluster. The performance of this method could be improved by using advanced data mining techniques that could lead to better results.

Zeng *et al.* [24] used a combined method for botnet detection, by monitoring host machine behavior such as monitoring changes in the file system, registry, port numbers and IP connections especially the applications that use utilize simple mail transfer protocol (SMTP). After that, each host will have a feature vector which will be classified by supervised machine learning technique to detect if the host is compromised or not. In the network analysis phase, the researchers use a technique similar to botminer technique with a different feature vector and datamining techniques for detecting similar flows. After that, the correlator combines alarms coming from host analyzers with hosts in suspicious clusters to decide whether a given host is compromised or not. Several issues face this framework. Firstly, the probability of a host analyzer is being infected and deceived the correlator by sending false alarms. Secondly, the network analyzer could not group suspicious bots in the same cluster due to their connection with different C&C servers. The third is the scalability issue. Abdullah *et al.* [25] presented a combined method for detecting P2P botnets. At the host level, the file

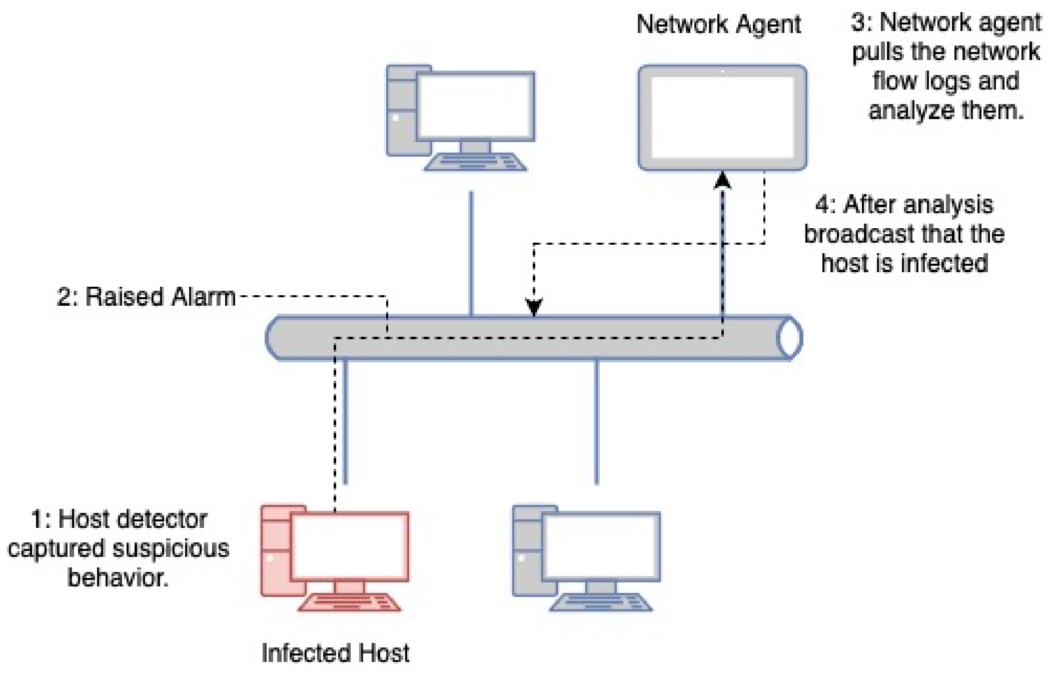
Network Agent
3: Network agent pulls the network flow logs and analyze them.
4: After analysis broadcast that the host is infected
2: Raised Alarm
1: Host detector captured suspicious behavior.
Infected Host

FIGURE 1. The collaborative framework lifecycle: Red-colored host is the infected host. Upon an infection, the host detector sends an alarm to the network agent for a deep investigation.

system, registry, and system logs are monitored at the same time the network analyzer performing full packet payload inspection to identify any suspicious activities from the traffic generated by the host. After that, filtering is performed to the flows according to the used protocol. Finally, in the correlator stage, the feature vectors that are constructed from the host and network data are clustered and then classified according to P2P botnet behavior dataset. This method encounters difficulties in identifying normal and abnormal P2P behavior for classification purposes.

This paper presents a collaborative framework for RAT-bot detection. This framework has two phases that represent the host phase and the network phase. The first phase is the host phase, where the host detector monitors the system behavior and raises an alarm if, and only if, any anomaly behavior is detected. The anomaly detection policy is based on a sequence of activities considered as an indication for suspicious behaviors. Any process exhibits this sequence is considered as a suspicious process.

Regarding the network phase, the network detector starts an investigation upon receiving an alarm from the host detector. An investigation is defined by mapping the infected host's network logs to a set of predefined features. Afterward, a final decision is made based on this investigation.

The introduced framework is designed for RAT botnet detection without considering the used C&C communication architecture or protocol. In addition, combining the host and network techniques enhance the performance and give a robust detection. The heuristic procedure has been selected for its efficiency and low response time to be used in the detection process on the host side. On the other hand, the use of machine learning techniques in the network detection phase is for avoiding the problems that float on the surface when using the signature detection techniques such as the failure in detection of zero-day botnets. Moreover, the usage of statistical attributes in the presented framework enables us to avoid problems that could be appeared if the C&C communication data is encrypted or the concerns that could be raised from saving the confidentiality and privacy of data of the users.

III. THE COLLABORATIVE FRAMEWORK

A. OVERVIEW

The proposed framework has a two-phase decision-making process; the first decision is about detecting unfamiliar behavior on the host machine. In case of any suspicious behavior, it decides to raise an alarm. Then such an alarm is passed to the second decision maker which pulls the suspected host's network flow for further analysis. Based on the analysis of results, a decision is made whether to report this host as an infected host or not.

For the sake of clarity, Figure 1 shows the lifecycle of an infected machine. Firstly, the host detected an anomaly in the host's behavior (message 1); therefore it reported this anomaly through an alarm (message 2). Upon receiving such an alarm, the network agent pulls this host's network flow then starts a deeper investigation. Based on the investigation results, it decides that this host is infected (message 3); then it broadcasts the investigation results.

As noticed from Figure1, the two-phase decision-making process is achieved through two agents. The Host Agent Detector (HAD) which is responsible for capturing any

TABLE 1. Host features.

Feature	Description	Type
F0	No. of opened connections	Network
F1	No. of distinct IPs	Network
F2	Rootkit behavior	Hooking
F3	Suspicious behaviors	Process

suspicious behavior while Network Agent Detector (NAD) hunts down any infected machines by analyzing their network's flow logs. NAD performs its task whenever it receives an alarm raised by HAD or on a predefined time period. Procedure.1 describes the HAD while Procedure 2 defines how the NAD is implemented.

B. HOST AGENT DETECTOR (HAD)

The main functionality of the Host Agent Detector (HAD) is to track the evidence of successful RAT infections through collecting memory, file system, registry, and network traces along with the sequence of API calls for a given host. These system activities are archived by the monitoring module. The tracking module tracks the infection evidence for each process and raises an alarm for any suspicious process. The correlation of intrusion measurements allows the analyst to acquire higher-level investigation of the alarms produced by the host. Consequently, it is necessary to refine the noise-level problems when using network Intrusion Detection System (IDS) only. The monitoring module is the module that is responsible for capturing the system behavior during the run time of the machine.

This module consists of three components to track the full functionality of the host to be monitored. The three components are responsible for monitoring activities such as process spawning, number of opened ports, number of connected IPs for each process, and gained root accesses for controlling the host as examples of RAT behaviors. The real-time behavior of each process is monitored by hooking the Windows API calls to intercept the sent arguments. It should be noted that the proposed framework uses the modules of Microsoft Operating System (MS-OS) service provider interface (SPI) to perform the hooking process. Table 1 shows the events that are tracked for each process.

The behavior analysis of many recent bots shows that most of them share the same suspicious behaviors when compromising a machine. Accordingly, a set of heuristic features have been selected to be used in the decision process as shown in Table 1. Number of opened connections (F0) and the distinct connected IPs (F1) capture RAT's network behaviors. A RAT process opens fewer connections and connects to a smaller number of distinct IPs compared with benign processes. F2 feature identifies Rootkits behavior. Any malicious behavior detected for any given process is reported as feature F3. Many suspicious behaviors could be monitored such as the execution of processes from paths dissimilar from their original ones; the spawning of processes from parents dissimilar from their original parents; the connection of some processes to the internet while in ordinary cases this connection should not exist. The Tracking module is the component that is responsible for capturing the infection evidence for each host as illustrated in Procedure 1.

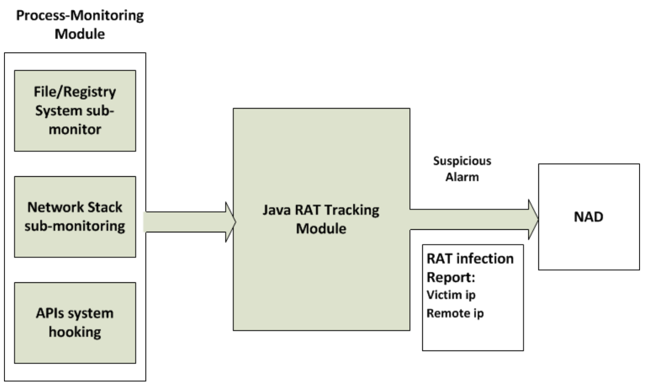

FIGURE 2. The host agent detector (HAD).

Procedure 1: Host Agent Detector

```
1. While TRUE
2.     Let F0 = CALL OPEN-CONNECTIONS-COUNT()
3.     Let F1 = CALL CONNECTED-IPS()
4.     Let F2 = CALL ROOTKITS()
5.     Let F3 = CALL SUSPICIOUS()
6.     If (F0 ∧ F1) ∧ (F2 ∨ F3) is TRUE
7.     then
8.         CALL RAISE-ALARM()
9.         CALL SEND-HOST-INFORMATION()
```

As depicted in Procedure 1, since the HAD has to monitor the host all the time; therefore, it should not have any termination condition.

At the beginning of each monitor cycle, the HAD captures opened connections ports, current connected IPs, Rootkit's behavior and any other suspicious activities such as the execution of processes from different paths; the spawning of processes from different parents; abnormal internet connections for some processes as depicted in lines 2-5 for Procedure 1. Then, the HAD raises an alarm when both an open connection with distinct IPs occurs with either a rootkit or any suspicious activity (lines 6-9).

C. NETWORK AGENT DETECTOR (NAD)

Regarding the Network Agent Detector (NAD), the well-known Random Forest classifier is used to decide whether the suspected host is infected or not through investigating its network flow logs and the received information from HAD. The NAD module consists of three components as shown in Figure 3. The first component is the component responsible for collecting the flows of the suspected host. The other component is the one, which is responsible for deciding on the host under investigation using the random forest machine learning technique on a given

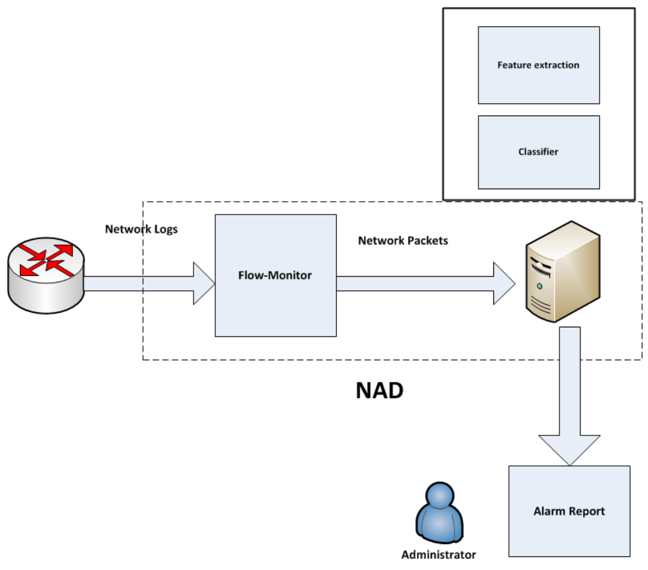

FIGURE 3. The network agent detector (NAD).

extracted set of features. The NAD algorithm is depicted in Procedure 2.

Procedure 2 Network Agent Detector

```
1. If CHECK-ON-ALARMS() $\vee$ TIME-OUT() is TRUE
2. then
3.     Let FEATURES = CALL QUEUE()
4.     If CHECK-ON-ALARMS() is TRUE
5.     then
6.         CALL PUSH(FEATURES, CALL GET-RAISED-INFO())
7.     If TIME-OUT() is TRUE
8.     then
9.         For Each HOST-INFO in CALL GET-HOSTS()
10.            CALL PUSH(FEATURES, HOST-INFO)
11.    While FEATURES is not EMPTY
12.        Let HOST-FEATURE = CALL POP(FEATURES)
13.        If CALL RANDOM-FOREST(HOST-FEATURE) is INFECTED
14.        then
15.            CALL REPORT-INFECTED-HOST()
```

According to procedure 2, the NAD executes its decision-process flow upon receiving an alarm from the HAD or at predefined time-intervals (lines 1-2). Based on the activation condition, the NAD decides whether to investigate a single host or the whole network hosts (lines 3-10) by collecting the features values and storing them in a queue. Each entry in the queue is checked through the trained classifier and in case of an infection detection; it reports such infection to an administrator.

In the following subsections, a detailed overview on how the NAD machine learning model was developed and trained starting from the feature extraction phase towards the applied training algorithm will be demonstrated.

TABLE 2. Selected network features.

Feature	Name	Description
F0	PacNumF	Number of sent packets in forward direction
F1	NoB	Number of sent Bytes in forward direction
F2	APS	Average Size of Packets in forward direction
F3	MAPS	Maximum Size of Packets in forward direction
F4	MIPS	Minimum Size of Packets in forward direction
F5	IAT	Packets Inter-Arrival Time in forward direction
F6	APS	Average PayLoad Size in forward direction
F7	HL	Header Length in forward direction
F8	FPacSize	First Packet Size in forward direction
F9	NSP	Number of Small size Packets in forward direction
F10	NPS	Counting the times of setting the PUSH in Forward direction
F11	NUS	Counting the times of Setting the URGENT flag in Forward direction
F12	NFS	Counting the times of Setting the FINE flag in Forward direction
F13	PacNumB	Number of sent packets in Backward direction
F14	NoBB	Number of sent Bytes in Backward direction
F15	APSB	Average Size of Packets in Backward direction
F16	MAPSB	Maximum Size of packets in Backward direction
F17	MIPSB	Minimum Size of Packets in Backward direction
F18	IATB	Packet Inter-Arrival time in Backward direction
F19	APSB	Average Payload Size in Backward direction
F20	HLB	Header Length in Backward direction
F21	FPacSizeB	First Packet Size in Backward direction
F22	NSP	Number of small size packets in Backward direction
F23	NPSB	Counting the times of setting the PSH in Backward direction
F24	NUSB	Counting the times of Setting the URG flag in Backward direction
F25	NFSB	Counting the times of Setting the FIN flag in Backward direction
F26	NONI	No. O/P packets/No. I/P packets
F27	SOSI	size O/P packets/size I/P packets

1) FEATURE EXTRACTION PROCESS

In the first phase, the statistical features of each host are extracted from the network logs of the host. To avoid any privacy disclosure as well as dealing with the encoded traffic [26], the extracted features mapped several observations of the network behavior of several analyzed RATs. These features show the distinct properties that can be used to differentiate between RAT and benign traffics.

Commonly, features such as packet size, payload length, and rate of input to output and interarrival time (IAT) between packets can be used for differentiating RAT traffics. Those features are selected based on several observations such as the input traffic to RAT servers. In RAT-bots, the input traffic is lower than the output traffic from RAT server, which represents RAT client queries. On the contrary, benign applications such as internet browsing are commonly having input traffic larger than the output traffic. Commonly, RAT server traffic tries to hide their communication patterns with their RAT clients as an evasion technique to make their traffic less dense than benign applications [27]. As a result, the attributes

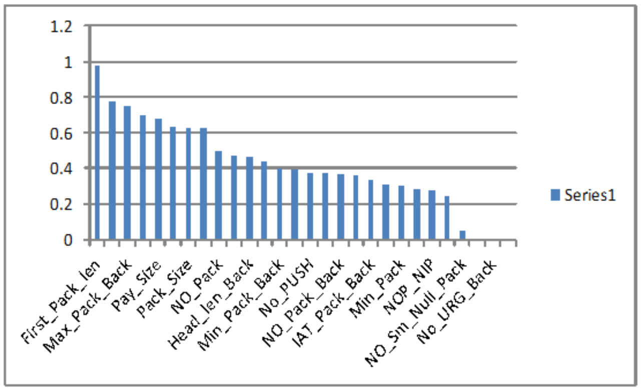

FIGURE 4. Information gain values.

TABLE 3. Reduced network features.

Feature	Name	Description
F0	NoB	Number of sent Bytes in Forward direction
F1	APS	Average Size of Packets in Forward direction
F2	MAPS	Maximum Size of packets in Forward direction
F3	FPacSize	First packet Size in Forward direction
F4	APS	Average Payload Size in Forward direction
F5	NOBB	Number of sent Bytes in Backward direction
F6	MAPSB	Maximum Size of packets in Backward direction
F7	FPacSizeB	First Packet Size in Backward direction

F0 to F27 are selected to achieve this purpose as illustrated in Table 2.

2) FEATURE SELECTION PROCESS

The feature selection phase aims to reduce the time complexity of the classification process without deteriorating the performance level; therefore a subset of features are selected amongst the whole feature set with taking care of not affecting the classification performance. The best features are the ones having the least redundant information and the most relevant to the class selection process [28]. The features are selected according to their information gain values that are computed using Eq. (1).

$$IG\,(Class,\ Feature) = H\,(Class) - H\,(Class|Feature) \qquad (1)$$

As illustrated in Figure 4, the information gain of a twenty-eight statistical network feature is computed to get the best set of features. According to these values, the features with high gain values are selected to be used for classification purpose as shown in Table 3. Eight features are selected for each host in the monitored network to be used in the classification process for the hosts. Several ML algorithms [29] are used for the classification stage to select the classifier with the best performance to be the one which will be used in the detection framework.

3) THE MACHINE LEARNING MODEL

In the classification phase, the feature vectors that are constructed by the feature extraction stage are classified into either benign or malicious which represents RAT actions. For reliable classification purposes, four machine learning algorithms [7] are selected namely K-Nearest Neighbour (KNN) classifier, Random Forest (RF) classifier, Support Vector Machine (SVM) classifier, and Naïve Bayes (NB) classifier. Random Forest (RF) is selected due to its efficiency and accuracy [30]. The SVM algorithm has high scalability with good performance [33]. Naive Bayes classifier and Nearest Neighbour classifier are chosen for their simple implementations and popularity amongst the machine learning applications [31]–[34]. The models have been acquired during the training phase for these classifiers using training data obtained from various benign applications and several malicious samples of RAT botnets.

a: SUPPORT VECTOR MACHINE (SVM)

It is a classification technique that is used in binary classification to separate two classes using a hyperplane. The hyperplane has n-1 dimensionality to classify data points of n dimensions. The margin of the SVM classifier is the maximum distance between the separating plane and the nearest data points. The two classes are best separated when the margin is maximum for both sides. The SVM algorithm has high scalability with good performance in solving difficult classification problems [33]. SVM model is generated using a group of m training points {(x1, y1), (x2, y2), (xk, yk),..., (xm, ym)}, where yk ϵ {− 1,1}demonstrating the category of k-th sample point. After a classification model is obtained, point x could be categorized according to Eq. (2).

$$f\,(x) = sign\,(w.x + c) = sign(\sum\nolimits_{i=1}^{m} a_i y_i\,(x_i.x) + c) \qquad (2)$$

where w and c are used to define the separation hyper plane, a_i denotes the data point's Lagrange multiplier value It is used as a representation of the closeness of the point to the hyper plane where $a_i > 0$ for the close points and zero otherwise.

To handle nonlinear problems, kernel functions should be used for mapping the problem to another feature space to enable the separation. Linear, radial and polynomial basis kernel functions are the most popular kernel functions used in the SVM classifier. In this paper, the Radial Basis Function (RBF) is elected as in Eq. (3). RBF is selected because of its ability to handle classification problems with high complexity as it has a small number of parameters and a lower complexity value than other kernel functions.

$$K\left(x_i, x_j\right) = \exp(\gamma \left\|x_i - x_j\right\|^2) \qquad (3)$$

The parameters (C *and* γ) should be tuned when dealing with the RBF kernel function to adjust the performance of the classifier as to reduce the complexity with acquiring low error rate in the same time [31]. The Kernel function and its parameters are selected and tuned according to the classification problem and its domain.

b: NAIVE BAYES (NB)

It is a probabilistic classification technique that is based on Bayes hypothesis that assumes that independence relationship exists between the features of a given data sample. Accordingly, the posterior probability of class C for a given

TABLE 4. Cross-Validation results for the introduced framework using the reduced feature set.

Classifier	Acc.(%)	TPR(%)	FPR(%)	TNR(%)	FNR(%)	AUC
RF	99.57	99.75	0.621	99.38	0.249	0.999
KNN	98.63	98.14	0.870	99.13	1.863	0.986
SVM	96.15	93.79	1.491	98.51	6.211	0.995
NB	88.94	89.07	11.18	88.82	10.93	0.950

sample with k features is shown in Eq. (4).

$$P(C \mid X) = P(C_n) \prod_{i-1}^{k} P(x_i \mid C) \tag{4}$$

where the data sample X is classified into the category with the highest probability. The simple hypothesis of features independence reduces the size of the training data which lead to decreasing the time needed for creating the classification model [31], [34].

c: KNN (lAZY CLASSIFIERS)

K-Nearest Neighbor is a classification technique in which the classification decision for a given sample is taken based upon the major class of the K-nearest data points to this sample. There is no training phase needed as no classification model is required to perform the classification process. In fact, the computations and decisions are taken during the classification process for the test samples. The decision is taken by comparing the test sample by all the stored samples to determine which K-samples are the closest to the test sample. The dissimilarity measures can be computed using various methods such as Manhattan, Minkowski, and Euclidean distances. Although overfitting issues are resolved using this technique, the classifier performance is degraded when a small training data set is used [24].

d: RANDOM FOREST (RF)

In this algorithm, a bagging technique is used for generating numerous sub-decision trees with a final decision computed by averaging the outputs of these trees. This methodology is used for enhancing the stability as well as the accuracy of the traditional decision tree (DT) algorithm [30].

4) THE MODEL TRAINING ALGORITHM

For reliable evaluations, real-world benign and malicious traffic are used to construct the dataset. The benign traffic is extracted from a benchmark dataset, which is DARPA intrusion detection evaluation dataset [35], and the normal traffic of Egyptian Computer Emergency Readiness Team (EG-CERT) premises [36]. Nine bots are considered the main source of the malicious dataset which are Storm botnet, Cerberus botnet, PandorRAT botnet, Turkojan botnet, Cybergate botnet, GremmRAT botnet, Waledac botnet, XtremeRAT botnet, and NovaliteRAT botnet. The nine RATs are acquired from the EG-CERT. The performance of NAD using various classifiers is illustrated in Table 4. In the training phase, a 10-fold cross-validation (CV) mechanism is used for choosing the training subset, where the captured dataset is divided into training and testing subsets for the purpose

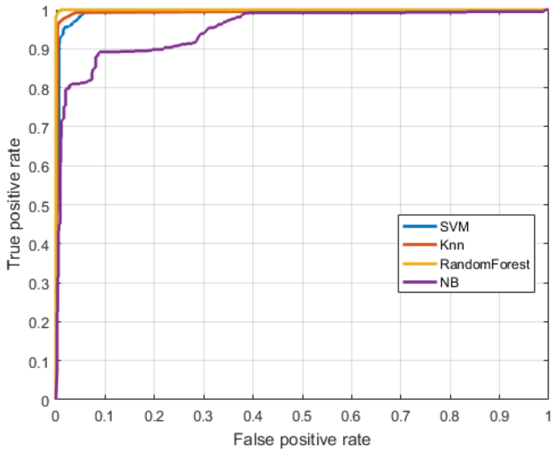

FIGURE 5. Various classifiers ROC curves.

of classification. In the training phase, the classifiers are trained to generate a classification model which is used in the testing phase to evaluate the performance of the classifier. The experiments are conducted using Matlab platform. It should be noted that the parameters for each of the used ML techniques are adjusted to achieve satisfactory performance on the acquired dataset.

As shown in Table 4, the Random Forest (RF) technique affords accuracy above 99% with 0.62% FPR, Followed by the SVM and KNN algorithms. The high results that are obtained from RF could be interpreted to the bagging methodology that is used by the RF algorithm. This technique treats the overfitting problems that are encountered with the decision tree (DT). The NB technique provides the worst performance with an accuracy of 88% and high false positive and false negative rates. Naïve Bayesian (NB) classifier provides low detection accuracy with a high false positive rate. The result of NB could be interpreted as an assumption on network features independence, which is not the case in this research.

Figure 5 represents the ROC curve that depicts the performance of the aforementioned classifiers used in the 10-fold cross validation training and evaluation process. As shown, the classifier with the best performance is the RF classifier with a curve that has an area under curve (AUC) value approaches to one.

For further investigations, several experiments with a different number of trees are carried out to select the best fit number of trees that achieve the best performance as shown in Table 5. However, and it should be noted that the RF classifier suffers from high time complexity.

Concerning the KNN technique, the best performance has been acquired at K = 1 with 98.63% accuracy and low FPR,

TABLE 5. 10-fold Cross-Validation results for the RF classifier for different tree numbers using reduced feature set.

Number of Trees	Acc.(%)	TPR(%)	FPR(%)	TNR(%)	FNR(%)	AUC
10	99.50	99.63	0.621	99.38	0.373	0.999
20	99.57	99.75	0.621	99.38	0.248	0.999
50	9944	99.75	0.870	99.13	0.248	0.999
100	99.57	99.63	0.470	99.50	0.373	0.999
200	99.57	99.75	0.621	99.38	0.248	0.999

TABLE 6. 10-fold cross validation results for KNN classifier for different K values using reduced feature set.

K value	Acc.(%)	TPR(%)	FPR(%)	TNR(%)	FNR(%)	AUC
1	98.63	98.26	0.994	99.01	1.739	0.986
2	98.32	97.64	0.994	99.01	2.360	0.990
3	98.01	97.52	1.491	98.51	2.485	0.992
4	97.83	97.39	1.739	98.26	2.609	0.994
5	97.45	97.02	2.112	97.89	2.981	0.995
6	97.27	97.02	2.485	97.52	2.981	0.995
7	97.27	96.89	2.360	97.64	3.106	0.994

TABLE 7. SVM with different kernel functions' 10-Fold CV Results.

Kernel Function	Acc.(%)	TPR(%)	FPR(%)	TNR(%)	FNR(%)	AUC
$C = 2^0, \gamma = 0.007$	77.52	88.70	33.70	66.30	11.3	0.775
$C = 2^4, \gamma = 0.5$	91.24	89.80	7.30	92.70	10.2	0.912
$C = 2^6, \gamma = 1.0$	94.72	99.10	9.70	90.30	9.0	0.947
$C = 2^8, \gamma = 16$	96.15	93.79	1.49	98.51	6.21	0.995

where various K values are tested, and their results are shown in Table 6. As depicted in the results, the performance of the KNN classifier is impacted by the size of the training data which is considered one of its drawbacks.

The usage of the sequential minimal optimization (SMO) algorithm with the radial basis function (RBF) for the SVM classifier gives an accuracy of 96.15%. The penalty (C) and kernel parameters (γ) are tuned to values which produce the best accuracy. Accordingly, several values are evaluated as shown in Table 7.

According to the aforementioned results, the high accuracy in detection combined with low false positive rates are the main reasons for selecting the RF classifier.

IV. EXPERIMENTAL RESULTS

A. TESTBED ARCHITECTURE

To evaluate the introduced framework, a virtual network with five connected hosts, one RAT client and a host network agent as shown in Figure 6.

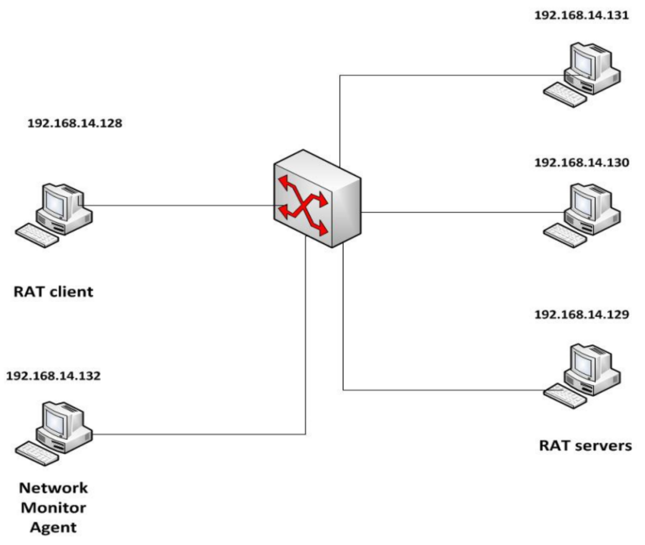

FIGURE 6. Experimental testbed for evaluating the framework.

B. BENCHMARKING METRICS

To evaluate the presented framework, the overall accuracy (ACC), false positive rate (FPR), and false negative rate (FNR) are used as key performance indicators (KPI) as in Eqs. (5):

$$\begin{aligned} ACC &= \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \\ FPR &= \frac{F_P}{F_P + T_N} \\ FNR &= \frac{F_N}{F_N + T_P} \end{aligned} \tag{5}$$

where T_P denotes the number of malicious samples that are detected as malicious. T_N represents the number of benign samples that are detected as benign. F_P indicates the number of benign samples detected as malicious F_N is the number of malicious samples detected as benign.

C. DATASETS

Several experiments are carried out on seven unknown RAT bots for evaluating the performance of the proposed framework. DarkComet botnet, SolitudeRAT botnet, Zeus botnet, SchwarzesonneRAT botnet, SpyNet botnet, ProRAT botnet, and NJRAT botnet are the seven malicious samples that are used in the testing phase. Benign samples, To evaluate the framework against benign applications that have similar RAT behaviors such as browsing applications, desktop applications, and real world DARPA dataset [35].

TABLE 8. Performance evaluation of the proposed framework in testing phase.

Framework	Acc.(%)	TPR(%)	FPR(%)	TNR(%)	FNR(%)	AUC
Hybrid	98.83	100	1.45	98.55	0.00	0.992

TABLE 9. Performance evaluation of the proposed framework in testing phase.

Detector	Independent C&C Architecture	Independent C&C Protocol	Independent Of Infection Cycle
Botminer	√	√	√
Bothunter	√	√	x
Rishi	x	x	√
Collaborative Framework	√	√	√
Botsniffer	x	x	√

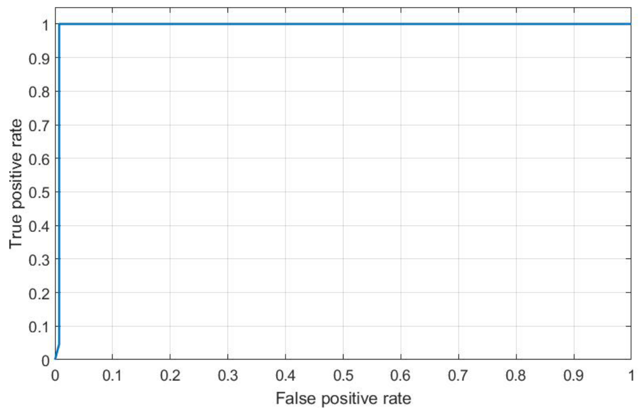

FIGURE 7. ROC curve for the proposed collaborative framework.

V. RESULTS AND DISCUSSION

Experimental results are explored in Figure 7 which depicts the Receiver operating characteristic (ROC) curve for the introduced framework. As depicted, the performance of our framework is illustrated by a ROC curve that has an AUC that is approximated to 1 in the test phase.

As shown in Table 8, the proposed framework provides the best detection accuracy with low false positive and negative rates. The purposed framework succeeded in detecting the malicious RAT samples and reporting their status to the administrator. After deep investigation, it has been found that the 1.45% FPR resulted from running irc program instances on the machines. As this program has the characteristics that are similar to the RAT botnets, the collaborative framework raised an alarm on the machines that have irc client instances ran on it.

A. COMPARISON WITH COMPETITIVE ADVERSARIAL FRAMEWORKS

According to the presented results in Table 9, it could be concluded that the introduced framework provides good detection accuracy with a low false positive rate in the detection of the RAT families which have been tested. Moreover, the collaborative framework presents a better accuracy and FPR than the detection framework that was introduced in [7]. For the host-based framework that was introduced in [8], the collaborative framework has a better FNR and detection accuracy. Combining the host and network approaches enhance the accuracy and lowering the FPR and FNR rates.

The dialog approach that is used by the bothunter [13] framework could results in a detection failure for botnets with infection cycle that is different from the infection cycle that is assumed by the bothunter. This is not the case in the proposed collaborative framework that uses the machine learning techniques in the NAD part for detecting botnet samples with different cycles of infection.

On the other hand, successful detection of infection resulted from the botminer framework [14] needs a large number of infected machines in the same network while the proposed collaborative framework gets over this requirement by using heuristic and machine learning techniques to detect even one infected machine in the monitored network. For the competitive framework in [15], a RAT botnet is detected by analyzing the network traffic to get specific management protocol commands for botnet detection. This could limit the detection capabilities specifically for RAT bots that do not use such types of commands. While the proposed detection framework is a hybrid framework depending on machine learning techniques the detection algorithm on the host machine. Rishi framework [39] and Botsniffer framework [40] both depend on the C&C communication protocol commands for bot detection. The comparison between the frameworks is summarized in Table 9.

It should be noted that the comparison process of botnet detection methodologies that are used in the literature is difficult to achieve due to several factors such as the existence of various RAT botnets versions without appropriate documentation, that help in accurate realizations for each of these techniques. Besides, obtaining datasets of malicious botnets to act as a common ground in the performance evaluation phase is a difficult task as reported in [37], [38], which lead to the infeasibility of comparing different frameworks using different botnet sets which will have different feature sets. The absence of standard methods for comparison as well as common error metrics introduces another challenge in the comparison process. Subsequently, it is concluded that the results of the proposed framework could not be compared with other techniques due to the lack of common botnet datasets, the unavailable description of the introduced techniques, the lack of common comparison methods and performance metrics [41].

B. COMPLEXITY ANALYSIS

For computational analysis purposes, the time complexity is used to evaluate the proposed framework. In this context, the time complexity of each module of the introduced collaborative framework is computed. According to Procedure 1, the time complexity of the HAD module is $O(p)$, where the p is the number of running processes in the host machine, while the running time for the NAD module is $O(nk)$, where n is the number of hosts in the monitored network and k is the number of features. Therefore, the overall time of complexity for the proposed framework is $O\,(p) + O(nk)$.

VI. CONCLUSIONS

This paper has introduced a collaborative framework for RAT-bots detection. Through describing a two-phase decision-making process for detecting infected RAT hosts using machine learning approaches. The first phase is the host agent detector (HAD) which resides on each host on the monitored network, and the network agent detector (NAD) which is installed on the network gateway. The HAD is responsible for monitoring the system behavior of the running machine raising an alarm for any anomalous behavior from any process. This alarm triggers the NAD to start investigating the suspicious host. The investigation is carried out through capturing the network logs of the suspicious host and mapping its network behavior into a structured feature vector for classification purposes. These approaches proved to outperform various available RAT-bots detection frameworks with 98.83% accuracy with 1.45% false positive rate in the experimental results. In the future, further investigation will be carried out along with more effective features are to be selected and computed for enhancing the robustness of the proposed framework. In addition, more RAT botnet samples will be acquired and tested using the detection framework.

ACKNOWLEDGMENT

The authors would like to thank the Malware Analysis and Reverse Engineering team in the EG-CERT for providing helpful ideas and the malicious database used in this research.

REFERENCES

[1] A. Karim, R. Salleh, M. K. Khan, A. Siddiqa, and K.-K. R. Choo, "On the Analysis and Detection of Mobile Botnet Applications," *J. Universal Comput. Sci.*, vol. 22, no. 4, pp. 567–588, 2016.

[2] K. Simon, C. Moucha, and J. Keller, "Contactless vulnerability analysis using Google and shodan," *J. Universal Comput. Sci.*, vol. 23, no. 4, pp. 404–430, 2017.

[3] A. Karim, R. B. Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet detection techniques: Review, future trends, and issues," *J. Zhejiang Univ. Sci. C*, vol. 15, no. 11, pp. 943–983, Nov. 2014.

[4] M. Zahid, A. Belmekki, and A. Mezrioui, "A new architecture for detecting DDoS/brute forcing attack and destroying the botnet behind," in *Proc. Int. Conf. Multimedia Comput. Syst.*, May 2012, pp. 899–903.

[5] A. R. Rodrìguez-Gómez, G. Maciá-Fernández, and P. García-Teodoro, "Survey and taxonomy of botnet research through life-cycle," *ACM Comput. Surv.*, vol. 45, no. 4, Aug. 2013, Art. no. 45.

[6] I. Ghafir, V. Prenosil, M. Hammoudeh, T. Baker, S. Jabbar, S. Khalid, and S. Jaf, "Botdet: A system for real time botnet command and control traffic detection," *IEEE Access*, vol. 6, pp. 38947–38958, 2018.

[7] A. A. Awad, S. G. Sayed, and S. A. Salem, "A network-based framework for RAT-bots detection," in *Proc. 8th IEEE Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2017, pp. 128–133.

[8] A. A. Awad, S. G. Sayed, and S. A. Salem, "A host-based framework for RAT bots detection," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Sep. 2017, pp. 336–342.

[9] D. Jiang and K. Omote, "An approach to detect remote access trojan in the early stage of communication," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl.*, Mar. 2015, pp. 706–713,

[10] M. Mimura, Y. Otsubo, H. Tanaka, and H. Tanaka, "A practical experiment of the HTTP-based RAT detection method in proxy server logs," in *Proc. 12th Asia Joint Conf. Inf. Secur. (AsiaJCIS)*, Aug. 2017, pp. 31–37.

[11] Y. Liang, G. Peng, H. Zhang, and Y. Wang, "An unknown trojan detection method based on software network behavior," *Wuhan Univ. J. Natural Sci.*, vol. 18, no. 5, pp. 369–376, Oct. 2013.

[12] S. C. Pallaprolu, J. M. Namayanja, V. P. Janeja, and C. T. S. Adithya, "Label propagation in big data to detect remote access trojans," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 3539–3547.

[13] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, *Bothunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation*. Berkeley, CA, USA: USENIX Association, 2007.

[14] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proc. 17th Conf. Security Symp.*, Berkeley, CA, USA, vol. 8, 2008, pp. 139–154.

[15] M. Yamada, M. Morinaga, Y. Unno, S. Torii, and M. Takenaka, "RAT-based malicious activities detection on enterprise internal networks," in *Proc. 10th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2015, pp. 321–325.

[16] S. Wu, S. Liu, W. Lin, X. Zhao, and S. Chen, "Detecting remote access Trojans through external control at area network borders," in *Proc. ACM/IEEE Symp. Architectures Netw. Commun. Syst. (ANCS)*, May 2017, pp. 131–141.

[17] B. Farinholt, M. Rezaeirad, P. Pearce, H. Dharmdasani, H. Yin, S. L. Blond, D. McCoy, and K. Levchenko, "To catch a ratter: Monitoring the behavior of amateur darkcomet rat operators in the wild," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 770–787.

[18] E. Stinson and J. C. Mitchell, "Characterizing bots' remote control behavior," In *Proc. 4th Int. Conf. Detection Intrusions Malware, Vulnerability Assessment, (DIMVA)*, vol. 7. Berlin, Germany: Springer-Verlag, 2007, pp. 89–108.

[19] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. Song, and H. Yin, "Automatically identifying trigger-based behavior in malware," in *Botnet Detection. Advances in Information Security*, vol. 36, W. Lee, C. Wang, and D. Dagon, Eds. Boston, MA, USA: Springer, 2008, pp. 65–88.

[20] W. Cui, R. Katz, and W.-T. Tan, "BINDER: An extrusion-based break-in detector for personal computers," in *Proc. USENIX Annu. Tech. Conf. (ATEC)*, Berkeley, CA, USA, vol. 5, Apr. 2005. pp. 363–366.

[21] F. Y. W. Law, K. P. Chow, P. K. Y. Lai, and H. K. S. Tse, *A Host-Based Approach to BotNet Investigation*. Berlin, Germany: Springer, 2010, pp. 161–170.

[22] M. Mimura, Y. Otsubo, and H. Tanaka, "Evaluation of a brute forcing tool that extracts the RAT from a malicious document file," in *Proc. 11th Asia Joint Conf. Inf. Secur. (AsiaJCIS)*, Aug. 2016, pp. 147–154.

[23] K. Muthumanickam and E. Ilavarasan, "P2P Botnet detection: Combined host- and network-level analysis," in *3rd IEEE Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2012, pp. 26–28.

[24] Y. Zeng, X. Hu, and K. G. Shin, "Detection of botnets using combined host- and network-level informati," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Networks (DSN)*, Jun. 2010, pp. 291–300.

[25] R. S. Abdullah, M. F. Abdollah, Z. A. M. Noh, M. Z. Mas'ud, S. Sahib, and R. Yusof, "Preliminary study of host and network-based analysis on p2p botnet detection," in *Proc. Int. Conf. Technol., Inform., Manage., Eng. Environ. (TIME-E)*, Bandung, Indonesia, Jun. 2013, pp. 105–109.

[26] R. Bapat, A. Mandya, X. Liu, B. Abraham, D. E. Brown, H. Kang, and M. Veeraraghavan, "Identifying malicious botnet traffic using logistic regression," in *Proc. Syst. Inf. Eng. Design Symp. (SIEDS)*, Apr. 2018, pp. 266–271.

[27] S. Khanchi, N. Zincir-Heywood, and M. Heywood, "Streaming botnet traffic analysis using bio-inspired active learning," in *Proc. IEEE/IFIP Network Oper. Manage. Symp. (NOMS)*, Apr. 2018, pp. 1–6.

[28] M. Oulehla and Z. K. Oplatková, and D. Malanik, "Detection of mobile botnets using neural networks," in *Proc. Future Technol. Conf. (FTC)*, Dec. 2016, pp. 1324–1326.

[29] D. Gavriluţ, M. Cimpoeşu, D. Anton, and L. Ciortuz, "Malware detection using machine learning," in *Proc. Int. Multiconf. Comput. Sci. Inf. Technol.*, Oct. 2009, pp. 735–741.

[30] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Inf. Sci.*, vol. 278, pp. 488–497, Sep. 2014.

[31] P. SangitaB and S. R. Deshmukh, "Use of Support Vector Machine, decision tree and Naive Bayesian techniques for wind speed classification," in *Proc. Int. Conf. Power Energy Syst.*, Dec. 2011, pp. 1–8.

[32] S. Garg, A. K. Singh, A. K. Sarje, and S. K. Peddoju, "Behaviour analysis of machine learning algorithms for detecting P2P botnets," in *Proc. 15th Int. Conf. Adv. Comput. Technol. (ICACT)*, Sep. 2013, pp. 1–4.

[33] P. Barthakur, M. Dahal, and M. K. Ghose, "A framework for P2P botnet detection using SVM," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery*, Oct. 2012, pp. 195–200.

[34] D. Xhemali, J. C. Hinde, and G. R. Stone, "Naïve Bayes vs. Decision Trees vs. neural networks in the classification of training Web pages," *Int. J. Comput. Sci. Issues*, vol. 4, no. 1, pp. 16–23, 2009.

[35] *Last Visited*. (2018). [Online]. Available: https://www.ll.mit.edu/ideval/data/1999data.html

[36] *last visited*. (Jan. 2019). [Online]. Available: http://www.egcert.eg/

[37] A. J. Aviv and A. Haeberlen, "Challenges in experimenting with botnet detection systems,"in *Proc. 4th Conf. Cyber Secur. Experimentation Test, (CSET)*, Berkeley, CA, USA, 2011, p. 6.

[38] P. E. Berg, K. Franke, and H. T. Nguyen, "Generic feature selection measure for botnet malware detection," in *Proc. 12th Int. Conf. Intell. Syst. Design Appl. (ISDA)*, Nov. 2012, pp. 711–717.

[39] J. Goebel and T. Holz, "Rishi: Identify bot contaminated hosts by irc nickname evaluation," in *Proc. USENIX Workshop Hot Topics Understand. Botnets (HotBots)*, Apr. 2007, p. 8

[40] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *proc. 16th Annu. Network Distrib. Syst. Secur. Symp.*, 2008, pp. 1–12.

[41] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.

AHMED A. AWAD received the B.Sc. and M.Sc. degrees in electronics, communications, and computer engineering from the Faculty of Engineering, Helwan University, in 2011 and 2018, respectively. He is currently pursuing the Ph.D. degree with Tennesse Tech University, USA. He is currently a Teaching Assistant with Helwan University, involved in many of its research projects. His research interests include malware analysis, data mining, algorithms and data structure, and botnets.

SAMIR G. SAYED received the B.S. and M.Sc. degrees from the Department of Electronics and Engineering, Helwan University, Egypt, in 1996 and 2003, respectively, the Ph.D. degree in electronic and electrical engineering from the University College London (UCL), U.K., in 2010. Since 2014, he has been an Honorary Lecturer with UCL. Since 2011, he has been the Director of the Malware and Reverse Engineering Department, Egyptian Computer Emergency and Readiness Team (EG-CERT). His research interests include cyber security, malware analysis, and wireless networks. In 2019, he will be promoted as an Associate Professor with the Electronics, Communications, and Computer Engineering Department, Helwan University, Cairo, Egypt.

SAMEH A. SALEM received the B.Sc. degree and the M.Sc. degree in communications and electronics engineering from Helwan University, Cairo, Egypt, in 1998 and 2003, respectively, the Ph.D. degree in engineering from the Department of Electrical Engineering and Electronics, The University of Liverpool, U.K., in 2008. In 2008, he was appointed as an Assistant Professor with the Department of Electronics, Communication, and Computer Engineering, Faculty of Engineering, Helwan University, Egypt. He is also selected to be a Coordinator and an Academic Advisor with the Department of Communication and Information Technology, Uninettuno University, Italy, incorporation with the Faculty of Engineering, Helwan University, Egypt. He is reviewing several proposals and research projects at the National Telecommunication Regulatory Authority (NTRA), Egypt. He is the Postgraduate Coordinator between the Faculty of Engineering-Helwan University and the Faculty of Engineering Sciences at Sinai University. In 2014, he was promoted to be an Associate Professor, and an Honorary Research Fellow with the Department of Electrical Engineering & Electronics, The University of Liverpool. He is currently a Consultant with the Egyptian Computer Emergency Response Team (EG-CERT) and the Head of Electronics, Communications, and Computer Engineering Department, Helwan University, Cairo, Egypt. His research interests include cyber security, malware analysis, clustering algorithms, machine learning, data mining, parallel computing, and cloud computing.