An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things

Nour Moustafa, *Member, IEEE,* Benjamin Turnbull, *Member, IEEE*, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*—Internet of Things (IoT) plays an increasingly significant role in our daily activities, connecting physical objects around us into digital services. In other words, IoT is the driving force behind home automation, smart cities, modern health systems, and advanced manufacturing. This also increases the likelihood of cyber threats against IoT devices and services. Attackers may attempt to exploit vulnerabilities in application protocols, including Domain Name System (DNS), Hyper Text Transfer Protocol (HTTP) and Message Queue Telemetry Transport (MQTT) that interact directly with backend database systems and client–server applications to store data of IoT services. Successful exploitation of one or more of these protocols can result in data leakage and security breaches. In this paper, an ensemble intrusion detection technique is proposed to mitigate malicious events, in particular botnet attacks against DNS, HTTP, and MQTT protocols utilized in IoT networks. New statistical flow features are generated from the protocols based on an analysis of their potential properties. Then, an AdaBoost ensemble learning method is developed using three machine learning techniques, namely decision tree, Naive Bayes (NB), and artificial neural network, to evaluate the effect of these features and detect malicious events effectively. The UNSW-NB15 and NIMS botnet datasets with simulated IoT sensors' data are used to extract the proposed features and evaluate the ensemble technique. The experimental results show that the proposed features have the potential characteristics of normal and malicious activity using the correntropy and correlation coefficient measures. Moreover, the proposed ensemble technique provides a higher detection rate and a lower false positive rate compared with each classification technique included in the framework and three other state-of-the-art techniques.**

***Index Terms*—Botnet, ensemble learning, Internet of Things (IoT), network intrusion detection system (NIDS), statistical flow features.**

Manuscript received May 11, 2018; revised June 24, 2018 and August 31, 2018; accepted September 16, 2018. Date of publication September 24, 2018; date of current version June 19, 2019. The work was supported in part by the SEIT–UNSW Canberra projects coded OP001Z6300-PS45716 and PS47084, and the Cloud Technology Endowed Professorship. *(Corresponding author: Kim-Kwang Raymond Choo.)*

N. Moustafa and B. Turnbull are with the Australian Centre for Cyber Security, University of New South Wales, Canberra Campus, Canberra, ACT 2600, Australia (e-mail: nour.moustafa@unsw.edu.au; b.turnbull@adfa.edu.au).

K.-K. R. Choo is with the Department of Information Systems and Cyber Security and the Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/JIOT.2018.2871719

I. Introduction

CURRENTLY, technologies underpinning the Internet of Things (IoT) are increasingly used for providing online services to users and public-/private-sector organizations. The rapid evolution of the manufacturing domain to incorporate these technologies into operation, for example, has increased the integration of physical and digital systems. This integration has seen an increase in the deployment of sensors, actuators, automatic identification, and wireless communications across many sectors, and are the underpinning technologies for smart houses, offices, cities, and nations, as well as modern healthcare systems and advanced manufacturing. For example, the potential process of reading temperature and humidity across cities as IoT solutions depend on sensors configured and deployed to publish to external systems over the Internet. This leads to a substantial increase in pervasive computing resources due to the users' dependence on the resources of Web client systems, Web server appliances, database systems and software they interact with.

IoT services operate via network protocols, such as Domain Name System (DNS), Hyper Text Transfer Protocol (HTTP), and Message Queue Telemetry Transport (MQTT), in the TCP/IP model. Network systems are partitioned into four layers, with each layer comprising different protocols that communicate with their counterpart in other networks [1]. When deploying IoT services, middleware tools, such as Node-Red, are connected to physical devices, such as temperature and light sensors, where data generated from these sensors are transmitted to IoT hubs, such as IoT Amazon Web Services (AWS), over TCP/IP. The MQTT protocol is often used for publishing and/or subscribing messages received from low bandwidth devices (e.g., sensors) across untrusted networks [2]. MQTT operates over TCP for sending and receiving the middleware's data to IoT hubs.

The DNS and HTTP protocols are two fundamental services for IoT applications, as they transact directly with user data. DNS is a hierarchical distributed system connected to the Internet or a network for mapping domain names, which can be simply memorized by users, to numeric IP addresses using authoritative name servers for each domain. The purpose of the DNS protocol is to prevent the conflict of the reserved domain names and facilitate computer services and devices. The HTTP protocol exchanges and transfers Hypertext, a structured textual format that includes hyperlinks to other

hypertext documents. HTTP uses a request–response mechanism and relies on the client–server paradigm. A client submits a HTTP request message to the server, and the server returns a response message to the clients, providing resources, such as HTML files.

Attackers generally seek to identify and exploit vulnerabilities and limitations in these protocols, for example using various deception approaches. For instance, attackers create untrustworthy accounts to target users who have limited experience in registering their information within Internet applications. Cyber attackers can breach weak points of websites and services by a range of exploitation techniques, such as polymorphic code, DNS spoofing, DNS cache poisoning, denial of service (DoS), distributed DoS (DDoS), exploits and URL interpretation. A botnet is one of the major threats that targets networks via different protocols, in particular MQTT, DNS, and HTTP, by exploiting target hosts using the preceding exploitation techniques. A bot mechanism is a self-propagating malware that infects compromised clients, developed to perform intrusive tasks after being triggered. The infected bot network is known as a botnet that is under the remote control of a master (i.e., botmaster), where bots receive instructions from the master across command and control (C&C) channel for executing malicious techniques involving DDoS and phishing.

In order to defend against IoT cyber-attacks, this paper presents a Network Intrusion Detection System (NIDS) based on an AdaBoost ensemble learning algorithm that takes statistical flow features as input for recognizing malicious botnet activities. The statistical features are established from aggregating network flows with the potential analysis of MQTT, HTTP, and DNS protocols. An NIDS is a technique designed for detecting malicious activities of network traffic using a set of features extracted from the layers of the TCP/IP model and a decision-making method of attack detection. Designing an effective NIDS requires a data source that comprises a set of features for evaluating its performance while classifying normal and attack instances using a decision-making method. This is one of the challenges that we try to address in this paper.

A set of features is proposed based on the flow identifiers of network packets, MQTT, DNS, and HTTP information for identifying malicious behaviors, which exploit IoT applications within these protocols. The data sources of the protocols are extracted from the source files of the UNSW-NB15 [3], [4] and NIMS botnet [5] datasets, as well as network traffic of temperature, humidity, and light sensors connected to the IoT AWS hub [6] to evaluate the impact of the proposed features using statistical and machine learning techniques. The Bro-IDS tool [7] is used for capturing the basic flows and fundamental information of the protocols. Also, a new extractor module is designed, which operates simultaneously with the Bro-IDS tool to generate additional statistical features of the transactional flows that cannot be generated using existing flow analysis tools.

There are several well-known flow analysis techniques and tools used to collect network data, including NetFlow, sFlow, and IPFIX, in addition to others proposed by academia [8]. However, there is a common limitation across all of these; each is only capable of aggregating one attribute at a time. This increases network's overheads when running network analysis [9]. Our extractor module is designed for accumulating more than one attribute of network flows at a time. For example, it is capable of collating the number of source/destination IPs and ports for each 10/100 network connections, as explained in Section III. The statistical measures of correntropy and correlation coefficient (CC) are applied to estimate the similarity and strength of the proposed features, respectively. Additionally, for detecting malicious instances, the AdaBoost ensemble learning methodology is used to combine three classification techniques of DT [10], NB [11], and ANN [11] for detecting those malicious instances, improving the performance of NIDS.

The key contributions in this paper are provided as follows.

1) A set of features is proposed from the MQTT, DNS, HTTP protocols and their flow identifiers to build an effective NIDS for detecting attacks that breach network applications within these protocols.
2) New data sources are generated for the protocols using the proposed features from the UNSW-NB15 and NIMS botnet datasets and network traffic of sensor linked with the IoT hub for evaluating their effects on classifying normal and suspicious instances.
3) An AdaBoost ensemble technique with an in-depth statistical analysis is suggested in order to evaluate the effect of our proposed features for effectively identifying malicious events.

The rest of this paper is organized as follows. Section II discusses the background and previous studies related to this paper. The details of the proposed statistical flow features and the used tools for extracting these features are explained in Section III. In Section IV, a framework is proposed, based on the AdaBoost ensemble learning for identifying malicious activity. Section V provides the experimental results and discussions of our proposed features and ensemble learning framework. Finally, Section VI concludes this paper.

II. Background and Related Work

The goal of this paper is to build an effective and adaptive NIDS for identifying malicious activities that attempt to exploit IoT services. The background and previous studies related to this paper are explained below.

A. NIDS Techniques

An NIDS is widely used for monitoring and detecting malicious activities from network traffic. A typical NIDS comprises four steps, namely: a data source, data preprocessing, decision-making method, and defense response [12]. First, a data source includes a set of network observations, each of which includes features used for distinguishing between legitimate and suspicious observations. Second, a data preprocessing which prepares input data by eliminating unnecessary features to create a set of patterns involves the properties of legitimate and suspicious activities. Third, a detection method includes a classification technique that identifies abnormal

observations. Finally, a defense response is a decision taken by software or cyber administrators to prevent attack actions.

The methodology of NIDS is classified into misuse-based and anomaly based and a hybrid of the two. To start with, a misuse-based NIDS monitors network traffic for matching observed instances against a well-known blacklist. Even though it produces higher detection rates and lower false positive rates (FPR), it cannot identify zero-day attacks. Moreover, it demands a significant and ongoing effort to update the blacklist with new rules of attacks based on previously discovered attacks. Conversely, an anomaly based NIDS constructs a normal profile and considers any variation from this profile as an attack. Since it can identify both existing and zero-day attacks and does not demand any effort to create rules, it is a more effective mitigation system than a misuse-based IDS if its decision-making method is efficiently designed, as proposed in this paper.

Many researchers have employed ensemble approaches in order to improve the performance of NIDSs. The goal is to correlate the alerts for decreasing the alarms, helping network security administrators to manage these alerts easily. Giacinto and Roli [13] designed an ensemble technique for recognizing different attack types using a set of features. In this proposal, the final decision of attack detection depends on the voting rule method. Chebrolu *et al.* [14] developed an ensemble technique of Bayesian networks (BN) and classification and regression trees to identify attack instances. Overall, the empirical results of these techniques showed that the overall performances of hybrid/ensemble techniques are better than each one individually, although often increasing their computational overhead.

B. NIDS for TCP/IP Protocols and IoT Technologies

The performance evaluation of any NIDS demands a data source that includes a set of network features, which are categorized into four types based on the classification purposes, namely; source and destination port numbers-based features, payload-based features, behavior-based features, and flow-based features.

Source and destination port numbers-based features are extracted using the Coral-Reef and Netflow tools. These features are ineffective because they only acquire simple information from packet headers that are highly untrustworthy in contemporary network environments to identify attacks with the dynamic changes and rapid speed of current networks. Payload-based features capture substantial signatures of well-known applications. These features can help in detecting malicious activities, providing high detection accuracy. These are also independent of the ports used, which allows for detection even when using nonstandard ports. However, these features require a huge effort to regularly update signatures of attack types, with the difficulty of capturing them from high network traffic rates. Behavior-based features are features which elicit interactions between hosts in terms of destinations, ports and behavior patterns of hosts. Finally, flow-based features capture the basic flow identifiers (i.e., source–destination IP and ports as well as protocols) and statistical features,

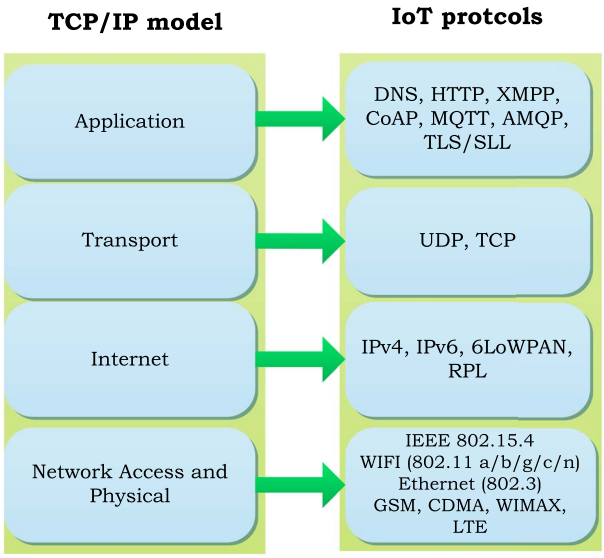

Fig. 1. TCP/IP model and IoT protocols.

such as interarrival times and packet sizes. The last two types achieve high detection accuracy if they are precisely applied, as suggested in this paper with proposing aggregating more than one attribute in the proposed extractor module in order to detect botnet schemes that happen by flooding massive malicious flows to the compromised hosts.

IoT technologies utilize the four layers of the TCP/IP model with adding new protocols for integrating physical and network objects. As shown in Fig. 1, The physical layer connects each device to the network (for example via CAT6 cabling, or wirelessly via IEEE 802.11 a/b/g/n). The Internet layer relates to logical addressing, such as IPv4 and IPv6 that is widely adopted for IoT appliances. The transport layer facilitates communication between end-to-end systems and interacts with the application layer, where HTTP and MQTT operate over TCP, whilst DNS operates primarily via UDP.

Several research studies have been conducted to construct data sources from TCP/IP protocols for evaluating NIDS performance, specifically focusing on the recognition of several types of botnets and exploitation mechanisms, such as DoS, DDoS, and phishing. For example, Marchal *et al.* [15] developed a scalable and distributed IDS via the collection of network instances from honeypot, DNS, HTTP, and IP-flow data. However, they did not provide the features used for implementing this system. Additionally, the simulated data was collected from different systems without publishing the configuration environment for evaluating the performance of new NIDSs.

Recent research studies in NIDS-based IoT solutions have focused on selecting flow features for discovering cyber attacks and botnets from IoT networks. For example, Teixeira *et al.* [16] proposed a technique for detecting insiders attacks in IoT systems through the examination of network traffic at each IoT node, and Cao *et al.* [17] suggested an NIDS for recognizing ghost attacks on ZigBee-based IoT systems. Azmoodeh *et al.* [18] developed a deep learning technique to identify malware of Internet of Battlefield Things by analyzing

their operational code. Chen *et al.* [19] developed an adaptive management framework for IoT networks used to identify cyber attacks with low human interference. Pajouh *et al.* [20] developed a two-layer dimension reduction technique and two-tier classification mechanism for detecting malicious events from IoT backbone networks.

Choi *et al.* [21] suggested a bot detection technique via the analysis of DNS flows based on generating features of DNS queries. This paper was built on extracting static information from DNS queries, such as source IP addresses and the query domain name. However, these can be hidden or altered through the use of virtual private networks with no analysis of aggregated statistical network flows that involve the potential characteristics of cumulative flooding threats, such as DDoS attacks [22]. Wang *et al.* [8] proposed a domain name generation algorithm for detecting botnets through the inspection of individual DNS queries. Although these studies utilize existing flow tools and techniques and evaluated their results through the use of machine learning techniques, the authors did not use the aggregated flows that efficiently identify botnet threats in the decision-making method. They also did not seek to determine the effective features of these protocols, as suggested in this paper.

III. Proposed Statistical Flow Features

This sections describes the tools used for extracting the proposed statistical flow features from network traffic of IoT systems. The feature generation and evaluation of the TCP/IP model, in particular MQTT, DNS, and HTTP protocols, are also discussed.

A. Tools of Statistical Feature Generation

Within this paper, three modules of testbed configuration, IoT sensors and feature generation tools have been designed and developed, in order to generate the proposed statistical flow features from network traffic in an IoT network. As shown in Fig. 2(a), the testbed configuration module of the UNSW-NB15 dataset is presented as an example to demonstrate how attack and normal observations were generated by capturing network traffic transmitted and/or received through two ingress routers. Similarly, this module is designed for the NIMS dataset for extracting statistical flows from its raw packets (i.e., pcap files).

To generate benign network traffic from sensors, we develop the IoT sensors module as depicted in Fig. 2(b). A node-red middleware was installed on a Raspberry Pi, running the Raspbian Operating System for simulating temperature, humidity, and ambient light sensor traffic. Node-Red allows for the connection to the AWS IoT hub via an MQTT broker. The Node-Red flow editor is configured with MQTT clients in order to subscribe and publish the data generated from the sensors to the MQTT server of the AWS IoT hub. While sending/receiving the sensors' data to/from the IoT hub, the tcpdump tool, a packet analyzer, was used to capture the raw network traffic.

Module C of Fig. 2 outlines the feature generation tools for extracting network traffic from the datasets, IoT networks and sensors. The implementation of this module uses three tools, namely: tcpdump, Bro-IDS, and our extractor module. The processing of data for feature extraction is as follows. first, the tcpdump tool is used for capturing network traffic in pcap files, then, the Bro-IDS tool analyzes the generated pcap and develops the flow-based features in addition to collecting general information about TCP/IP protocols, which are then stored in log files. The HTTP log file contains the request–response pairs and all appropriate metadata about the protocol, whereas the DNS log file contains the DNS queries associated with their responses. The data length of the MQTT data (i.e., feature 7 in Table I) was computed by inspecting the TCP transport layer of the TCP/IP model. The files were logged in a MySQL database to make it easier to generate additional statistical features and labeling the instances, either normal or attack, as discussed below.

Additionally, a new extractor module has been developed to generate additional statistical features from the stored features in the database. While the extracted features of the Bro-IDS are stored in the database, the extractor module simultaneously runs in order to generate the statistical features. This configuration shows that the proposed features can be simply established in a real-world IoT network. To illustrate the details of the extractor module, a given data source has a set of vectors ($V = \{v_1, v_2, \ldots, v_N\}$) and each vector includes a set of features ($F = \{f_1, f_2, f_2, \ldots, f_D\}$), where N is the number of vectors and D is the number of features. The mean ($\bar{f}$) and length ($\lambda(f)$) of feature values are computed using (1) and (2), respectively,

$$\bar{f} = \frac{\sum_{n=1}^{N} f_n}{N} \tag{1}$$

$$\lambda(f) = \sum_{n=1}^{N} C(f_n). \tag{2}$$

In the above two equations, $C(f_n)$ is the length of feature values, and both equations are used to generate the statistical features. As statistical behaviors cannot be computed from one record only, the statistics for each 100 observations are applied and ordered sequentially, with respect to the time session of packets. The 100 record connections could be used to identify the patterns of normal and attack activities, as well as efficiently define the correlations between these observations. As a result, the majority of classification techniques use the statistical features as input for distinguishing legitimate and suspicious observations. The steps of designing the extractor module are provided in Algorithm 1. For each 100 records stored in the database, the length, rate, and/or mean of categorical feature values, for example, *query, answer, method*, and *host*, are calculated to generate the new statistical features.

Table I lists the proposed features that are generated from the Bro-IDS tool and extractor module. The features created by the novel extractor module are the transactional flow features numbered from 7 to 12, the features numbered from 18 to 26 and the features numbered from 33 to 36, whilst the rest of the features were extracted by the Bro-IDS tool.

Due to the significant roles of specific protocols in IoT services; specifically MQTT, DNS, and HTTP, statistical

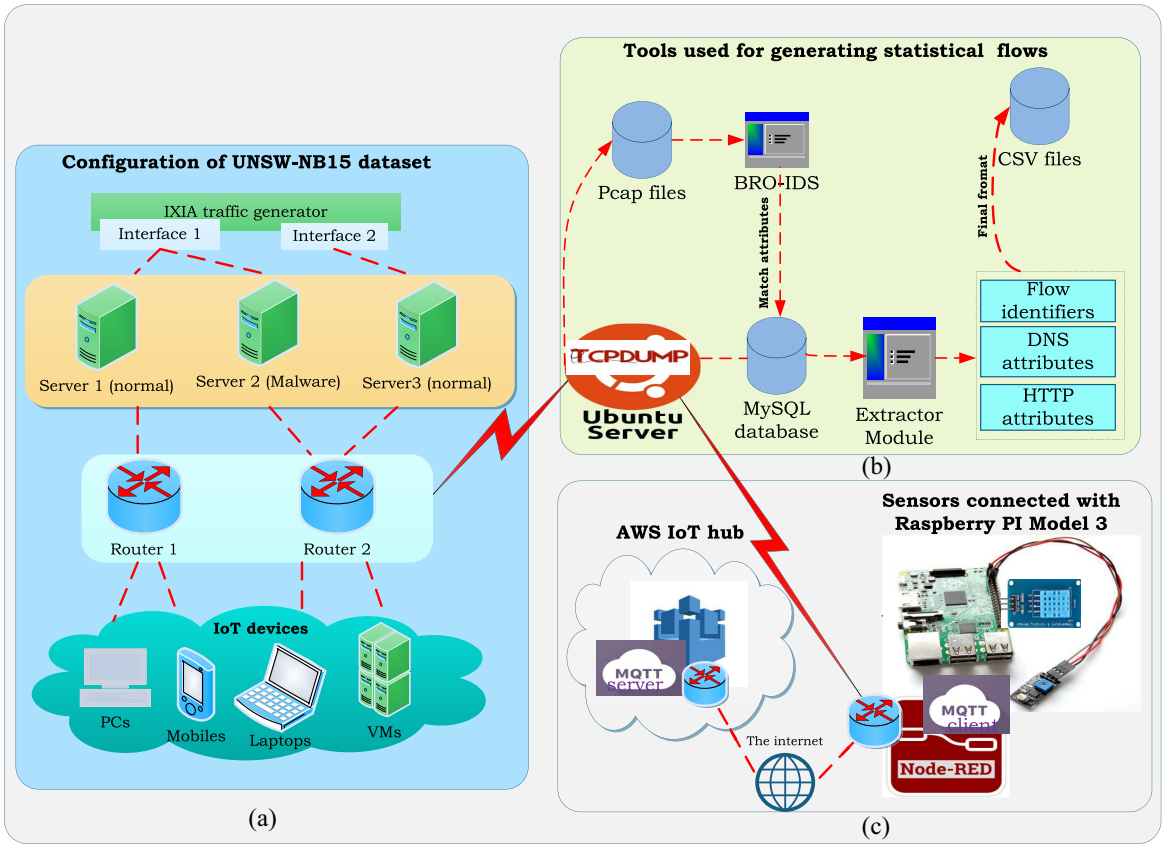

Fig. 2. Testbed and tools used for generating statistical flow features.

Algorithm 1 Generating Statistical Flow Features

Input: $f[N][D]$, $n = 1$, $rate = 0$ // N is the number of vectors, D is the number of features, and n is a counter of the similar feature values that starts with an element in the data source of $f[N][D]$

Output: $f_n[]$, $f_{rate}[]$ //arrays to store features

for $i = 1$ to N **do**
  for $j = 1$ to D **do**
    if $(f[i][j] == f[i][j+1])$ **then**
      $n++$
    end if
  end for
  $f_n[i] = n$
  $j = m$ // m is number of records (e.g., 100 records)
  $rate = n/m$
  $f_{rate}[i] = rate$
end for
return (f_n, f_{rate})

features are proposed for building an effective NIDS using the AdaBoost ensemble learning methodology as discussed in Section IV.

The proposed features consist of flow-based and service-based features, as depicted in Fig. 3. The flow-based features of the TCP/IP protocols are extracted from raw packets to elicit only the flow identifiers and their transactional and MQTT features, involving the intrinsic information of normal and abnormal activity. These features are widely used in the online analysis of network systems, which provide a higher detection rate (DR) using machine learning algorithms when values of legitimate observations differ from suspicious ones [23].

The flow-based features contain source–destination IP addresses and ports as well as protocol types, in addition to the transactional features, which include flow statistics, such as the data length. The header information of raw packets was captured by the tcpdump tool for sequentially storing the flow identifiers. The transactional features are generated from the interactions of the flow identifiers depending on the packet session times. They are created based on the time window of a certain number of receiving packets (i.e., feature 6 in Table I) for keeping the online detection of malicious activity from IoT networks.

The service-based features comprise intrinsic information of the DNS and HTTP protocols which are acquired from the application layer of the TCP/IP model to deal directly with information of sensor and network data included in the protocols. First, the DNS features are extracted from the common DNS query responses and domain names. Moreover, statistical features are generated. These include the length and mean of the *query* and *answer* attributes. Second, the HTTP features are created from analyzing the HTTP requests and responses, and generating statistical features

TABLE I
PROPOSED STATISTICAL FEATURES FOR IOT NETWORKS

No.	Name	Type	Description
			Flow Features
1	srcip	N	Source IP address
2	port	I	Source port number
3	dstip	N	Destination IP address
4	dsport	I	Destination port number
5	proto	N	Protocol type
6	ltime	T	Last time of connection
			Transactional and MQTT Features
7	len_mqtt	I	Length of MQTT's data computed from the TCP protocol
8	ct_dst_ltm	I	Number of connections to the same destination (3) in 100 records according to the ltime (6)
9	ct_src_ ltm	I	Number of connections of the same source (1) in 100 records according to the ltime (6)
10	ct_src_dport_ltm	I	Number of connections of the same source address (1) and the destination port (4) in 100 records according to the ltime (6)
11	ct_dst_sport_ltm	I	Number of connections to the same destination (3) and the source port (2) in 100 records according to the ltime (6)
12	ct_dst_src_ltm	I	Number of connections of the same source (1) and the destination (3) in 100 records according to the ltime (6)
			DNS Features
13	query	N	Domain name subject of the query
14	q_type	I	Value specifying the query type
15	answers	N	List of resource descriptions in answer to the query
16	ttls	I	Caching intervals of the answers
17	len_qry	I	Length of the query (12)
18	len_ans	I	Length of the answers (15)
19	mean_class	F	Mean of the class in 100 records according to the ltime (6)
20	mean_type	F	Mean of the type in 100 records according to the ltime (6)
21	ct_src_qry	I	Number of the srcip (1) and the query (12) in 100 records according to the ltime (6)
22	avg_ttl_ltm	F	Average value of the ttls (16) values in 100 records according to the ltime (6)
23	ct_dm_src	I	Number of domains (12) to same srcip (1) in each 100 records according to the last time (6)
24	ct_dm_dstip	I	Number of domains to same dstip (3) in each 100 records according to the ltime (6)
25	dm_ratio	F	The percentage of the domain in each 100 records according to the ltime (6)
			HTTP Features
26	method	N	HTTP Request Method e.g., GET, POST, HEAD
27	host	N	Value of the HOST header
28	trans_depth	I	Pipelined depth into the connection
29	url	N	URI used in the request
30	req_body_len	I	Actual uncompressed content size of the data transferred from the client
31	res_body_len	I	Actual uncompressed content size of the data transferred from the server
32	len_host	I	Length of the HOST header value
33	len_url	I	Length of the URL
34	ratio_mth_host	F	Percentage of the method (26) with same the host (27) in each 100 records according to the ltime (6)
35	ratio_mth_url	F	The percentage of the method (26) with same the url (29) in each 100 records according to the ltime (6)
36	ratio_host_url	F	Percentage of the host (27) with same the url (29) in each 100 records according to the ltime (6)

Type- I: Integer, F: Float, N: Nominal, T: Timestamp

from the basic information of interacting clients and servers, such as the length and mean of the *method* and *URL* attributes.

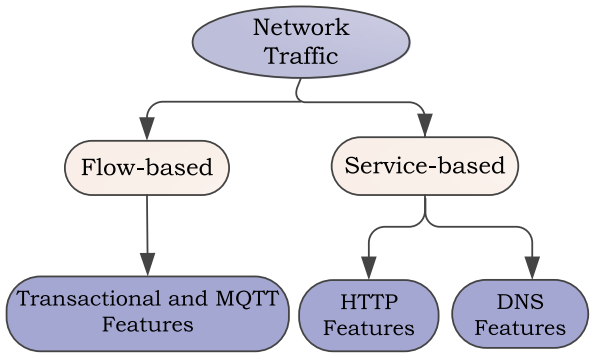

Fig. 3. Proposed features for IoT networks.

It is empirically observed that the statistical features can define the basic information of the protocols that have the patterns of legitimate and malicious instances. This is because that statistical features can aggregate traffic flows with removing redundant observations that occur with flooding attacks, such as DoS and DDoS [24]. These types of attacks send a massive amount of flows to disrupt resources of compromised hosts. Moreover, the transactional features can assist the NIDS in discriminating the potential differences between normal and abnormal activities, as they include information about more than one attribute, such as source/destination IPs and ports for each particular time, and they cannot be generated using the existing flow exporters. These features can be easily created while running a real NIDS; hence, they can assist in designing a reliable NIDS if its decision engine is correctly designed.

B. Correntropy Measure for Evaluating Proposed Features

The correntropy measure is used for estimating the similarities of the proposed feature vectors, where this measure can precisely identify the difference between normal and attack instances. If there is a clear dissimilarity between these instances, then it will be statistical evidence that demonstrates the importance of these features for identifying malicious activity. This measure is one of the second-order statistics and a nonlinear similarity criterion for defining the interactions of given feature observations. Hence, it is less sensitive to outliers than the mean square error. Given two random variables f_1 and f_2, their correntropy is computed by

$$V_\sigma(f_1, f_2) = E\big[K_\sigma(f_1 - f_2)\big] \tag{3}$$

such that $E[.]$ denotes to the mathematical expectation, $K_\sigma(.)$ refers to the Gaussian kernel function, and σ is the kernel size, as defined by

$$K_\sigma(.) = \frac{1}{\sqrt{2\Pi\sigma}} \exp\left(-\frac{(.)^2}{2\sigma^2}\right). \tag{4}$$

Mathematically speaking, the joint probability density function $P_{F_1,F_2}(f_1, f_2)$ is usually unidentified while a finite number of observations $\{fi, fj\}_{i,j=1}^{M}$ can be obtainable. Consequently, the correntropy in (3) is calculated by

$$\hat{V}_{M,\sigma}(A, B) = \frac{1}{M} \sum_{i,j=1}^{M} K_\sigma\big(f_i - f_j\big). \tag{5}$$

TABLE II
EXAMPLE FOR COMPUTING CORRENTROPY TO THREE FEATURES

f_1	f_2	f_3	*Correntropy*
16	12	2	0.024
7	5	1	-0.045
23	21	2	0.022

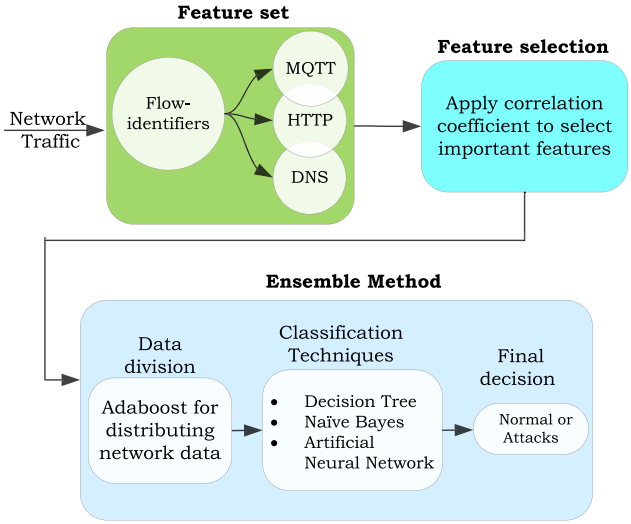

Fig. 4. Proposed framework for detecting cyber-attacks from IoT networks.

For applying the correntropy measure to multivariate network data, as defined in (6), it is estimated for both normal and abnormal feature vectors.

$$I_{1:N} = \begin{bmatrix} f_{11} & f_{12} \ldots \\ f_{21} & f_{22}\, f_{ij} \end{bmatrix} \quad Y_{1:N} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}. \tag{6}$$

In the above equation, I is the observations of network data, Y is the class label (c) of each observation, N is the number of observations, and F is the feature set.

If there is a difference between the normal and suspicious vector values, then it will be significant evidence that ensures the importance of our proposed features. Table II lists an example for computing the correntropy to three features (f_1, f_2, and f_3) using (4) and (5). The correntropy for each vector is computed for the normal and attack instances, and then they are plotted to reveal that there are differences between these instances as explained in Section V-C.

IV. PROPOSED ENSEMBLE LEARNING FRAMEWORK-BASED NIDS FOR IOT NETWORKS

A proposed ensemble-learning framework is presented in order to detect botnet attacks that expose IoT networks via the TCP/IP protocols by analyzing the MQTT, DNS, and HTTP protocols, as shown in Fig. 4. This framework consists of three main steps, namely: a feature set, feature selection, and ensemble method. To begin with, a feature set contains the proposed features listed in Table I. Then, the CC is utilized for selecting the lowest correlated features that have the potential characteristics of legitimate and malicious patterns, as discussed in Section IV-A. Finally, the ensemble method is the technique used for classifying normal and suspicious instances. In this method, three machine learning techniques of Naive Bayes (NB), decision tree (DT) and artificial neural network (ANN) are applied based on the AdaBoost classifier to distribute the data between these techniques according to an error function explained below.

A. Feature Selection Method

Feature selection plays a key role in NIDSs for selecting important features and removing unnecessary ones that can assist in discriminating legitimate and malicious instances, and improving the overall performance of any NIDS. The goal of feature selection is to reduce the computational cost of NIDS, remove information redundancy, enhance the accuracy of NIDS, and assist in analyzing the normality of network data. In this paper, the simplest feature selection method is used, and specifically the CC, which computes the strength degree between some features. The lowest N ranked features are selected as the most important ones that pass to the ensemble method for identifying abnormal behaviors of the DNS and HTTP instances.

The CC of the features f_1 and f_2 is calculated by

$$\mathrm{CC}(f_1, f_2) = \frac{\mathrm{cov}(f_1, f_2)}{\delta_{f_1}.\delta_{f_2}}$$

$$\mathrm{CC}(f_1, f_2) = \frac{\sum_{i=1}^{N}(a_i - M_{f_1})(b_i - M_{f_2})}{\sqrt{\sum_{i=1}^{N}(a_i - M_{f_1})^2}.\sqrt{\sum_{i=1}^{N}(b_i - M_{f_2})^2}}. \tag{7}$$

In the above equation, δ is the standard deviation of a feature, cov() is the covariance of features, and a_i and b_i, respectively, denote the values of f_1 and f_2. The mean of f_1 and f_2 are calculated via $M_{f_1} = [(\sum_i^N a_i)/N]$ and $M_{f_2} = (\sum_i^N b_i/N)$, respectively.

From (7), the outcomes of the CC vary in a fixed range of $[+1, -1]$. If the value is close to $+1$ or -1, then it refers to the strongest correlation of the two features f_1 and f_2. In contrast, if the value is near to 0, then it implies that there is no correlation between these features. A positive sign indicates that the trend of both features is in the same direction, whereas a negative sign indicates that the trends of both features are opposite.

B. Ensemble Method of Classification Techniques

Three machine learning techniques of DT, NB, and ANN are applied to classify normal and attack records. The techniques are implemented as an ensemble method with the AdaBoost mechanism for designing an adaptable NIDS, where each technique is considered a weak classifier and its findings are not high enough compared with the findings of the ensemble method.

In order to clarify why these classification techniques are chosen in the proposed ensemble method, the correntropy measure is a key factor in selecting them. Based on the results presented in Section V-C1, the correntropy showed that there is a slight difference between the feature vectors of normal and abnormal. This led to need to decide which of the classification techniques could be selected in the ensemble method based on the potential procedures of designing those techniques, such as distance- and probability-based learning. Let

there be two vectors v_1 (normal) and v_2 (attack) and the difference between them is very small (i.e., $v_1 - v_2 \approx 0$). Since each classification technique was designed based on a particular kernel function, such as a probability, weight or feature value itself, three types of functions are combined that can classify network data that has slight differences between its normal and suspicious observations.

Since the correntropy outcomes revealed that there are small variations between legitimate and suspicious vectors, the classification techniques should be chosen to classify these vectors with small differences. Consequently, the DT, NB, and ANN are selected because they can classify such vectors effectively. The ANN depends on weighting each feature, producing accurate linear class boundaries. There are several advantages for classifying network data using the ANN technique. It demands less formal statistical training and defines complex nonlinear correlations between dependent and independent variables. Moreover, it can identify all possible potential interactions between predictor variables. The NB relies on estimating the posterior probability that computes the entire possibilities of data distribution, finding the small differences between each class label in the training phase. It has many merits while recognizing abnormal observations. It requires less training data and linearly scales with the predictors and features' values. Furthermore, it is not complicated while optimizing its parameters. Ultimately, the DT uses the feature values themselves, where it divides the feature space into regions with mostly the same label. Hence, it can easily find the small differences between feature vectors. It has multiple advantages while classifying network data. It operates for selecting important features and can prepare learning data points easily as it deals directly with the values of features. Furthermore, there are nonlinear relations between its parameters that do not affect its performance.

The three techniques are briefly described below.

1) *DT:* Reference [10] It is a structural technique similar to a tree, which has leaves and branches. The leaves represent classification rules while branches symbolize features that lead to classifying data. The decision rule depends on if-then rules to classify data inputs. It can be used especially for solving multitype feature problems (e.g., string and real numbers). The C4.5 DT technique is used [55] because of its simplicity of implementation that establishes trees from the instances in the training phase using the information entropy principle. In the training phase, each sample comprises a d dimensional vector, including the feature values and the corresponding class label. At each node of the tree, the technique selects the feature, which most effectively splits samples into subsets using the information gain method. The feature with the highest information gain is selected to make the final decision.
2) *ANN [11]:* It is a technique that transforms inputs into outputs to match targets (i.e., the output of each class), generating a set of hidden layers. The execution steps depend on an activation function and weight to each neuron, which predicts outputs. In the methodology of NIDS, an ANN demands some information about the normal data class to systematically change the interconnection neurons to train the weights of the network and build a model that can distinguish between normal and abnormal network data. It uses an activation function that depends on a large number of input observations I, and its basic function is defined as

$$f(I) = \tau\left(\sum\nolimits_j W_j \cdot I_j\right) \tag{8}$$

such that $f(I)$ is the predicted output of the class label, τ is the sigmoid activation function, and W_j is a weight of each instance I_j.
3) *NB [11]:* It is a type of BN technique that assumes that all features are independent. The basic classification method of the NB is the Bayes rule theorem to search for the maximal likelihood hypothesis that identifies the class label. It can be used for predicting data points using the maximum posterior function as computed by

$$P(C|I) = \text{argmax}_{w\in\{1,2,..,N\}} P(C_w) \prod_{J=1}^{N} P\left(I_j|C_w\right) \tag{9}$$

such that C is the class label, I is the observation of each class, w is the class number, $P(C|I)$ is the probability of a given class and $\prod_{j=1}^{N} P(I_j|C_w)$ is the multiplication of all instance probabilities subject to their classes for achieving the maximum output.

The important selected features using the CC are used to train and validate these techniques. For distributing network data between the techniques, the AdaBoost model [25] is used. This allows us to establish many base learners by sequentially reweighing the observations in the training phase. Each observation incorrectly classified by the former base learner will get a higher weight in the next round of the training phase. The key concept underlying the boosting technique is to frequently utilize a base learner to changed forms in the training phase, creating a sequence of base learners for a selected number of iterations. More specifically, all the observations are initiated with equal weights, and then each iteration is used as a base learner to these weighted instances. To distribute the data, the weight of the correctly classified observations reduces while the incorrectly classified observations increase. The final model obtained by the boosting algorithm is a linear combination of several base learners weighted by their own performance.

In this paper, the AdaBoost technique [25] is used, which is the most widely applied to ensemble learning mechanisms for distributing input data on the machine learning techniques used. The flowchart and steps of the AdaBoost technique are presented in Fig. 5 and Algorithm 2, which explains how it was applied to the proposed flow features.

V. Experimental Results and Discussion

This section explains the DNS and HTTP data sources, as the MQTT's features included in the transactional features of these sources, and the evaluation metrics used in the experiments. Then, the statistical measures of the correntropy and correlation coefficients are discussed to demonstrate the

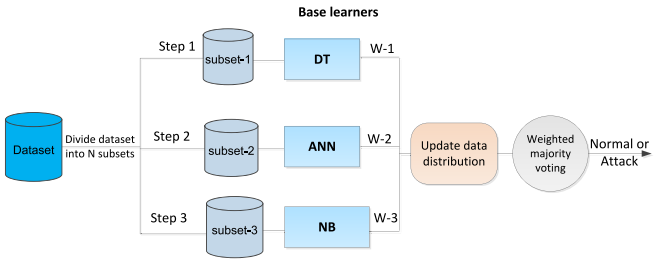

Fig. 5. AdaBoost flowchart.

Algorithm 2 Steps of AdaBoost Model

Input: Dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, L is base learning techniques, I is a number of iterations, and $D_i(j)$ is a parameter of the weight distribution

Steps:

1: $D_i(j) = 1/m$ // initialize the weight distribution
2: **for** i to T
3: $h_i = L(D, D_i)$ // train a base learner h_i from D using distribution D_i
4: $\epsilon_i = P_{i,D_i}[h_i(x_i \neq y_i)]$ // estimate the h_i error
5: $w_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$ // determine the h_i weight
6: $D_{i+1}(j) = \frac{D_i(j)}{Q_i} * \left\{ \begin{array}{l} \exp(-w_i) \ if h(x_i) = y_i \\ \exp(w_i) if h_i(x_i) \neq y_i \end{array} \right\}$// update the distribution, where Q_i is computed by (6)
7: $Q_i = \frac{D_i(j) \exp(-w, y_i, h_i(x_i))}{Q_I}$// normalization factor which enables $D_{i+1}$to be a distribution end
8: $H(x) = sign \sum_{i=1}^{I} w_i h_i(x)$
9: **end for**
8: **return** $H(x)$

impact of the proposed features for detecting malicious events. Finally, the findings of the ensemble learning framework and its techniques are illustrated.

A. DNS and HTTP Data Sources

The UNSW-NB15 [3], [4] and NIMS datasets [5] are used to evaluate our proposed framework, in addition to the network traffic of the simulated sensors explained in Section III. First, the proposed features of the DNS and HTTP protocols are generated from the datasets. The UNSW-NB 15 dataset contains several forms of data; pcap files, Argus files, Bro files, and CSV files for evaluating NIDS. It has a collection of authentic modern normal and abnormal network traffic (see Table III). The proposed features of both protocols and their flow-based features are elicited from the pcap files of the datasets using the tools elaborated in Section III-A. The labeling of their instances, classified as either "normal" or "attack," are implemented by matching the flow-based features with the flow-based feature of the attack report, published in [3].

The CSV files of the DNS and HTTP features are classified into eight botnet activity types based on the IXIA traffic generator reports [3], as discussed below.

1) *Analysis:* Attack methods which breach Internet applications via ports (e.g., port scans), e-mails (e.g., spam) and Web scripts (e.g., HTML files).

TABLE III
DATA DISTRIBUTION OF DNS AND HTTP

Record types	DNS data	HTTP data
Normal	345561	295689
DoS	130	1814
Exploits	689	15804
Fuzzers for suspicious activity	1008	1105
Generic	1563	1885
Reconnaissance	49	1983
Analysis	*0*	369
Backdoor	*0*	212
Worms	*0*	139

2) *Backdoor:* A technique of bypassing a stealthy normal authentication, securing unauthorized remote access to a device.
3) *DoS:* An attacker which seeks to disrupt the computing resources via memory for normal business disruption, by preventing authorized requests from accessing a device.
4) *Exploit:* A sequence of instructions that takes advantage of a glitch, bug or vulnerability, causing an unintentional or unsuspected behavior on a host or a network.
5) *Fuzzers for Suspicious Activity:* The IXIA traffic generator uses fuzzing strike lists to find out security vulnerabilities that crash systems. The fuzzing is a program designed to discover weak points in an application, an operating system or a network by feeding it with the massive inputting of random data.
6) *Generic:* A technique that establishes against block-cipher to cause a collision without respect to the configuration of the block-cipher.
7) *Reconnaissance:* Also can be defined as a probe, and it is an attack which gathers information about a computer network to evade its security controls.
8) *Worm:* A malware program that requires a user action to trigger and spread on computer systems and uses a computer network to replicate itself, depending on security failures of the target computer.

The data sources of the protocols from the UNSW-NB15 dataset include 349 000 records of DNS and 319 000 records of HTTP. The distribution of normal and attack records are provided in Table IV. These data sources are used for evaluating the ensemble method and measuring the effect of the proposed features for detecting malicious activities that expose Internet applications within these protocols.

The traffic of the NIMS dataset was captured from the four servers of Alexa, Zeus, Citadel, and Conficker, where the domain names of them were published in [5]. Alexa was configured to generate normal and malicious domain names. Zeus botnet was used for generating the domain names provided by ZeusTracker and DNS-BH blocklists. The domain names from the Zeus Tracker Citadel list forms the basis for Citadel botnet. Finally, Conficker domain names were collected from the University of Bonn and DNS-BH released Conficker domain name lists. HTTP-based communication with the domain names was collected from these servers. There are 95 729 normal and 45 370 botnet DNS instances in this dataset, and 7570 normal and 4256 botnet event HTTP instances. The description of the malicious types is provided

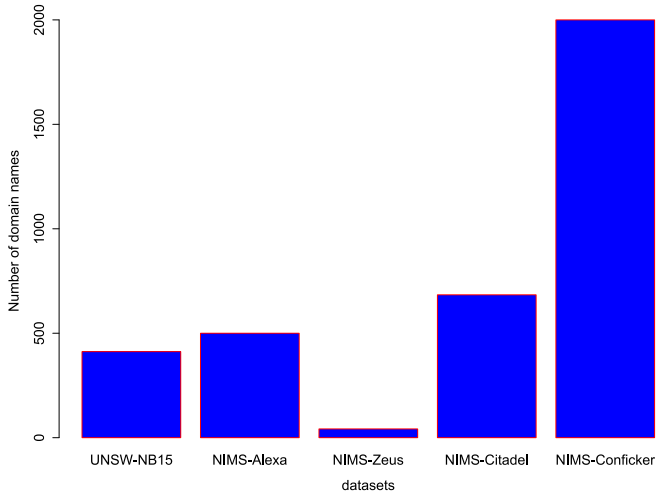

Fig. 6. Number of domain names included in both datasets.

in [5]. The number of domain names captured from both datasets are illustrated in Fig. 6. The botnet activities involved in the abnormal events of both datasets are combined with normal events to generate the statistical flow features.

B. Performance Evaluation

Several experiments were carried out for assessing the effectiveness and performance of the DT, NB, ANN, and ensemble method using the proposed data sources for identifying malicious events. To achieve this, the evaluation metrics of accuracy, DR, FPR, and ROC curves are used. These metrics relies on the four terms of true positive (TP), true negative (TN), false negative (FN) and false positive (FP). TP is the number of actual anomalous records detected as attacks, TN is the number of actual legitimate records detected as normal, FN is the number of actual anomalous records classified as normal and FP is the number of actual legitimate records classified as attacks. The metrics are described as follows.

1) The accuracy is the percentage of all legitimate and anomalous records that are correctly detected, that is,
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{10}$$
2) The DR is the percentage of correctly identified anomalous records, that is,
$$DR = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{11}$$
3) The FPR is the percentage of incorrectly identified anomalous records, that is,
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{12}$$
4) The *ROC curve* represents the relationship between the DR in the y-axis and the FPR in the x-axis, reflecting the overall performance of a system.

C. Discussion of Proposed Features

The features of the DNS and HTTP protocols, listed in Table I, were generated from the UNSW-NB15 and NIMS datasets. The statistical techniques of correntropy and CC is used for specifying the similarity and strength between normal feature vectors and malicious feature ones. Then, the ensemble learning framework is used for evaluating their performances on the two data sources. These techniques are developed using the R programming language on a Windows 7 operating system with 16 GB RAM and an i7 CPU.

These experiments were conducted with a tenfold cross validation model for appraising a fair performance with the dynamic change of data inputs. Models were trained and validated on a massive number of instances without duplications, as listed in Table IV, to overcome the under-fitting and over-fitting problems. The techniques were adjusted by the following parameters.

1) The DT technique was adjusted by *seed* = 1, *Confidence factor* = 0.25, *SubtreeRaising* = *True*, *batchSize* = 100, and *Unpruned* = *False*.
2) The NB was tuned by *NumDecimalPlaces* = 2, *UsekernelEstimator* = *True*, and *batchSize* = 100.
3) The ANN technique was fitted by *Seed* = 0, *Learning* − *rate* = 0.3, *Momentum* = 0.2, *batchSize* = 100, and *validationthreshold* = 20.
4) The ensemble technique was adjusted with the same parameters of the three techniques in addition to *treedepth* = 3, *Numrounds* = 200 and *Verbose* = *True*.

These data sources have quantitative and qualitative features but the statistical techniques can only handle quantitative features. Consequently, the qualitative features were converted into numeric ones to make it easier while running the statistical and machine learning techniques. For example, the UNSW-NB15 dataset has some categorical features including protocols (e.g., TCP and UDP), so the values of these protocols are replaced with ordered numbers, such as TCP=1, UDP=2, and so on. The conversion of categorical and numerical features might change statistical behaviors of normal and abnormal data, but it is essential when statistical learning models are utilized to identify cyber-attack events.

1) Evaluation of Correntropy Measure: The major benefit of using the correntropy measure for estimating the similarities between the normal and abnormal feature observations is that its mathematical functions are designed by computing the mean difference between two kernel density functions of given feature vectors. These functions are the basic solution for smoothing and identifying boundaries of feature values, obviously showing their variation and similarity [26]. Therefore, the dissimilarities between the DNS and HTTP features are presented in the correntropy plots for demonstrating the variations between their normal and suspicious feature vectors.

The dissimilarity between the normal and attack DNS and HTTP instances is shown in Figs. 7 and 8. The differences between their feature vectors is highlighted by plotting them within different colors to show the proposed features, emphasizing the clear difference between legitimate and suspicious instances. This has been developed with the ProfileR package [27]. Fig. 7 compares the normal and attack DNS instances for 100 instances. It can be seen that the values of normal instances vary between 0 and 28, which steadily rise and fall over the whole observations. Conversely, the values of attack

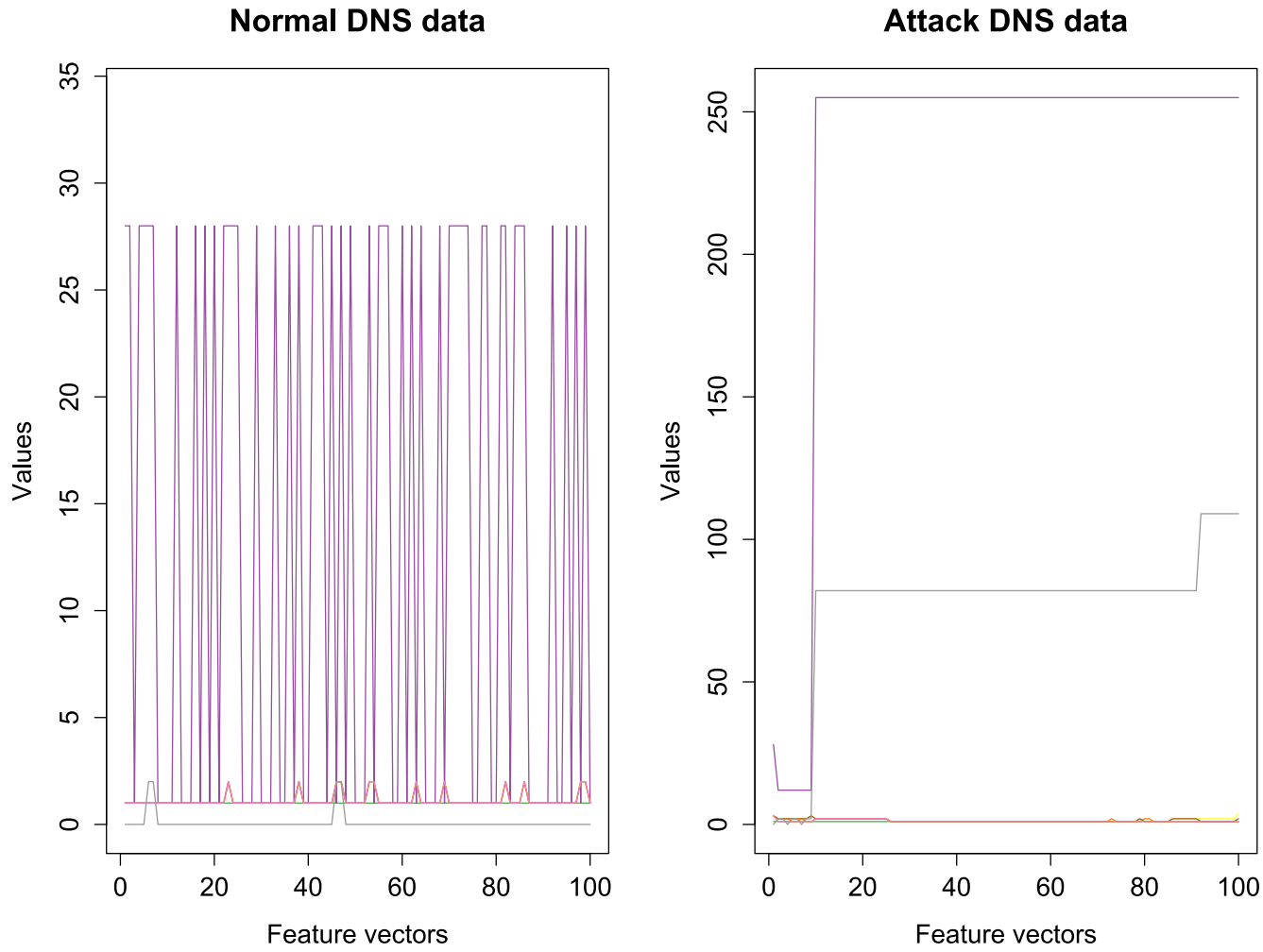

Fig. 7. Dissimilarity of normal and attack DNS instances.

instance vary between 0 and 250, which are clearly represented in three lines of 0, 85, and 250 in the y-axes. Overall, the proposed features of the DNS show the clear dissimilarity between the normal and attack instances.

Similarly, the normal and attack HTTP instances of 100 vectors are depicted in Fig. 8. The values of the normal vectors are between 0 and 500, while maintaining the same pattern over the instances, with values between 0 and 20. However, the values of attack observations fluctuated considerably between 0 and 150 across all the instances. Overall, the proposed features of the HTTP demonstrate the clear dissimilarity between normal and attack vectors.

The correntropy plot in Fig. 9 also statistically represents the slight differences between legitimate and malicious feature vectors over 100 instances for both protocols. This could simplify the role of machine learning algorithms to proficiently identify malicious observations, as explained below. The key reason for making the difference between normal and abnormal feature vectors is that the majority of the proposed features were created based on analyzing the statistical potential characteristics, such as the mean and length of protocol information with the flow identifiers. These characteristics lead to make clear variations between normal and attack activity for both protocols.

2) Correlation Coefficient for Selecting Important Features: The CC measure was applied to select the most important features due to its simplicity in computing the strength of the proposed features in a confidence interval of $[-1, 1]$, ranking these features easily. Features are considered relevant if they are uncorrelated to each other and correlated to the predictor (i.e., the normal or attack label). As shown in Fig. 10, the CC between the proposed features of the DNS and HTTP are grouped into three clusters in order to estimate their correlation scores.

Also in Fig. 10, each cluster refers to the strongest relationships to the lowest feature values, selecting the important ones from the lowest group as the most relevant features. It can be seen that there are weak relationships between the proposed features, meaning that each two different features are low in the correlation score. For example, in Fig. 10, the majority of the correlation scores are between -0.6 and 0.6, demonstrating the lower associations for these features. Hence, these features are relevant to classifying normal and attack observations. Also, the accuracy and detection rates of the classification techniques below reveal the effective impact of these features for establishing an efficient NIDS that can detect the DNS and HTTP malicious activity.

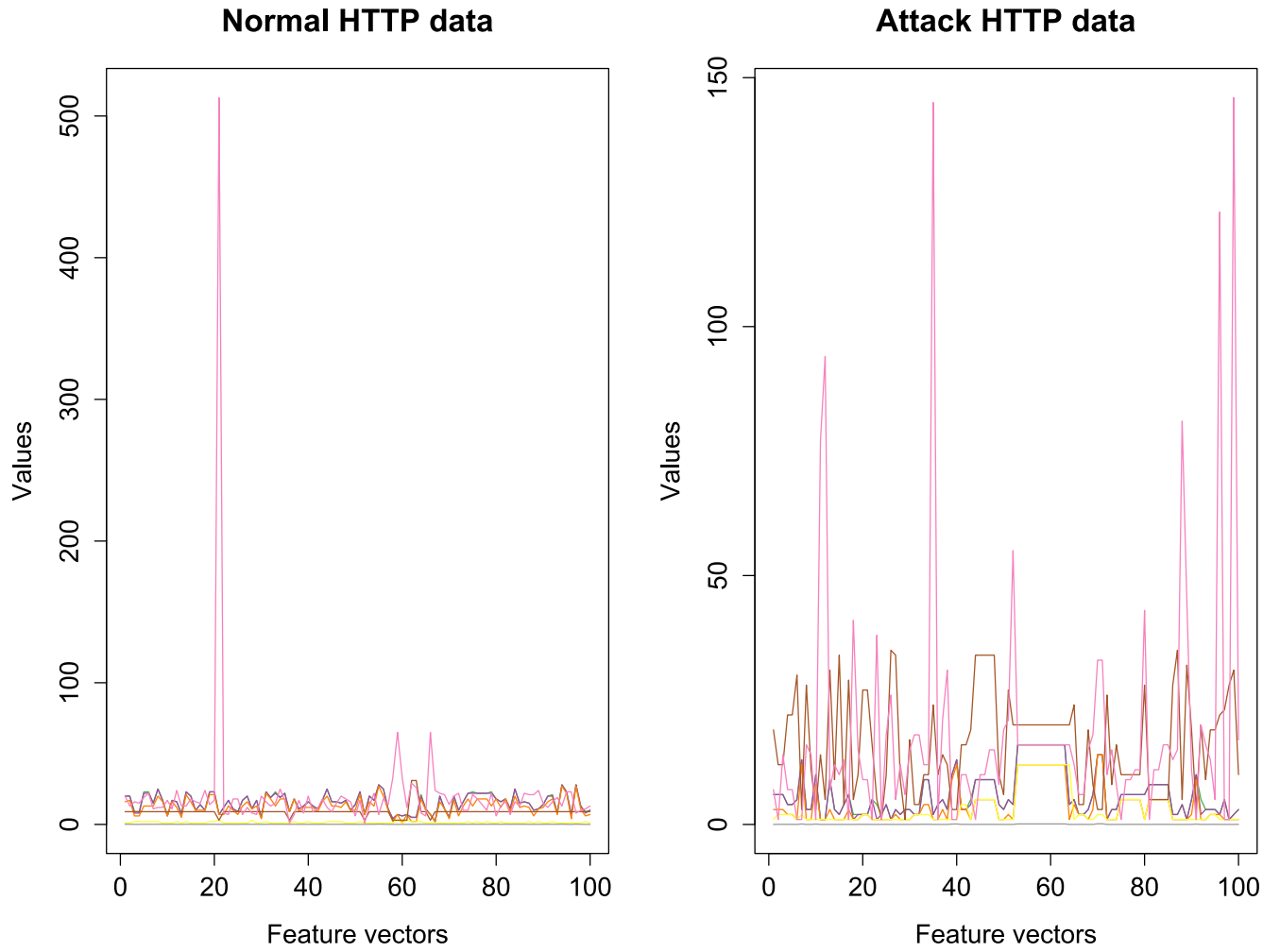

Fig. 8. Dissimilarity of normal and attack HTTP instances.

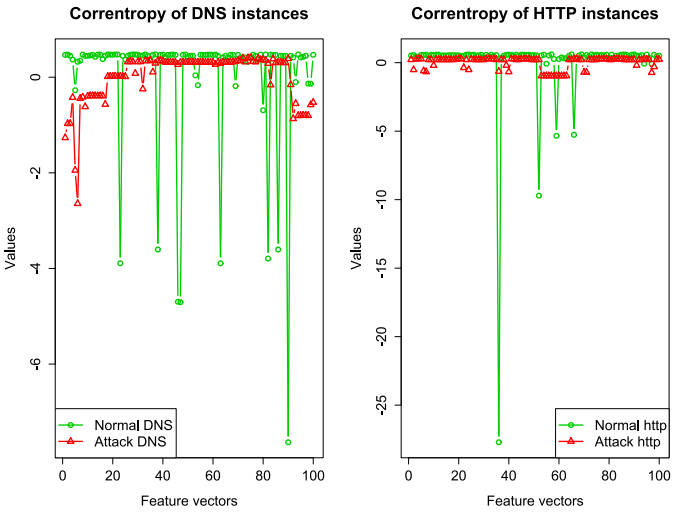

Fig. 9. Correntropy of DNS and HTTP instances.

D. Evaluation and Discussion

The overall performance evaluation of the DT, NB, ANN, and the suggested ensemble method in terms of Accuracy (Acc), DR, FPR, and processing time (Time) is discussed using the DNS and HTTP data sources of the UNSW-NB15 and NIMS botnet datasets as demonstrated in Tables IV and V, respectively. Moreover, the ROC curves that reflect the relationship between the DR and FPR are presented in Figs. 11 and 12.

On the one hand, using the DNS data source of the UNSW-NB15 dataset, the accuracy and DR of the ensemble method achieve 99.54% and 98.93%, respectively, while the FPR produces 1.38%, which outperforms the performance of the DT, NB, and ANN techniques. The DT technique produces a 95.32% accuracy, 94.15% DR, and 5.22% FPR, and then the ANN technique achieves a 92.61% accuracy, 91.48% DR, and 7.87% FPR. Lastly, the NB technique achieves an accuracy rate of 91.17%, 90.78% DR and 8.25% FPR. The ensemble technique consumes about 150.8 s on an average of 200 000–300 000 samples that is ranked as a second technique compared with the other techniques. Although this method sometimes takes computational resources, the AdaBoost method can pass the data to each technique that correctly classifies abnormal behaviors better than each algorithm alone. The computational complexity of the AdaBoost method is very low because each instance takes O(T), where T is the time of the voting toward the strong classifier.

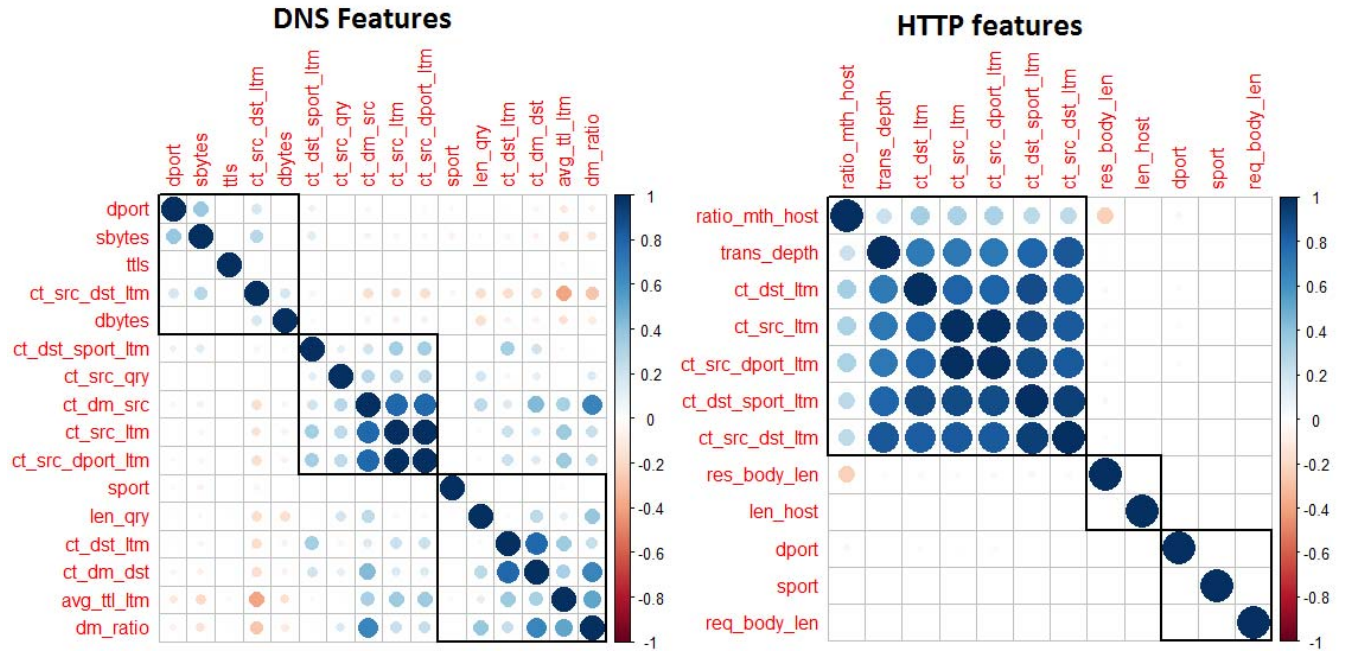

Fig. 10. Correlations of proposed DNS and HTTP features.

TABLE IV
EVALUATION PERFORMANCE ON UNSW-NB15 DATASET: A COMPARATIVE SUMMARY

Alg.	DNS data source				HTTP data source			
	Acc (%)	DR (%)	FPR (%)	Time Sec	Acc (%)	DR (%)	FPR (%)	Time Sec
DT	95.32	94.15	5.22	125.3	97.13	96.34	3.43	124.3
NB	91.17	90.78	8.25	130.2	95.91	95.25	4.18	131.1
ANN	92.61	91.48	7.87	220.2	96.27	95.53	4.26	217.2
Ensemble	99.54	98.93	1.38	150.8	98.97	97.02	2.58	148.3

TABLE V
EVALUATION PERFORMANCE ON NIMS DATASET: A COMPARATIVE SUMMARY

Alg.	DNS data source				HTTP data source			
	Acc (%)	DR (%)	FPR (%)	Time Sec	Acc (%)	DR (%)	FPR (%)	Time Sec
DT	96.10	95.02	4.19	128.5	97.23	95.92	4.65	129.6
NB	88.28	87.15	11.15	126.4	93.83	92.19	6.87	125.3
ANN	94.22	93.47	6.76	216.7	95.52	94.34	5.13	209.1
Ensemble	98.29	97.38	2.01	142.1	98.36	97.95	2.15	145.5

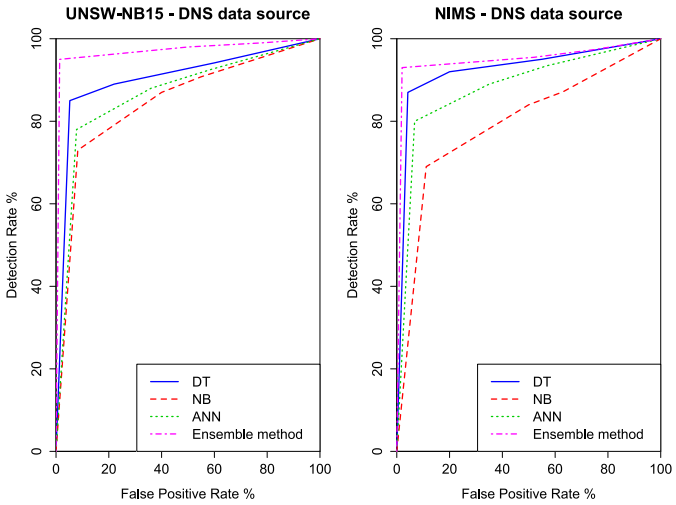

Fig. 11. ROC curves of classification techniques on DNS features.

On the other hand, using the HTTP data source of the UNSW-NB15 dataset, the ensemble method outnumbers each classification techniques separately, where it provides a 98.97% accuracy and 97.02% DR while a 2.58% FPR. The DT technique comes in the second order, providing a 97.13% accuracy, 96.34% DR and 3.43% FPR, and then the ANN technique produces a 96.27% accuracy, 95.53% DR, and 4.26% FPR. Finally, the NB technique constitutes a 95.91% accuracy, 95.25% DR, and 4.18% FPR. The ensemble method takes around 148.3 s on a range of 200 000–300 000 samples that is ordered as a third technique compared with the three techniques.

Using the DNS data source of the NIMS dataset, the accuracy and DR of the ensemble method accomplish 98.29% and 97.38%, respectively, whereas the FPR introduces 2.01%, which outweighs the performance of the other three techniques. The DT technique provides a 96.10% accuracy, 95.02% DR and 4.19% FPR, with around 142.19 s while processing the technique on 300 000 samples. Then, the ANN technique produces a 94.22% accuracy, 93.47% DR, and 6.76% FPR. Finally, the NB technique provides a 88.28% accuracy, 87.15% DR, and 11.15% FPR. However, using the HTTP data source of the NIMS, the ensemble method outnumbers each classification techniques separately, where it reflects a 98.36% accuracy and 97.95% DR while a 2.15% FPR, and its processing time is approximately 145.56 s. Second, The DT technique achieves a 97.23% accuracy, 95.92% DR, and 4.65% FPR, and then the ANN technique accomplishes a 95.52% accuracy, 94.34% DR, and 5.13% FPR. Finally,

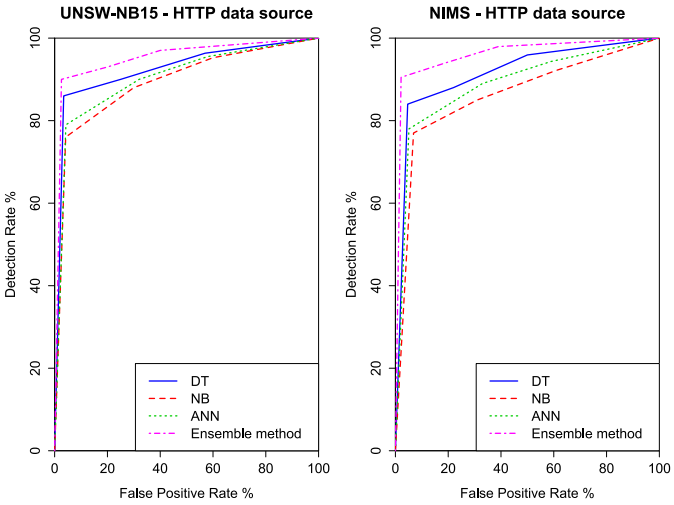

Fig. 12. ROC curves of classification techniques on HTTP features.

TABLE VI
COMPARISONS OF DRS (%) AND FPRS (%) USING ENSEMBLE METHOD

Record types	DNS data source		HTTP data source	
	DR	FPR	DR	FPR
Normal	99.53	0.24	99.01	0.15
DoS	98.22	0.31	98.54	0.42
Exploits	96.57	0.52	95.25	0.57
Fuzzers	99.86	0.01	99.23	0.02
Generic	99.43	0.13	96.63	0.72
Reconnaissance	99.78	0.04	98.57	0.24
Analysis	-	-	99.20	0.38
Backdoor	-	-	95.25	0.61
Worms	-	-	99.62	0.05

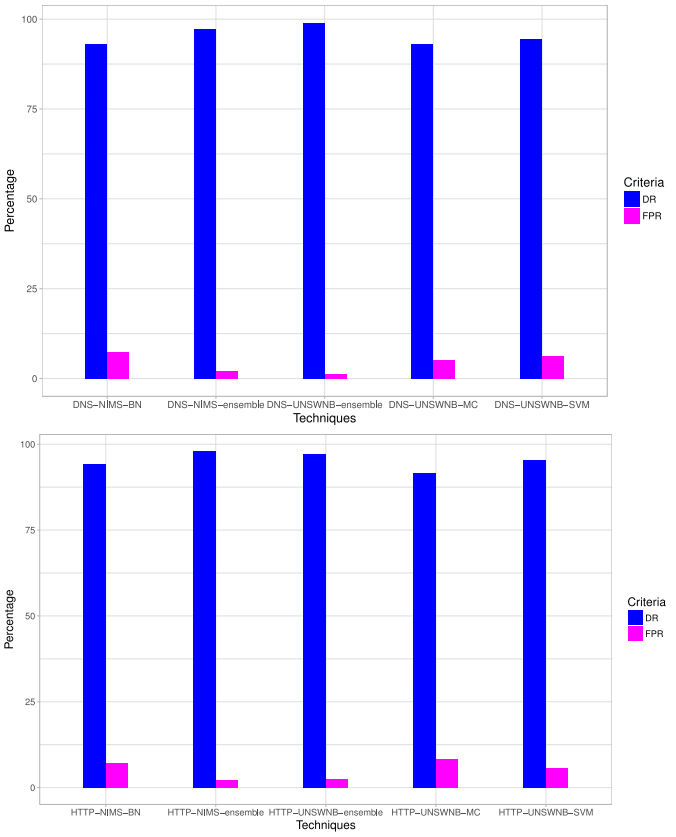

Fig. 13. Comparison of four techniques on both datasets.

the NB technique accounts for a 93.83% accuracy, 92.19% DR, and 6.87% FPR.

The evaluation of the ensemble method for record types on both data sources is listed in Table VI. The performance of the ensemble method shows that DR and FPR are approximately between 95.25% and 99.86% and between 0.01% and 0.72%, respectively. These rates reveal that the ensemble method with the proposed features can efficiently detect the attack types that expose the DNS and HTTP protocols. Moreover, the detection rates of the normal data using the two data sources are more than 99%, revealing the lowest FN rates while identifying the normal observations.

For describing which learner utilized by the ensemble method during the implementation, the AdaBoost fits the DT, ANN, and NB techniques as three weak learners on different weighted training data. It begins by predicting network data and gives equal weight to each instance. If the prediction is wrong using the first learner, then it applies the other learners till reach the correct prediction using a ten cross validation model. Selecting the strong classifier depends on executing shuffling in the data instances and taking the average as the final output.

The classification techniques of the ensemble method are chosen based on the statistical interpretation of the network data. This means that the DT can classify data instances with the corresponding class labels using the highest information gain function to take a decision if a record is an attack or not. Moreover, the ANN can weigh the small difference of features in order to define abnormal instances using an activation function. Finally, the NB can take the decision, whether a record is normal or attack, based on the posterior probability of that record with a specific threshold. By combining these techniques and distributing the network data sources of the DNS and HTTP using the AdaBoost methodology, the NIDS performance can be improved in terms of accuracy, DR and FPR with no effect on the time processing compared with each classification technique.

The experimental results of our ensemble framework are compared with three existing techniques; support vector machines (SVM) [28], BN [28], and Markov chain (MC) for botnets [29]. The evaluation criteria of the DR and FPR show the superiority of our framework compared with the three techniques as depicted in Fig. 13.

Although the proposed method produces a very low FPR rate using different data sources, there are various synthetic attack types used while training and validating the model that could not concurrently breach real production systems. The performance of the method has the potential to be considerably improved by incorporating with one the following ways. First, the method needs to be integrated with another technique that can generate statistical feature profiles of different TCP/IP protocols for effectively discovering attack types. Second, the method demands new kernel functions that can discriminate between small variations of normal and attack values for considerably improving the DR of attack types.

There are two main reasons why the proposed features and ensemble method can significantly detect malicious events that attempt to breach network applications within the DNS and HTTP protocols. First, the proposed features are designed based on the statistical analysis measures, specifically the mean, length and number of event occurrences. These measures can precisely identify the small differences between legitimate and anomalous observations. As previously discussed, the correntropy and CC are used for estimating the similarity and strength of the proposed features. The two measures showed that the proposed features can considerably differentiate between the legitimate and suspicious patterns of both protocols. With the wide variety of abnormal types in the UNSW-NB15 and NIMS botnet datasets, these measures clarified the differences between the legitimate and malicious patterns, as shown in Figs. 6–8.

Second, the use of the AdaBoost ensemble method enhanced the performance evaluation compared with each technique involved in the proposed ensemble framework. The ensemble method was designed based on the AdaBoost model, which distributes the input data to handle the base learners with an error function. This function assigns the error value to each instance to decide which of the base learners can correctly classify this instance. This is because the adaptive boosting can deal with the small differences of the feature vectors via computing the error function, and if the classifier cannot correctly recognize a particular instance, another classifier will be used for classifying that instance. Since modern attacks attempt to mimic normal behaviors, there are small differences between those normal and attack behaviors, as in the two datasets, so the proposed ensemble framework can effectively process those data sources.

VI. Conclusion

In this paper, a set of features is identified from an in-depth analysis of the TCP/IP model, in particular MQTT, DNS and HTTP protocols, and their flow identifiers to build an effective NIDS for detecting attacks that exploit IoT networks. These features were extracted by the Bro-IDS to capture the basic information of the protocols, and a new extract module to generate statistical features from the basic information captured. To measure the effect of these features, data sources were generated using the proposed features from the source files of the UNSW-NB15 and NIMS datasets, in addition to the network traffic of simulated IoT sensors. The effect of these features is evaluated using the correntropy and CC measures. Separately the performance of the proposed ensemble framework is analyzed. An AdaBoost ensemble method using three techniques of DT, NB, and ANN was applied to improve the overall performance in terms of accuracy, DR, and time processing compared with some of the state-of-the-art-methods. The experimental results demonstrated that the suggested ensemble method on the data sources of the DNS and HTTP protocols outperformed existing techniques in the ensemble method and existing mechanisms of SVM and BN and MC. This can be explained as the proposed features have the potential characteristics of legitimate and suspicious events as well as the architecture of building the ensemble method-based NIDS in lower overhead compared with other methods.

In the future, this paper will be extended to collect relevant features from other IoT protocols in order to build a dynamic profile of normal and attack patterns. Such a database of features can be used to facilitate the detection of existing and zero-day attacks using the proposed ensemble method.

References

[1] W. Shang, Y. Yu, R. Droms, and L. Zhang, "Challenges in IoT networking via TCP/IP architecture," NDN Project, Rep. NDN-0038, 2016. [Online]. Available: https://named-data.net/wp-content/uploads/2016/02/ndn-0038-1-challenges-iot.pdf

[2] S. K. Malladi, T. M. Ravi, M. K. Reddy, and K. Raghavendra, "Edge intelligence platform, and Internet of Things sensor streams system," U.S. Patent Appl. 15 250 720, Mar. 2, 2017.

[3] (Mar. 2018). *The-UNSW-NB15-Dataset*. [Online]. Available: https://www.unsw.adfa.edu.au/australian-centre-for-cybersecurity/cybersecurity/ADFA-NB15-Datasets/

[4] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. IEEE Military Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.

[5] (Mar. 2018). *The-NIMS-Dataset*. [Online]. Available: https://projects.cs.dal.ca/projectx/Download.html

[6] (Mar. 2018). *AWS-IoT-Hub*. [Online]. Available: https://aws.amazon.com/iot-core/

[7] (Mar. 2018). *The-Bro-IDS-Tool*. [Online]. Available: https://www.bro.org/

[8] T.-S. Wang, H.-T. Lin, W.-T. Cheng, and C.-Y. Chen, "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis," *Comput. Security*, vol. 64, pp. 1–15, Jan. 2017.

[9] N. Moustafa, G. Creech, and J. Slay, "Flow aggregator module for analysing network traffic," in *Progress in Computing, Analytics and Networking*. Singapore: Springer, 2018, pp. 19–29, doi: 10.1007/978-981-10-7871-2_3.

[10] Y. Bouzida and F. Cuppens, "Neural networks vs. decision trees for intrusion detection," in *Proc. IEEE/IST Workshop Monitor. Attack Detect. Mitigat. (MonAM)*, vol. 28, 2006, p. 29.

[11] D. M. Farid, L. Zhang, C. M. Rahman, M. A. Hossain, and R. Strachan, "Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1937–1946, 2014.

[12] P. Kazienko and P. Dorosz, "Intrusion detection systems (IDS) Part I—(Network intrusions; attack symptoms; IDS tasks; and IDS architecture)," vol. 20, no. 2009, Apr. 2003.

[13] G. Giacinto and F. Roli, "An approach to the automatic design of multiple classifier systems," *Pattern Recognit. Lett.*, vol. 22, no. 1, pp. 25–33, 2001.

[14] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Comput. Security*, vol. 24, no. 4, pp. 295–307, 2005.

[15] S. Marchal, X. Jiang, R. State, and T. Engel, "A big data architecture for large scale security monitoring," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, 2014, pp. 56–63.

[16] F. A. Teixeira *et al.*, "Defending Internet of Things against exploits," *IEEE Latin America Trans.*, vol. 13, no. 4, pp. 1112–1119, Apr. 2015.

[17] X. Cao *et al.*, "Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 816–829, Oct. 2016.

[18] A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust Malware detection for Internet of (battlefield) Things devices using deep eigenspace learning," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2018.2809665.

[19] Q. Chen, S. Abdelwahed, and A. Erradi, "A model-based validated autonomic approach to self-protect computing systems," *IEEE Internet Things J.*, vol. 1, no. 5, pp. 446–460, Oct. 2014.

[20] H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, and K.-K. R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2016.2633228.

[21] 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)*, 2007, pp. 715–720.

[22] M. Anagnostopoulos, G. Kambourakis, P. Kopanos, G. Louloudakis, and S. Gritzalis, "DNS amplification attack revisited," *Comput. Security*, vol. 39, pp. 475–485, Nov. 2013.
[23] M. Nassar, B. al Bouna, and Q. Malluhi, "Secure outsourcing of network flow data analysis," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, 2013, pp. 431–432.
[24] K. Jee *et al.*, "Host level detect mechanism for malicious DNS activities," U.S. Patent Appl. 15 644 018, Jan. 11, 2018.
[25] G. Wang, J. Sun, J. Ma, K. Xu, and J. Gu, "Sentiment classification: The contribution of ensemble learning," *Decis. Support Syst.*, vol. 57, pp. 77–93, Jan. 2014.
[26] W. Ma, H. Qu, and J. Zhao, "Estimator with forgetting factor of correntropy and recursive algorithm for traffic network prediction," in *Proc. IEEE 25th Chin. Control Decis. Conf. (CCDC)*, 2013, pp. 490–494.
[27] (Jun. 2018). *The-ProfileR-Package*. [Online]. Available: https://cran.r-project.org/web/packages/profileR/profileR.pdf
[28] F. Haddadi and A. N. Zincir-Heywood, "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1390–1401, Dec. 2016.
[29] Z. Abaid, D. Sarkar, M. A. Kaafar, and S. Jha, "The early bird gets the botnet: A Markov chain based early warning system for botnet attacks," in *Proc. IEEE 41st Conf. Local Comput. Netw. (LCN)*, 2016, pp. 61–68.

Nour Moustafa (GS'15–M'18) received the bachelors and master degrees in computer science from the Faculty of Computer and Information, Helwan University, Helwan, Egypt, in 2009 and 2014, respectively, and the Ph.D. degree in cyber security from the University of New South Wales (UNSW), Canberra, ACT, Australia, in 2017.

He is a Post-Doctoral Fellow with the UNSW Canberra Center, School of Engineering and Infromation Technology, UNSW. He started his academic career as an Assistant Lecturer with the Faculty of Computers and Information, Helwan University, in 2011. His current research interests include cyber security, in particular, network security, host-and network-intrusion detection systems, statistics, deep learning and machine learning techniques, designing and developing threat detection and forensic mechanisms to the Industry 4.0 technology for identifying malicious activities from cloud computing, fog computing, IoT, and industrial control systems over virtual machines and physical systems.

Benjamin Turnbull (M'16) received the Ph.D. degree from the University of South Australia, Adelaide, SA, Australia.

He is a Senior Lecturer with the University of New South Wales, Canberra, ACT, Australia. He was with the Australian Government Defence Science and Technology Organization, Canberra, initially for the Computer Network Defence and Forensics Group and later for automated analytics and decision support. His current research interests include novel cyber security defence strategies, cyber simulation, and understanding the physical impact of cyber attacks. As part of this, he investigates the nexus of cyber security and kinetic effect to understand the true impacts of cyber attack, best-practice automated analysis, and visual techniques to aid decision support. This involves research in the fields of digital forensics, cyber security, knowledge representation, and visual analytics domains.

Kim-Kwang Raymond Choo (SM'15) currently holds the Cloud Technology Endowed Professorship with the University of Texas at San Antonio (UTSA), San Antonio, TX, USA.

Prof. Choo was a recipient of the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, the IEEE TrustCom 2018 Best Paper Award, the ESORICS 2015 Best Research Paper Award, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008. He was named the Cybersecurity Educator of the Year—APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn) in 2016. He and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen–Nuremberg in 2015.