Hindawi
Security and Communication Networks
Volume 2018, Article ID 1263123, 13 pages
https://doi.org/10.1155/2018/1263123

WILEY | Hindawi

Research Article

Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest

Mohamed Idhammad,[1] **Karim Afdel**,[1] **and Mustapha Belouch**[2]

[1]*LabSIV, Department of Computer Science, Faculty of Science, Ibn Zohr University, Agadir, Morocco*
[2]*LAMAI, Department of Computer Science, FSTG, Cadi Ayyad University, Marrakesh, Morocco*

Correspondence should be addressed to Mohamed Idhammad; idhammad.mohamed@edu.uiz.ac.ma

Received 29 November 2017; Revised 19 April 2018; Accepted 29 April 2018; Published 5 June 2018

Academic Editor: Huaizhi Li

Cloud Computing services are often delivered through HTTP protocol. This facilitates access to services and reduces costs for both providers and end-users. However, this increases the vulnerabilities of the Cloud services face to HTTP DDoS attacks. HTTP request methods are often used to address web servers' vulnerabilities and create multiple scenarios of HTTP DDoS attack such as Low and Slow or Flooding attacks. Existing HTTP DDoS detection systems are challenged by the big amounts of network traffic generated by these attacks, low detection accuracy, and high false positive rates. In this paper we present a detection system of HTTP DDoS attacks in a Cloud environment based on Information Theoretic Entropy and Random Forest ensemble learning algorithm. A time-based sliding window algorithm is used to estimate the entropy of the network header features of the incoming network traffic. When the estimated entropy exceeds its normal range the preprocessing and the classification tasks are triggered. To assess the proposed approach various experiments were performed on the CIDDS-001 public dataset. The proposed approach achieves satisfactory results with an accuracy of 99.54%, a FPR of 0.4%, and a running time of 18.5s.

1. Introduction

Cloud Computing aims to provide convenient and on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interactions [1]. Cloud Computing services and APIs are often delivered through HTTP protocol. This facilitates access to services and reduces costs for both providers and end-users. However, this makes Cloud services vulnerable to attacks that exploit vulnerabilities of HTTP protocol such as HTTP DDoS attacks.

Despite the important evolution of the information security technologies in recent years, DDoS attacks continue to threat Internet services and new records are breached each year. Recently, a destructive DDoS attack has brought down more than 70 vital services of Internet including Github, Twitter, Amazon, Paypal, etc. Attackers have taken advantages of Cloud Computing and Internet of Things technologies to generate a huge amount of attack traffic, more than 665 Gb/s [2, 3].

HTTP DDoS attacks are highly sophisticated DDoS types working at the application layer. Several HTTP request methods are often used to address web servers' vulnerabilities and create multiple scenarios of HTTP DDoS attack such as Low and Slow or Flooding attacks. These attacks are often based on POST and GET request methods which are widely used for submitting sensitive data to the Internet. HTTP DDoS attacks are divided into two main categories: high rate and low rate attacks. In the high rate attack scenario the victim is flooded by a large number of HTTP requests. While, in the low rate attack scenario slow and compromised requests are used to engage the victim in high complexity computations which consume a significant amount of its resources [4]. As there

is no illegitimate TCP or UPD packets in these attacks they avoid easily the network layer detection techniques [5, 6]. The engines behind these attacks belong to one or several Botnets. A Botnet is a network of compromised computers in the Internet which run a malicious program called Bot or Agent. These computers, often known as Zombies or Reflectors, are remotely controlled by the Botmaster [7–11]. The Reflectors are mainly used to reflect the behavior of the attacker and hide his identity. A new kind of Botnet based on mobile devices, smart-phones and tablets, is starting to emerge and gain popularity among cybercriminals [12]. The exploitation of this new Botnet infrastructure is very effective and powerful compared to traditional Botnets due to the huge number of mobile devices linked to Internet. For instance, cybercriminals have used this Botnet mobile infrastructure to accomplish one of the most powerful DDoS attacks in history of Internet [2, 3].

Several techniques were developed to mitigate the high rate attacks such as establishing a speed floor of network flow, requests counting, bandwidth limitation, etc. In contrast, few researches have been done for detecting and mitigating the slow HTTP DDoS attacks. The fact of highly mimicking the normal behaviors and the use of completely different strategies makes these attacks hard to detect.

Since these attacks are easy to mount with a minimum of resources they make barriers to the adoption of the Cloud Computing [19]. In a Cloud environment a large number of attacks may occur simultaneously and many services may be targeted at the same time [20, 21]. This causes generating significant amounts of network traffic data between the Reflectors and the targeted services in the Cloud. The analysis of this network flow data is a crucial task for detecting the attack. It is clear that a defense system against these attacks should preprocess and classify efficiently and in a fastest pace this large volume of network traffic data.

In this paper we present a detection system of HTTP DDoS attacks in a Cloud environment based on Information Theoretic Entropy and Machine Learning. The proposed detection system consists of three main steps: entropy estimation, preprocessing, and classification. A time-based sliding window algorithm is used to estimate the entropy of the network header features of the incoming network traffic. When the average of the features' entropy exceeds its normal range the preprocessing algorithm cleans and normalizes the network traffic data of the current time window. The preprocessed network traffic data is then classified into normal and HTTP DDoS traffic. A test procedure is used to select appropriate classifier for HTTP DDoS detection based on accuracy, FPR, AUC, and running time metrics. The obtained results from experiments in Section 8.2 revealed that the Random Forest ensemble classifiers depict high detection performance for HTTP DDoS attacks.

The main contribution of this paper can be summarized as follows:

(i) Proposing a HTTP DDoS attacks detection system in a Cloud environment based on Information Theoretic Entropy and Machine Learning.

(ii) Adopting a time-based sliding algorithm for estimating Entropy of the incoming network traffic to the Cloud infrastructure.

(iii) Adopting a procedure to select appropriate machine learning classifier for HTTP DDoS attack detection.

The remainder of this paper is organized as follows. Section 2 gives an overview of the related works. Section 3 is devoted to the dataset used in this paper. The HTTP DDoS attacks are detailed in Section 4. Section 5 illustrates the impact of HTTP DDoS attacks on a Cloud environment. A thorough explanation of the proposed approach is presented in Section 6. Section 7 gives the details of the conducted experiments and the performance metrics used to evaluate the proposed approach. The results and discussion are given in Section 8. Finally, this paper ends with the main conclusion.

2. Related Works

In this paper information theory and machine learning techniques are used to improve the HTTP DDoS attacks detection accuracy, false positive rates, and running time. Many previous works were devoted to enhance the detection performance of HTTP DDoS attacks. In this section we summarize some of the recent works in the detection of HTTP DDoS attacks.

Sangjae L et al. [22] have proposed an App-DDoS detection method based on a sequence-order-independent network traffic profiling technique. Authors extract attributes from web page request sequences and use a PCA-based model for the profiling of normal web browsing behaviors. The DDoS attacks are then detected according to a criterion based on the reconstruction error of the profiling model. The method is evaluated with various types of App-DDoS attacks and important results are achieved.

Dantas Y et al. [23] have proposed a defense mechanism against HTTP POST Flooding attack, called SeVen, which is based on Adaptive Selective Verification (ASV). As ASV was designed for mitigating Network Layer DDoS attacks, it assumes that communications are simple client-server stateless syn-ack interactions. This, however, is not enough for mitigating Application Layer DDoS attacks, as the protocols used by these attacks, such as HTTP, have a notion of state. SeVen, thus, extend ASV by incorporating the defense a notion of state. The main key in SeVen for defending against HTTP POST Flooding attack is the number of pieces of data already processed. An application using SeVen sends an acknowledgment only when the total payload is received. However, this keeps the resources busy while waiting for receiving the total payload. Therefore, the mechanism will fail against a distributed HTTP POST Flooding attack when a large number of Reflectors are used to send payloads.

The authors of [24] have presented a method of DDoS attack detection using HTTP packet pattern and rule engine in a Cloud Computing environment. The method integrates between HTTP GET flooding among DDoS attacks and MapReduce processing, for fast attack detection in a Cloud

Computing environment. This method can ensure the availability of the target system for accurate and reliable detection of HTTP GET flooding. The method was compared with the Snort IDS based on the processing time and the reliability when the congestion increases in the Cloud infrastructure.

Thomas V et al. have proposed in [5] a system for defending against two types of Application Layer DDoS attacks in the Cloud environments, in particular XML-DDoS and SOAP-DDoS. The proposed defense system is specific for threats involved with web service deployment. It does not replace the lower-layer DDoS defense systems that target network and transportation attacks. The authors propose an intelligent, fast, and adaptive system for detecting XML and HTTP application layer attacks. The intelligent system works by extracting several features and use them to construct a model for typical requests. Finally, outliers detection can be used to detect malicious requests. Furthermore, the intelligent defense system is capable of detecting spoofing and regular flooding attacks. The system is designed to be inserted in a Cloud environment where it can transparently protect the Cloud broker and even Cloud providers.

Also, Aiello et al. [25] have proposed a detection method that analyzes specific spectral features of traffic over small time horizons without packet inspection. Real traffic traces mixed with several low rate HTTP DDoS attacks are collected locally from their institute, LAN, and are used to evaluate the method. Satisfactory results are obtained by the method.

Recently, the authors of [26] have proposed a Protocol-Free Detection (PFD) against Cloud oriented Reflection DoS (RDoS) attacks. They focus on analyzing the network flow of the Cloud services, by studying the basic traffic correlation near the victim Cloud under RDoS attack. PFD is protocol-free and its computation cost will not be affected by network throughput. In PFD, packet rate is sampled in upstream router and correlation of flows is tested using flow correlation coefficient (FCC), and the detection result is given by considering current FCC value and historical information. In the Cloud environment, PFD is designed to be inserted in a protected virtual LAN. However, a protected VLAN requires deployment of other security techniques which consume the Cloud resources and effect against it. Also, deploying the PFD inside the Cloud instances makes it vulnerable to the HTTP DDoS attacks.

Qin et al. [17] have proposed an Application Layer DDoS attack detection system based on two machine learning techniques. First, authors consider users' requests frequency sequence as sparse vector and then use a classification algorithm called sparse vector decomposition and rhythm matching (SVD-RM). Then the clustering algorithm L-Kmeans is used as embedded classifier in SVD-M. The system was tested against four Application Layer DDoS attacks and the system achieved good detection results.

Sree et al. [18] have proposed to detect HTTP GET flooding attack by reading the web server logs, extracting the relevant features and using analytical hierarchical process to predict whether the attack has occurred or not, and detecting the suspicious sources by using Dempster-Shafer theory of evidence. The authors use MapReduce techniques to analyze large volumes of log data which allows improving the processing time of their method. The method was evaluated using the public Cyber Research Centre (CDX) dataset [18] and a dataset generated locally by authors.

Zecheng He et al. [14] have proposed a DDoS detection system based on machine learning techniques. The system is designed to be implemented on the Cloud provider's side in order to early detect DDoS attacks sourced from virtual machines of the Cloud. The system leverages statistical information from both the Cloud server's hypervisor and the virtual machines, in order to prevent network packages from being sent out to the outside network. Nine machine learning algorithms are evaluated and the most appropriate is selected based on the detection performances.

Similarly, Sreeram et al. [15] have proposed a Bio-Inspired Anomaly based Application Layer DDoS attack (App-DDoS attack) detection in order to achieve fast and early detection. The proposed system is a bioinspired bat algorithm which is used to detect the HTTP DDoS attacks. The authors have evaluated their system using the CAIDA dataset. The system achieved satisfactory results for the detection of HTTP flooding attacks.

Irfan et al. [16] have proposed a detection system of HTTP DDoS flooding attacks based on machine learning techniques. Four machine learning classifiers, namely, Nave Bayes, MLP, SVM, and Decision trees, are evaluated using a local dataset. The system achieved satisfactory classification results of the collected network traffic data.

Most of the HTTP DDoS detection approaches proposed in the literature use statistical and thresholding techniques to discriminate attack and normal traffic. This however results in low detection accuracy when attacks' behaviors are changed and large false positive rates for the thresholding techniques. A major solution to the above issues is to adopt appropriate machine learning techniques and models in order to increase the accuracy and reduce false positive rates. Furthermore, Information Theoretic entropy allows focusing only on abnormal network traffic. This reduces drastically the amount of data to preprocess and to classify which improve performance and running time of the HTTP DDoS detection.

3. The Adopted CIDDS-001 Dataset

CIDDS-001 (Coburg Intrusion Detection Dataset) is an up-to-date labeled flow-based dataset created by M. Ring et al. [27] in a Cloud environment based on OpenStack platform. This environment includes several clients, emulated using a set of Python scripts, and typical servers including E-Mail server, Web server, etc. The dataset contains realistic normal and attack traffic allowing important benchmarking of network intrusion detection systems in a Cloud environment. The dataset is divided into four parts each is created during a week. The CIDDS-001 is a flow-based format dataset containing unidirectional NetFlow [27] data. The dataset contains 14 attributes, the first 10 attributes are the default NetFlow attributes, and the last four attributes are additional attributes. Names and descriptions of features are tabulated in Table 1. A total of 32 million of normal and attack flows are captured in the dataset within four weeks. Authors have exploited 92 types of attacks to create the dataset including

Table 1: Features of the CIDDS-001 dataset [13].

Feature	Description
1. Src IP	Source IP Address
2. Src Port	Source Port
3. Dest IP	Destination IP Address
4. Dest Port	Destination Port
5. Proto	Transport Protocol (e.g., ICMP, TCP, or UDP)
6. Date first seen	Start time flow first seen
7. Duration	Duration of the flow
8. Bytes	Number of transmitted bytes
9. Packets	Number of transmitted packets
10. Flags	OR concatenation of all TCP Flags
11. Class	Class label (normal, attacker, victim, suspicious or unknown)
12. AttackType	Type of Attack (portScan, dos, bruteForce, —)
13. AttackID	Unique attack id. All flows which belong to the same attack carry the same attack id.
14. Attack Description	Provides additional information about the set attack parameters (e.g., the number of attempted password guesses for SSH-Brute-Force attacks)

DDoS attack over HTTP. This makes the dataset an important benchmark for HTTP DDoS detection systems. The reasons above motivated us to use this dataset in this paper. In our experiments we used only the first day of the first week, Monday 15 March 2017. This part contains 1,501,856 instances of network traffic data with 11.12% corresponding to DDoS attack traffic.

4. HTTP DDoS Attacks

HTTP DDoS attacks are divided into two main categories: high rate and low rate attacks. In the high rate HTTP DDoS attacks, often known as HTTP flooding attacks, the target is flooded by a large number of HTTP requests. These attacks are simple flooding attacks that generates as many as possible of HTTP request packets by using all available resources of the Reflectors. They behave very similar to the traditional DoS attacks which results in a total consumption of the victim's resources and/or exhausting the communication channel. Many tools are available on the Internet to generate HTTP flooding attacks such as HTTPFlooder [28]. This type of attacks can be easily detected by packets counting or by bandwidth thresholding techniques [24].

Unlike flooding, low and slow HTTP DDoS attacks do not require a large amount of network traffic or high resources of attackers to deny services at the victim server. They target specific design flaws or vulnerabilities on the target server with a relatively small amount of compromised packets. This engages the target in high complexity computations or in massive allocation of resources. These attacks are very hard to detect and to mitigate, because they involve connections and data transfer appearing to be at normal rate [29]. There exist several types of slow HTTP DDoS attacks. Attackers often target vulnerabilities in different HTTP request methods to accomplish the attack. In this work we cover three categories of slow HTTP DDoS attacks, namely, Slow Read, Slow Message Body, and Slow Header attacks. These three categories are generated using specific vulnerabilities in POST and GET request methods. Although these attacks were created first to target vulnerabilities of HTTP/1.1, they are still effective on the updated HTTP/2 [30]. A detailed categorization of slow HTTP DDoS attacks can be found in [31].

4.1. Slow Message Body Attack (Slow POST). When a legitimate client aims to send data to a web server using the POST method, after establishing a TCP connection, it sends first an HTTP message header in which it mentions the amount of data to send. The field used to communicate this information to the web server is the *Content-Length*. When the server receives the message header it reads the *Content-Length* value. Then, it maintains the connection alive and resources running until receiving the total volume of data mentioned in the *Content-Length* field. In its turn, the client divides the message body to chunks according to his network resources. Then, it sends the chunks successively to the server [23, 29]. Most web servers obey the *Content-Length* value and maintain resources running while waiting until the end of the message body [29, 32]. This behavior makes the majority of web servers vulnerable to the Slow Message Body attack.

A malicious client follows several steps to accomplish the Slow Message Body attack. First, the malicious client builds a legitimate HTTP POST header with a fake *Content-Length* value, 5000 kilobytes, for example, and sends it to the server [33]. Usually, this *Content-Length* value is larger than the actual size of the message body. Most web servers accept up to 2 GB in the *Content-Length* value. The server receives the header portion and waits for receiving the total bytes of the message body [32, 34]. Next, the malicious client divides the message body into small chunks and sends them very slowly to the server, for example, 1 byte per chunk every 25s.

Figure 1 shows the sequential diagram of the Slow Message Body attack. As fast as possible, the malicious client repeats the operation with all his Reflectors and expands his Slow Message Body script over his Botnet. This action

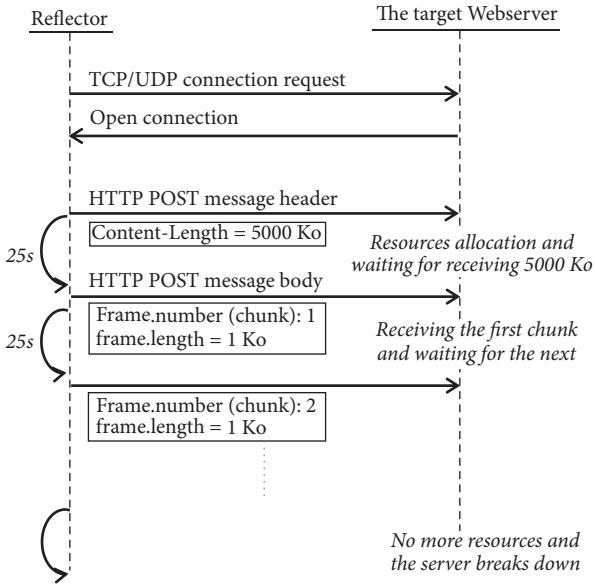

Figure 1: Sequential diagram of the Slow Message Body attack.

multiplies the magnitude and the efficiency of the attack [32]. The web server tries to manage this large number of connections. Consequently, within minutes the web server consumes all his resources and breaks down resulting in a denial of services.

Several methods have been used to detect this attack using statistics, information theory, and machine learning techniques. However, attackers mimic the normal behavior and success to hijack the existing detection systems. Hence, the detection of Slow Message Body attack requires further sophisticated techniques that combine information theory techniques to monitor the network traffic and machine learning models to learn the new behaviors of the attack.

4.2. Slow Header Attack. In normal scenario when a user requests data from the web server via the web browser using the GET request method, it sends a GET request completed by an empty line, two sequences of Carriage Return Line Feed (CRLF) [34]. Usually, the user' request takes a short time duration to be sent totally to the server. This time duration, called $\Delta_{request}$, depends on hardware and networking capabilities of the user. $\Delta_{request}$ is computed using the formula: $\Delta_{request} = t_{start\_request} - t_{end\_request}$. Then, the server provides the content as a web page. When the content is received, the user reads the content of the web page. After a time interval the user requests another web page based on links in the first page [35]. This time interval, called Δ_{next}, is the time gap between the end of receiving the server' response and the start of the next request in the same stream. Δ_{next} is computed using the formula: $\Delta_{next} = t_{start\_request_{next}} - t_{end\_response}$ [36].

The Slow Header attack also known as Slowloris attack is a sophisticated attack based on the GET request method [31]. The idea behind this attack is to keep alive a large number of connections to the target web server as long as possible in order to consume all its available resources. To achieve

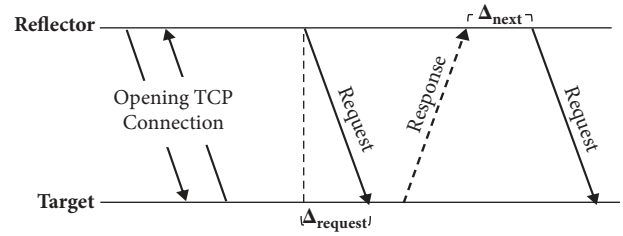

Figure 2: Sequential diagram of the Slow Header attack.

this the attacker sends incomplete HTTP GET requests to the victim by not transmitting the two CRLF sequences that denote the end of the HTTP header. Then a group of Reflectors is used to open normal connections with the target server and periodically send to it incomplete HTTP GET requests. Affected servers will keep these connections open, filling their maximum concurrent connection pool. This enables the attacker to deny connection requests of legitimate users. Therefore, this attack type is able to cause a DoS by a small quantity of packets and in a short period of time [5, 35, 37]. This attack is hard to detect and mitigate because it implies that the bandwidth consumption and the $\Delta_{request}$ lie in normal ranges but generates the attack continuously and repeatedly. The obvious behavior of the Slow Header attack is shown in Figure 2.

4.3. Slow Read Attack. The purpose of the Slow Read attack is to maintain as many as possible number of connections alive with the target server by slowly reading the replies received by the server.

To achieve this, first the attacker establishes a normal connection with the target server. Then the attacker keeps sending legitimate HTTP POST or GET requests to the target at a normal rate with a receiver window size of 0 byte telling the server that is not yet ready to receive more data. This forces the target web server to hold the connection and waits to receive a nonzero update window size from the client. Usually, attackers use Botnets to lunch the attack and tend to request large files in order to speed the DoS timing.

5. Impact of HTTP DDoS Attacks on a Cloud Environment

Cloud Computing services are often delivered through HTTP protocol. This means that the HTTP protocol' attacks, vulnerabilities, misconfiguration, and bugs have direct impact on the users services deployed on the Cloud. HTTP DDoS attacks are classified among the major threats of web services availability. Hence, they are a major threat of the Cloud services' availability.

In the Cloud Computing context, we distinguish two ways to achieve a DoS: direct implying which consists of predetermination of the target service's host and indirect implying which consists of denying other services being hosted on the same host or network of the target. The resources autoscaling characteristic of the Cloud enables, on one hand, the providers to supply the clients with a

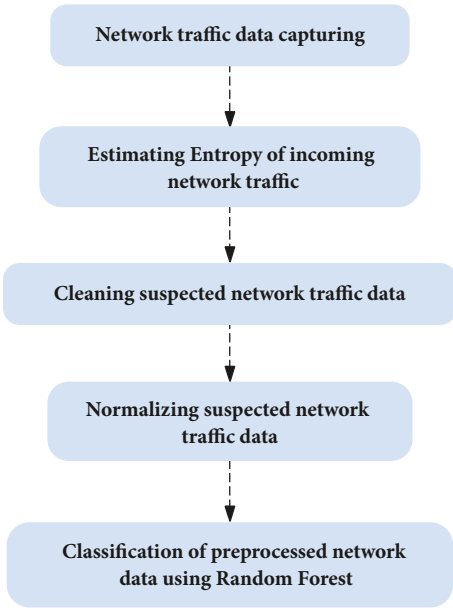

Figure 3: Layered architecture of the proposed detection system.

large pool of resources. The clients are, then, charged based on a pay-per-use model. On the other hand, this enables attackers to deny many Cloud services with a single attack. The detection of HTTP DDoS attacks in the Cloud requires a deep monitoring of the network traffic and strong modeling of the Cloud users' behaviors.

6. The Proposed Detection System

In this section the process followed to detect HTTP DDoS attacks as well as the components of the proposed detection system are given. The proposed detection system consists of three major steps: entropy estimation, data preprocessing, and network traffic classification. A time-based sliding window algorithm is used to estimate the entropy of the network header features of the incoming network traffic. When the average of the features' entropy exceeds its normal range the preprocessing module cleans and normalizes the network data traffic of the current time window. The preprocessed network traffic data is then classified into normal and HTTP DDoS traffic. A layered architecture of the entire proposed system is given in Figure 3. The layered architecture illustrates the five steps followed to detect the HTTP DDoS attacks and the main components of the proposed system.

A test procedure is used to select and to optimize appropriate classifier for HTTP DDoS detection based on accuracy, FPR, AUC, and running time metrics. The Random Forest ensemble classifiers depict high detection performance for HTTP DDoS attacks. The detailed process followed to detect HTTP DDoS attacks in the proposed system is illustrated in Figure 4.

6.1. Incoming Network Traffic Entropy Estimation. A time-based sliding window algorithm is used to estimate the entropy of a set of network flow-header features. By definition the entropy is a measure of the diversity or the randomness of a distribution, e.g., network flow data [38]. The analysis of the network flow entropy over time windows allows reducing high dimensionality of the network traffic distribution to a single metric describing its dispersion [39–41]. The connection definition (CD) features, the source/destination IPs and the source/destination Ports, are used to estimate the entropy. The reason for using the CD features is that during a HTTP DDoS attack the zombie hosts open a large number of connections with the victim. Hence, estimating entropy of the CD features allows detecting sensitively the abrupt changes in the network flow caused by HTTP DDoS attack. The Shannon formula is used to estimate the entropy of the CD features which is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log\left(p(x_i)\right) \quad (1)$$

where X represents a CD feature, $x_i (i = 1, \ldots, n)$ are the frequencies of items of X received during the current time window, and n is the total number of items of X. To better interpret the estimated entropy values, the CD features' entropy values are normalized using the formula: $H_0(X) = H(X)/\log n$. The pseudocode of the Entropy estimation time-based sliding window algorithm is given in Algorithm 1.

Algorithm 1 starts with setting the time window size. To efficiently maintaining the trade-off between resources consumption and accurately detecting abrupt changes in the network flow distribution, we picked a 2 minutes time window. On one hand, using a large time window is costly in terms of memory and the victim services may fall within it. Also, in the use of a large time window causes a smoothing of the estimated entropy which prevents seeing the abrupt changes in the network. On the other hand, using a small time window is costly in terms of computation which may late the detection of attack. More discussions about the time window size selection are given in Section 8.1.

Algorithm 1 gives as output the average of the computed entropy of the CD features set S. This allows us to use only two thresholds, lower (δ_l) and upper (δ_u), for each S in order to detect anomaly within the incoming network traffic. The lower (δ_l) and upper (δ_u) thresholds are predetermined form the first six hours of the CIDDS-001 dataset that contain only normal traffic. δ_l and δ_u correspond, respectively, to minimum and maximum estimated entropy of the normal traffic.

6.2. Network Traffic Preprocessing. Here, two main preprocessing tasks are performed on the network traffic data of the time window that have average entropy out of the normal range $[\delta_l, \delta_u]$. First the network traffic subset of each abnormal time window is cleaned by dropping rows that contains empty values and maintains unique format and columns number for rows in the dataset and unifying columns data types. Next the cleaned subsets are normalized using the *MinMax* method. In *MinMax* the values of features are scaled to the range $[0, 1]$ as follows:

$$x_i^{new} = \frac{x_i - \min(X)}{\max(X) - \min(X)} \quad (2)$$

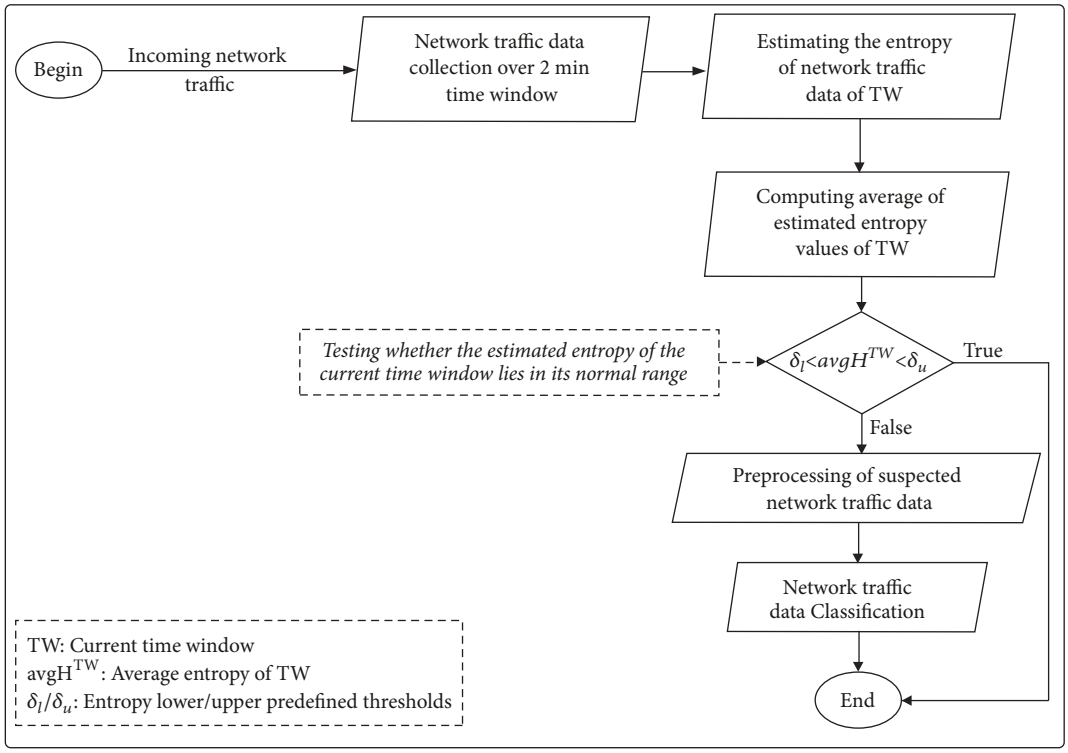

Figure 4: Flowchart of the proposed detection system.

```
1  avgEntropy (data, S)
   Input: Network traffic data data, Features set S
   Output: AverageEntropy
2  TW ← 2min
3  H_S ← {}
4  while IncomingNetworkTraffic in TW do
5      foreach X in S do
6          p(x_i) ← (∑_j^N x_i)/N    //Estimate the probability mass function
7          H(X) ← −∑_i^n p(x_i) · log p(x_i)    //Compute the Shannon entropy
8          H_0(X) ← H(X)/log n    //Normalize the entropy
9          H_S ← H_S ∪ H_0(X)
10     end
11     AverageEntropy ← sum(H_S)/4    //Return average entropy of the CD features
12 end
```

Algorithm 1: Estimate entropy of the connection definition features.

where X is a feature of the abnormal traffic subset, x_i is the current value of X to normalize, and x_i^{new} is the normalized value.

6.3. HTTP DDoS Attacks Detection. Here, we aim to classify each network traffic subset obtained from Algorithm 2 with the appropriate machine learning (ML) classifier. For this purpose five ML classifiers are fitted with 60% of the CIDDS-001 dataset and tested with the reminder 40%. The ML classifiers used here are Random Forest, Decision Tree

```
1  Preprocess (Abnormal network traffic subset)
   Input: Abnormal network traffic subset
   Output: Cleaned and normalized network traffic subsets
2  foreach Abnormal subset in Dataset do
3      Cleaning each column the suspected traffic data
4      Maintain unique data format for each column
5      Normalize values of each column
6  end
```

Algorithm 2: Preprocessing module algorithm.

(DT), K-Nearest Neighbor (KNN), Naive Bayes (NB), and Multilayer Perceptron (MLP). The appropriate classifier is selected based on accuracy, false positive rate, training time, and testing time metrics. The selected classifier is used to classify the incoming network traffic data of the time windows having abnormal entropy values. This allows filtering malicious HTTP traffic of each time window and speeding up the mitigation process of HTTP DDoS attacks.

7. Experiments

This section describes the experiments conducted and the performance metrics used to evaluate the proposed approach. The experiments aim to illustrate the impact of the time sliding window entropy estimation algorithm on the performance of the proposed approach for HTTP DDoS attacks detection. Also, other experiments are performed to select and optimize appropriate classifier for HTTP DDoS detection.

To validate the contributions of our approach the CIDDS-001 dataset [27] is used. A configuration of 60% for training and 40% for testing of the dataset is used in all experiments. The performances of the entire proposed approach are assessed and compared with the performances of different classifiers tested directly on the dataset without the time sliding window entropy estimation algorithm. Furthermore, the obtained results of the proposed system are compared with the state-of-the-art HTTP DDoS detection methods given in [14–18].

The proposed approach is developed using the Python programming languages and the ML algorithms are tested using their default parameters. The hardware used in the experiments is a core i3 2.4 GHz with 6 GB of memory running under Debian 8 x64.

7.1. Performance Metrics. The main purpose of the proposed approach is to classify the network traffic data as either positive or negative which correspond, respectively, to HTTP DDoS traffic and normal traffic. The obtained results are evaluated using the following performance metrics:

Accuracy: percentage of the traffic records that are correctly classified is as follows:

$$Accuracy = 100 \times \frac{correctly\ classified\ records}{total\ records} \quad (3)$$

False Positive Rate (FPR): percentage of the normal records which are classified as attack records is as follows:

$$FPR = 100 \times \frac{FP}{FP + TN} \quad (4)$$

where FP is the number of normal records incorrectly classified as attack records. TN is the number of correctly classified normal records.

Processing time: HTTP DDoS detection time depends on two time metrics, namely, training time and testing time.

ROC and AUC curves: Receiver Operator Characteristic (ROC) and Area under ROC (AUC) curves are commonly used to present results for binary decision problems in machine learning. The ROC curve shows how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. AUC is a criterion used to measure the quality of the classification model.

8. Results and Discussion

8.1. Estimated Network Entropy for CIDDS-001 Dataset. In this section we give the obtained results of the time sliding window entropy estimation algorithm on the CIDDS-001 dataset. The variations of estimating CD features entropy over 2-minute time window are presented in Figure 5. As shown in Figure 5 the estimated entropy during the first 180 time windows is in its normal range. The entropy estimation algorithm considers that this part does not contain HTTP DDoS attacks; hence no preprocessing or classification is triggered. It is worth noting that these results are in consistency with the CIDDS-001 dataset documentation, as mentioned in [27] that the first six hours of the dataset contains only normal traffic. Entropy estimation allows focusing only on suspected time windows that may contain HTTP DDoS attacks. As a result the amount of the network traffic data to preprocess and to classify in CIDDS-001 is drastically reduced; more than 58% of the test subset is reduced.

The performances of the network entropy estimation algorithm rely mainly on the time window size. To maintain the trade-off between resources consumption and accurately detecting abrupt changes in the network flow distribution, we picked a 2-minute time window. On one hand, using a large time window is costly in terms of memory. Also, large time windows are not suitable when dealing with HTTP DDoS attacks because the victim services may fall within the time window. Moreover, a major issue facing the use of large time windows is that they cause smoothing of the estimated entropy curves which prevent seeing the abrupt changes in the network traffic. This effect is illustrated in Figure 6, as the time window size increases the pikes of the entropy curves are smoothed and reduced causing low efficiency of the algorithm. On the other hand, using a small time window is costly in terms of computation.

8.2. Classifier Selection for HTTP DDoS Detection. Here, we give the results of the classifier selection procedure for HTTP DDoS attacks detection. To obtain these results the five classifiers mentioned in Section 6.3 are fitted and tested with the CIDDS-001 dataset. The classification results are illustrated in Figure 7. As shown in Figure 7 Random Forest achieves the highest AUC of 0.969 followed by NB classifier with an AUC of 0.845. The remaining classifiers KNN, MLP, and DT achieve low AUC of respectively 0.708, 0.577, and 0.436. More results of the comparison of different classifiers for HTTP DDoS attacks detection are tabulated in Table 2. It is obvious that the Random Forest classifier achieves the highest accuracy of 97% with a false positive rate of 0.33%. The accuracy of Random Forest exceeds NB accuracy by 3% and deceeds its FPR by 0.01%. The accuracy of NB, KNN, DT, and MLP classifiers for HTTP DDoS detection is, respectively, 94%, 86%, 73%, and 28%. The obtained results depict that Random Forest outperforms other tested classifiers in terms

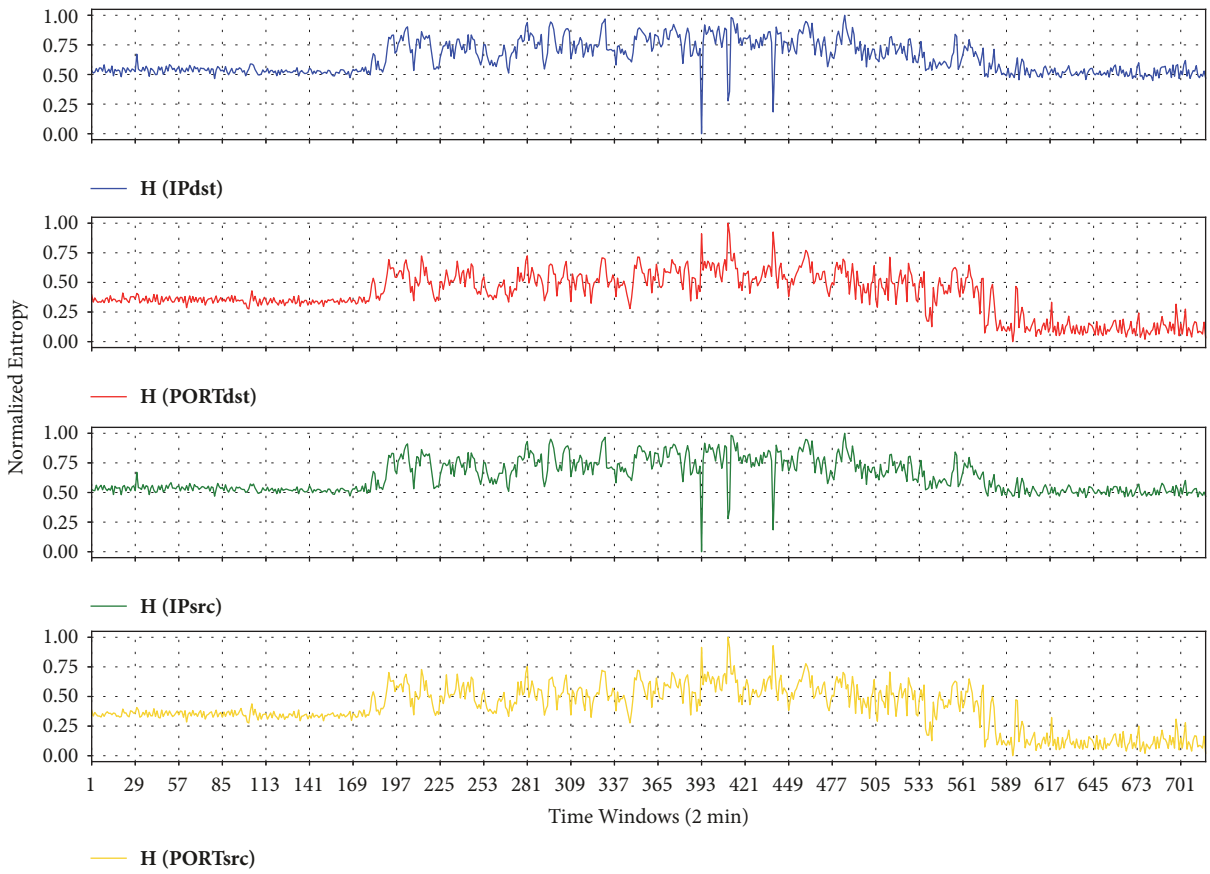

Figure 5: Estimated entropy time series variations of the CIDDS-001 dataset.

Table 2: Comparison of different classifiers for HTTP DDoS detection.

Classifier	Accuracy(%)	FPR(%)	Train Time (s)	Test Time (s)
Random-Forest	97	0.33	1.23	0.18
NB	94	0.34	0.31	0.16
KNN	86	0.36	1270.56	108.16
DT	73	0.38	1.35	0.11
MLP	28	0.6	321.26	2.2

of accuracy and false positive rates. However, in terms of running time NB classifier achieved the best timing with 0.16s for training and 0.31s for testing. The Random Forest classifier depicts also high running time with 1.23s for training and 0.18s for testing.

8.3. Entire Proposed Approach. In this section the results of the entire proposed approach are given. The results represent the variations of three performance metrics collected over the time windows classified by network traffic entropy estimation algorithm as abnormal. The collected results are accuracy, FPR, and running time. To obtain these results the network traffic data captured during each abnormal time window is preprocessed and classified using the Random Forest ensemble classifiers.

Figure 8 gives the accuracy variations of the entire proposed approach over the abnormal time windows. It is obvious that the proposed approach achieves high accuracy for the majority of the time windows. This is due to the entropy estimation algorithm that reduced large amount of normal network traffic data for classification. For HTTP DDoS detection this normal traffic data is considered as noisy and irrelevant. The average accuracy obtained for all the time windows is 99.54%. An important improvement of the accuracy of 2.54% is noticed here compared to the accuracy of Random Forest tested directly on the CIDDS-001 which is 97%.

Figure 9 shows the FPR variations of the proposed approach over the abnormal time windows. We notice low FPR for the most abnormal time windows except for the 20^{th} to the 28^{th} time windows that have nearly a FPR of 0.7%. This increase in the FPR during these abnormal time windows is accompanied by an important decline in the accuracy as it is obvious in Figure 8. This may be caused by the increase

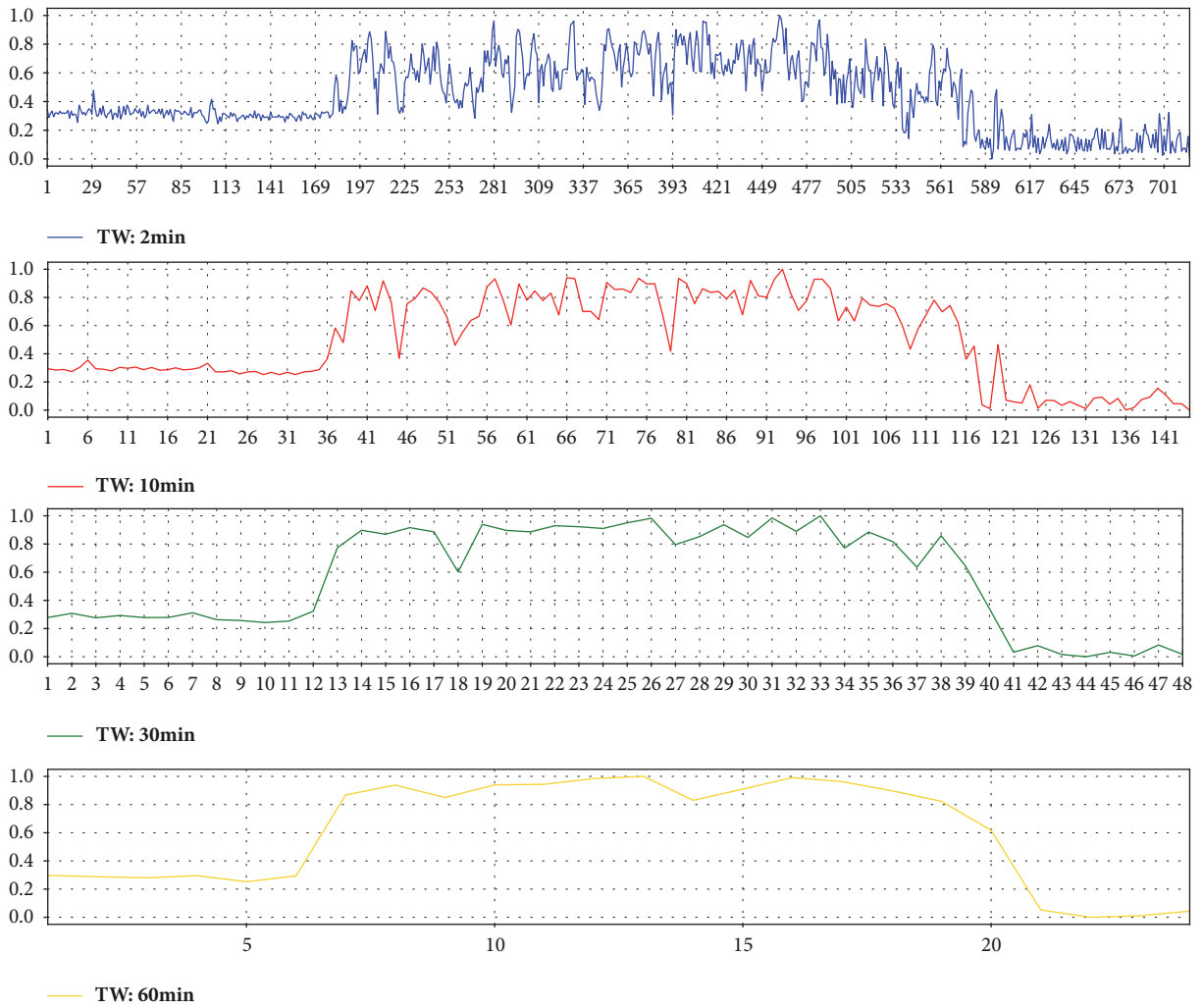

Figure 6: Estimated entropy variation of different time window sizes.

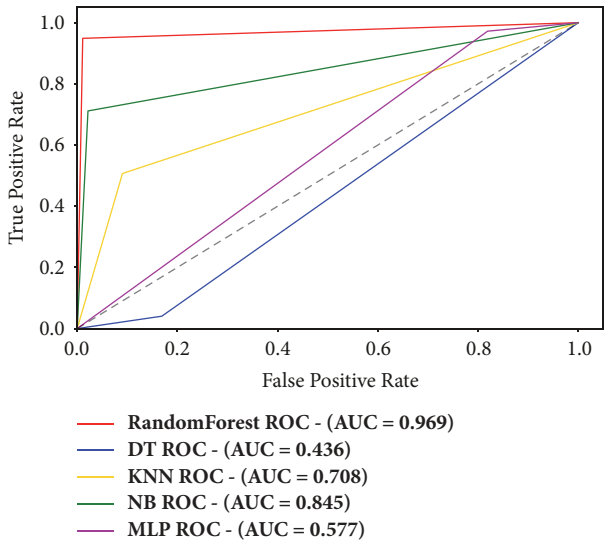

Figure 7: Obtained results of different classifiers trained and tested on the CIDDS-001 dataset.

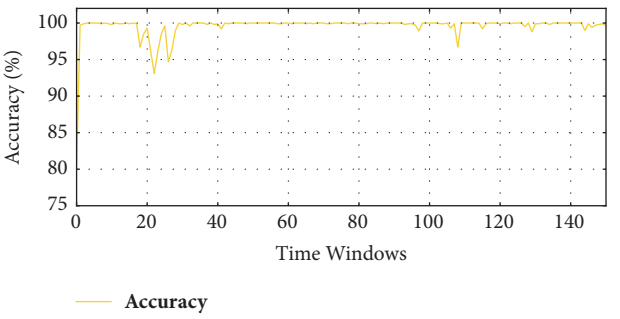

Figure 8: Accuracy variations of the proposed approach over time windows.

of the normal and noisy traffic data during this period. However, the FPR continues to fluctuate for the remaining time windows. The average FPR for all the abnormal time windows is 0.4% which is acceptable when compared to the FPR of the different tested classifiers as shown in Table 2.

Table 3: Comparison of the proposed approach with the state-of-the-art HTTP DDoS detection approaches.

Approach	Dataset	% Training	% Testing	Accuracy (%)
Our approach	CIDDS-001	60	40	99.54
Approach of [14]	Local/Intern	80	20	99.73
Approach of [15]	CAIDA	60	60	94.8
Approach of [16]	Local	66	34	98.89
Approach of [17]	Local	NA	NA	90.11
Approach of [18]	CDX/Local	NA	NA	92.63

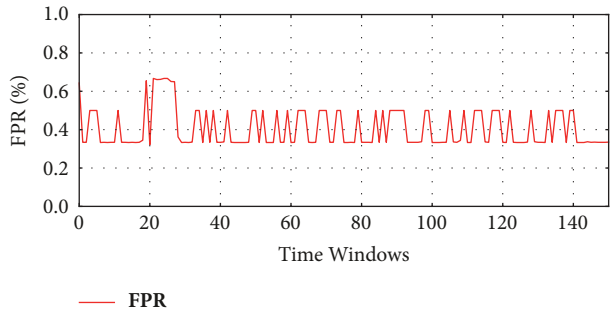

Figure 9: False positive rates variations of the proposed approach over time windows.

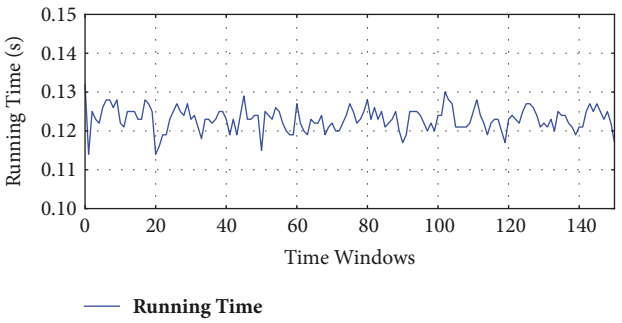

Figure 10: Running time variations of the proposed approach over time windows.

Running time is an important performance metrics when dealing with DDoS attacks in general. Figure 10 shows the variations over time windows of the running time of the entire proposed approach including the entropy estimation, the preprocessing and the classification steps. As Figure 10 shows the running time continues to fluctuate for all the abnormal time windows. When a HTTP DDoS attack occurs the amount of network traffic data increases causing an increase in the entropy estimation, preprocessing, and classification time. The total running time for all the abnormal time windows is 18.57s. However, an average running time per time window is, time to process network data collected during each time window, of 0.12s is noticed for proposed approach. The obtained average running time is important when compared with running time of other tested classifiers as shown in Table 2. Also, this average running time of the proposed approach is very important because the HTTP DDoS mitigation and filtering actions can be taken early for each time window.

8.4. Validation of the Obtained Results. To validate the obtained results, the proposed approach is compared with the state-of-the-art HTTP DDoS detection methods given in [14–18]. For this purpose we used 60% of the dataset for training and 40% for testing as it is the commonly datasets splitting configuration used in the literature. The approaches are compared based the dataset splitting configuration used and the overall accuracy. The comparison results are summarized in Table 3.

It is clearly shown in Table 3 that the proposed approach achieves the highest accuracy of 99.54% when compared with other approaches evaluated using 60% of the dataset for training and 40% for testing. However, the highest accuracy is achieved by the approach proposed in [14]. This high accuracy may be due to the configuration used to split the dataset adopted by the authors. Compared to the state of the art the results obtained by the proposed approach are satisfactory.

9. Conclusion

In this paper a detection system of the HTTP DDoS attacks in a Cloud environment is proposed. HTTP DDoS are sophisticated attacks that highly mimic the normal behaviors with completely different strategies. During such attacks large amounts of network traffic are generated. The detection of these attacks have becomes very hard and new techniques are required. For these reasons we have proposed a detection system based on three techniques, namely, network traffic entropy estimation, preprocessing, and classification using Random Forest ensemble classifiers.

Various experiments were conducted in order to evaluate our approach and satisfactory results are obtained. Five machine learning classifiers are used to conduct the experiments on the CIDDS-001 public dataset. The entire proposed system achieved high detection performances compared to single classifiers tested directly on the dataset and to the state-of-the-art HTTP DDoS detection methods.

Despite the fact that the proposed approach depicts high HTTP DDoS attacks detection performances with the CIDDS-001 public dataset, it is important to evaluate its performances in real world scenarios. For future work, we are planning to perform real world deployment of our approach and evaluate it against several HTTP DDoS tools.

Conflicts of Interest

The authors declare that they have no competing financial interests.

Authors' Contributions

Mohamed Idhammad proposed the main approach to detect the HTTP DDoS attacks. Also, Mohamed Idhammad has designed the preprocessing algorithm and implemented the entire detection method in Python. Karim Afdel attributed to the design and the optimization of the proposed method and also has contributed to the assessment methodology. Mustapha Belouch has contributed to the design and the implementation of the proposed approach.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," *Cloud Computing and Government: Background, Benefits, Risks*, pp. 171–173, 2011.

[2] Wikipedia, "2016 dyn cyberattack," 2016, https://en.wikipedia.org/wiki/2016_Dyn_cyberattack.

[3] theguardian, "Ddos attack that disrupted internet was largest of its kind in history, experts say," 2016, https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

[4] E. Cambiaso, G. Papaleo, and M. Aiello, "Taxonomy of Slow DoS Attacks to Web Applications," in *Recent Trends in Computer Networks and Distributed Systems Security*, vol. 335 of *Communications in Computer and Information Science*, pp. 195–204, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[5] T. Vissers, T. S. Somasundaram, L. Pieters, K. Govindarajan, and P. Hellinckx, "DDoS defense system for web services in a cloud environment," *Future Generation Computer Systems*, vol. 37, pp. 37–45, 2014.

[6] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004.

[7] S. Yu, *Distributed Denial of Service Attack and Defense*, SpringerBriefs in Computer Science, Springer, New York, NY, USA, 2014.

[8] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[9] R. K. C. Chang, "Defending against flooding-based distributed denial-of-service attacks: a tutorial," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 42–51, 2002.

[10] J. Kristoff and R. Joffee, "Botnets and Packet Flooding DDoS Attacks on the Domain Name System," *The International Journal of Forensic Computer Science*, pp. 9–18, 2007.

[11] M. Anagnostopoulos, G. Kambourakis, and S. Gritzalis, "New facets of mobile botnet: architecture and evaluation," *International Journal of Information Security*, vol. 15, no. 5, pp. 455–473, 2015.

[12] P. Farina, E. Cambiaso, G. Papaleo, and M. Aiello, "Are mobile botnets a possible threat? the case of SlowBot Net," *Computers & Security*, vol. 58, pp. 268–283, 2016.

[13] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for Network Intrusion Detection Systems using Distance-based Machine Learning," *Procedia Computer Science*, vol. 125, pp. 709–716, 2018.

[14] Z. He, T. Zhang, and R. B. Lee, "Machine Learning Based DDoS Attack Detection from Source Side in Cloud," in *Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 114–120, New York, NY, USA, June 2017.

[15] I. Sreeram and V. P. Vuppala, "HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied Computing and Informatics*, 2017.

[16] I. Sofi, A. Mahajan, and V. Mansotra, "Machine learning techniques used for the detection and analysis of modern types of ddos attacks," *learning*, vol. 4, no. 06, 2017.

[17] Q. Liao, H. Li, S. Kang, and C. Liu, "Application layer DDoS attack detection using cluster with label based on sparse vector decomposition and rhythm matching," *Security and Communication Networks*, vol. 8, no. 17, pp. 3111–3120, 2015.

[18] T. R. Sree and S. M. S. Bhanu, "HADM: detection of HTTP GET flooding attacks by using Analytical hierarchical process and Dempster–Shafer theory with MapReduce," *Security and Communication Networks*, vol. 9, no. 17, pp. 4341–4357, 2016.

[19] N. Gonzalez, C. Miers, F. Redigolo et al., "A Quantitative Analysis of Current Security Concerns and Solutions for Cloud Computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, pp. 231–238, 2012.

[20] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring internet denial-of-service activity," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 115–139, 2006.

[21] J. McHugh, "Intrusion and intrusion detection," *International Journal of Information Security*, vol. 1, no. 1, pp. 14–35, 2001.

[22] S. Lee, G. Kim, and S. Kim, "Sequence-order-independent network profiling for detecting application layer DDoS attacks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, article no. 50, 2011.

[23] Y. G. Dantas, V. Nigam, and I. E. Fonseca, "A Selective Defense for Application Layer DDoS Attacks," in *Proceedings of the 2014 IEEE Joint Intelligence and Security Informatics Conference (JISIC)*, pp. 75–82, The Hague, Netherlands, September 2014.

[24] J. Choi, C. Choi, B. Ko, and P. Kim, "A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment," *Soft Computing*, vol. 18, no. 9, pp. 1697–1703, 2014.

[25] M. Aiello, E. Cambiaso, M. Mongelli, and G. Papaleo, "An on-line intrusion detection approach to identify low-rate DoS attacks," in *Proceedings of the 2014 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6, Rome, Italy, October 2014.

[26] L. Xiao, W. Wei, W. Yang, Y. Shen, and X. Wu, "A protocol-free detection against cloud oriented reflection DoS attacks," *Soft Computing*, vol. 21, no. 13, pp. 3713–3721, 2017.

[27] M. Ring, S. Wunderlich, D. Gr, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the in Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS). 1em plus 0.5em minus 0*, pp. 361–369, 2017.

[28] NA. Http flooder, 2010, https://code.google.com/archive/p/httpflooder/wikis/Usage.wiki.

[29] R. Security, *DDoS Survival Handbook The ultimate guide to everything you need to know about DDoS attacks*, 2015.

[30] N. Tripathi and N. Hubballi, "Slow rate denial of service attacks against http/2 and detection," *Computers Security*, vol. 72, pp. 255–272, 2018.

[31] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, "Slow DoS attacks: definition and categorisation," *International Journal of Trust Management in Computing and Communications*, vol. 1, no. 3-4, pp. 300–319, 2013.

[32] Owasp, "(2010) Owasp http post tool," 2010, http://www.owasp.org/images/4/43/Layer7DDOS.pdf.

[33] E. Damon, J. Dale, E. Laron, J. Mache, N. Land, and R. Weiss, "Hands-on denial of service lab exercises using SlowLoris and RUDY," in *Proceedings of the the 2012 Information Security Curriculum Development Conference*, pp. 21–29, Kennesaw, Georgia, October 2012.

[34] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC Editor RFC7230, 2014.

[35] Y. Choi, I. Kim, J. Oh, and J. Jang, "AIGG Threshold Based HTTP GET Flooding Attack Detection," in *Information Security Applications*, vol. 7690 of *Lecture Notes in Computer Science*, pp. 270–284, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[36] M. Aiello, E. Cambiaso, S. Scaglione, and G. Papaleo, "A similarity based approach for application DoS attacks detection," in *Proceedings of the 2013 IEEE Symposium on Computers and Communications (ISCC)*, pp. 000430–000435, Split, Croatia, July 2013.

[37] H. Kim, B. Kim, D. Kim, I. Kim, and T. Chung, "Implementation of GESNIC for Web Server Protection against HTTP GET Flooding Attacks," in *Information Security Applications*, vol. 7690 of *Lecture Notes in Computer Science*, pp. 285–295, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[38] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *Computer Communication Review*, vol. 35, no. 4, pp. 217–228, 2005.

[39] A. Wagner, A. Wagner, B. Plattner, and B. Plattner, "Entropy Based Worm and Anomaly Detection in Fast IP Networks," in *Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, pp. 172–177, Linkoping, Sweden.

[40] T. Liu, Z. Wang, H. Wang, and K. Lu, "An entropy-based method for attack detection in large scale network," vol. 7, pp. 509–517, 2014.

[41] M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for DDoS detection," *Applied Intelligence*.