Journal of King Saud University – Computer and Information Sciences 32 (2020) 73–87

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

BARTD: Bio-inspired anomaly based real time detection of under rated App-DDoS attack on web

K. Munivara Prasad [a,*], A. Rama Mohan Reddy [b], K. Venugopal Rao [c]

[a] Department of CSE, JNTUH, Hyderabad, Telangana, India
[b] Sri Venkateswara University College of Engineering, Sri Venkateswara University, Tirupati, Andhra Pradesh, India
[c] GNITS, Shaikpet, Hyderabad, Telangana, India

ARTICLE INFO

Article history:
Received 30 March 2017
Revised 15 June 2017
Accepted 11 July 2017
Available online 14 July 2017

Keywords:
Denial of Service (DoS) Attacks
Distributed DoS (DDoS) Attacks
Application Layer DDoS (APP-DDoS)
Bio inspired approaches

ABSTRACT

The internet network is mostly victimized to the Distributed Denial of Service (DDOS) Attack, which is one that intentionally occupies the computing resources and bandwidth in order to deny that services to potential users. The attack scenario is to flood the packets immensely. If the attack source is single, then the attack is referred as denial of service (DOS) and if attack is sourced from divergent servers, then it is referred as DDOS. It is imperative from the analysis that there are constraints in the existing models since the most of these models are user session based and/or packet flow patterns. The session based evolution models are vulnerable to botnets and packet flow pattern based models are vulnerable if attack sources are equipped with human resource and/or proxy servers. Hence, there is inherent need for improving the solutions towards addressing the App-DDoS attacks over the system. The crux for such system is about ensuring that fast and early detection with minimal false alarming in streaming network transactions, and ensures that the genuine requests are not impacted. To address such a system, the model of Bio-Inspired Anomaly based App-DDoS detection aimed, and the proposed model depicted in detail along with experimental inputs. Results attained from the process exemplify the significance and robustness of the model towards achieving the objectives considered for the solution.

1. Introduction

Global network of computers interconnected through different media using a standard protocol is called internet. Modern human beings rely on the Internet for their education, trade, socialization and entertainment, among many other important aspects of human life. Information sharing, E-commerce and entertainment have taken a new dimension. Evidently, the Internet is the biggest revolution in the computing and communications world. Web threats pose a broad range of risks, including financial damages, identity theft, loss of confidential information or data, theft of network resources, damaged brand/personal reputation, and erosion of consumer confidence in e-commerce and online banking.

DoS attack is an intentional attempt by malicious users to completely disrupt or degrade the availability of services/resources to legitimate users. Distributed denial of service (DDoS) attack is a form of DoS attack which slowdowns the server in responding to the client/refuses the client request. Now-a-days, the impact of DDoS attacks on internet security is growing excessively. In general, this type of attack is launched explicitly from a collection of compromised systems known as botnet by an attacker. The main goal of such attack is to exhaust server resources such as CPU, I/O bandwidth, sockets and memory etc. As the result, the resources available to other normal users/clients get limited or sometimes may not be available. The recent familiar victims of DDoS attack are explored in Udhayan and Anitha (2009) and Chun-Tao et al. (2012) and strategies for successful attack mitigating are explored in Lee (2004).

According to (Kumar and Nene, 2013), the DDoS attacks are classified based on different factors. On the basis of network protocol stack, DDoS can be further classified as Network/transport level and Application level DDOS attacks.

* Corresponding author.
E-mail addresses: prasadkmv27@gmail.com (K.M. Prasad), ramamohansvu@gmail.com (A. Rama Mohan Reddy), kvgrao123@gmail.com (K.V. Rao).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2017.07.004
1319-1578/

Network/*transport level DDoS attack*: These attacks are launched at half opened connections by using TCP, UDP, ICMP and DNS protocols.

Application level DDoS attack: These attacks typically consume less bandwidth and are stealthier in nature in comparison to volumetric attacks. However, they will have an identical impact to service as they aim specific characteristics of well-known applications like HTTP, DNS, VoIP or simple Mail Transfer Protocol (SMTP). These attacks are specialize in disrupting legitimate users services by exhausting the resources. An Application layer DDoS attack overloads an application server by creating excessive login, information search or search requests. Application DDoS attacks are tougher to detect than other forms of DDoS attacks. As the connections are already established, the requests could seem to be from legitimate users. However, once identified, these attacks will be stopped and back-traced to source more simply than the other varieties of DDoS attacks.

Application Layer DDoS attack is a DDoS attack that sends out requests following the communication protocol, thus these requests are indistinguishable from legitimate requests in the network layer .Consequently, traditional defense systems become less or even not applicable for application layer DDoS attacks which make use of the asymmetric computation between client and server, as they are proper-looking requests from the protocol and traffic.

Flooding attacks: Flooding attacks are launched in following ways (Wang et al., 2015):

1.1. In reflection/amplification based flooding attacks

The attacker initiates small DNS queries with forged source IP addresses which provoke a large extent of network traffic. And the DNS response messages are significantly larger than DNS query messages. As the result, this large extent of network traffic is directed towards the targeted system to incapacitate it.

1.2. HTTP based flooding attacks are classified into four types

In Session flooding attack, the Session connection request rates initiated from the attackers are higher the requests generated from legitimate users. Thus the server resources are exhausted and lead to flooding**.**

In Request flooding attack, the attacker send sessions that contains more number of requests than the normal users, which leads to flooding.

In Asymmetric attack, the attacker sends sessions that contains larger amount of high workload requests. The ultimate aim of the attacker is to devour resources like CPU, memory of the server and degrade it.

In Slow request/response attack, the attacker sends HTTP request in pieces slowly (one at a time) and the request is not complete initially. As the result, the server keeps the indulged resources in waiting stage until it receives the entire data. This attack is categorized into slowloris attack, HTTP fragmentation attack, slow post attack and slow reading attack.

The major focus of an HTTP flood DDoS attack is toward generating attack traffic that closely simulates legitimacy of a human user. Thereby it becomes harder for a victim to differentiate between legitimate and attack traffic. Because of this type of attacks, the server becomes unavailable to legitimate users. The main impact of application layer DDoS attacks are:unusually slow network performance (opening files or accessing web sites), unavailability of a particular web site, inability to access any web site, dramatic increase in the number of spam emails received.

2. Related work

The recent escalation of application layer DoS attacks have attracted a significant interest of a research community. Since application layer attacks usually do not manifest themselves at the network level, they avoid traditional network based detection mechanisms. As such, security community focused on specialized application-layer DoS attacks detection mechanisms. These research efforts can be broadly divided into several groups: application-based, puzzle-based approaches and network traffic characteristics based.

Application-based techniques are generally geared toward legitimate and thus expected characteristics of an application behavior. These approaches include detection of deviations from normal behavior of users browsing web pages (Xie and Zheng Yu, 2006; Yu et al., 2007; Ye and Zheng, 2011; Ranjan and et al., 2006), monitoring characteristics of HTTP sessions (Ranjan et al., 2009; Xuan et al., 2010), monitoring a number of clients requests (Xie and Zheng Yu, 2009), and analyzing popularity of certain websites (Mehra et al., 2011). In many of these approaches, rate-limiting serves as a primary defense mechanism.

Puzzle-based methods are similar to these approaches. However, instead of monitoring characteristics of particular applications, puzzle-based methods, as the name suggests, offer a puzzle to solve and detect potential DoS attack by the ability of the client at the IP addresses to solve it or by their reaction to the offered puzzle. One of these techniques is the detection of attacks using CAPTCHA puzzle (Mori and Malik, 2003). Although this technique may offer a simple approach to attack detection and mitigation, a number of studies showed its ineffectiveness (Jung et al., 2002; Nam and Lee, 2009).

Monitoring characteristics of *network traffic for application-layer* DoS detection has been suggested by Bhatia et al. (2012) and has been employed for differentiation of flash crowd and true DoS attacks. The approach has also found its application in several studies in a form of IP address monitoring (Tang, 2012; Maciá-Fernández et al., 2010). Most of these studies deal with general type application-layer denial-of-service attacks. With the introduction of low-rate application-layer DoS attack, a number of research efforts were focused on various detection and mitigation techniques (Maciá-Fernández et al., 2007; Macia-Fernandez et al., 2009; Salmen et al., 2015; Zolotukhin et al., 2016). Most of these techniques focus specifically on characteristics of incoming network traffic aiming to reveal/prevent patterns specific to low-rate DoS attacks. As such Tang (Maciá-Fernández et al., 2007) developed a CUSUM-based approach that monitors packet arrival rate. Macia-Fernandez et al. proposed to modify the implementation of application servers in terms of their processing of incoming requests (Macia-Fernandez et al., 2009).

Fadir Salmen et al. (2015) created digital signature of network phase for flow analysis by using two meta-heuristic approaches. Here DSNSF contains information like packets per second and bits per second. Two meta-heuristics approaches are firefly algorithm and genetic algorithm created the digital signature by aggregation information using sFlow pattern from the State University of Londrina (UEL). To investigate the behavior of planned approaches they injected abnormal traffic and showed improved accuracy in detection DDoS attacks however the primary model is incapable for detection the DoS attacks. The results shown during this aren't important in acquiring the high detection rate constantly for DoS and DDoS attacks.

Mikhail Zolotukhin et al. (2016) proposed a model that notice intermediate and trivial application layer DDoS attacks that are in sort of encrypted network traffic. Anomaly detection primarily based approach is applied to extracted information from the

network packets. The conversations between server and its client are determined and divided into some clusters and analyzed how conversations initiated by one client to the server throughout some short time interval are distributed among these clusters. The practical cyber surroundings that generated realistic traffic patterns of end users are used to check the proposed approach. Still the results were satisfactory, further can be improved by using the machine learning approach.

Kambouakis et al. (2007) proposed mechanism DAAD (DNS Amplification attack detector) to defend against reflection/amplification flooding attack where the DNS response messages are larger than the request. On the receipt of DNS message, the DAAD engine determines whether the received message is a response or request; creates a new entry in request/response table. Once the message is identified as response the DAAD checks for the existence of the corresponding request. If the response doesn't match with the request logged already within time frame, then that response is marked as suspicious. An alert is generated when the alert count exceeds the threshold and the attacker is blocked. This mechanism is easy to deploy but the fact here is the database size would increase rapidly in case of high traffic rate.

Vijayalakshmi and Mercy Shalinie (2012) proposed IP Traceback defense mechanism used to detect both network layer and application layer attacks. It comprises of three functions: Attack detection, Hybrid IP traceback and DDoS mitigation component. In attack detection, packet header is analyzed by generating histograms and later payload is analyzed. Under feature selection of attack detection function, IP address used to identify DDoS flooding attacks and payload used to identify non flooding application layer attacks. The hybrid IP traceback function comprised of Packet marking (IP address is fragmented and marked), Reconstruction procedure (two phases: address identification and address recovery) and Attacker's source identification using entropy (Entropy variation is calculated). In mitigation component, when attack is detected an alert file generated.

Yu et al. (2010) proposed TMH (Trust Management Helmet) which is light weight mechanism uses trust management to differentiate normal (legitimate) user and attackers. It records four aspects of user's trust based on access history: short-term trust, long-term trust, negative trust and misusing trust .These values are stored as part of a license at clients and also the list of clients with lower trust value will be entered in the black list with an expiration time. A license is comprised of: user identification (such as ID, IP address) and trust computation, which is cryptographically secured. The main purpose of license is to identify whether the user is legitimate or attacker.

Ranjan et al. (2009) proposed DDoS-Shield which is a counter mechanism to protect web servers from application layer attacks. It consists of two parts: a suspicion assignment mechanism and a DDoS-resilient scheduler. The suspicion assignment uses session history to accredit a suspicion measure to every client session. The DDoS resilient scheduler determines which session is granted to forward requests and when, relying on the scheduling policy and scheduler service rate.

Wen et al. (2010) proposed CALD is a defense mechanism to protect web servers against application layer DDoS attacks that pretend as flash crowds. CALD consist of three functions: abnormal traffic detection, DDoS attack detection and filter. In the abnormal traffic detection, the function is deployed at the front-end sensor. And the ATTENTION signal is initiated when observed behavior and model output differs. The DDoS attack detection component is activated when it receives ATTENTION signal from front end sensor. And it traces each incoming source IP address as well as visiting webpage, records average frequency and calculates the entropy based on the average frequency. The anomalous source IP (identified based on threshold of entropy) is sent to filter so that it can defend the attack.

Liu and Chang (2011) proposed DAT (Defending Systems Against Tilt DDoS Attacks) is built with two coordinated defenders namely In/egress filter (IF) and Behavior Analyzer (BA). These two defenders comprised of four functions: connection handling, packet filtering, and behavior monitoring and counter-attack mechanism. The connection handling mainly deals with filtering attacks of connection establishing protocols, such as incomplete connections, etc. Packet filter is to block the packets/requests from malicious users (identified with help of black list given by behavior monitor). Behavior monitoring analyzes each user behavior in different aspects and determines the degree of deviation of the user. The counter-attack mechanism offers different services to each user depending on their degree of deviation.

Yu et al. (2007) proposed DOW (Defense and Offense Wall) defense mechanism is integration of detection technology and currency technology. The anomaly detection method (detection technology) used to detect and defend asymmetric attacks and request flooding attacks. The encouragement model (currency technology) used to defend session flooding attack. The anomaly detection method is comprised of three phases: Training, Detecting and Filtering. In First phase, it uses K-means clustering to build normal client behavior profile. In second phase, a cluster distance based method used to detect the attack. In third phase, the filter expels the suspicious sessions based on the trust value of the session. The encouragement model encourages the session expelled by anomaly detection method if it is legitimate. That is encourages the users to resend the session connection requests.

Xiang et al. (2011) proposed a generalized entropy metrics and information distance metrics to detect low rate application layer ddos attack. This is a router based solution and requires control of all routers in the network. Xuan et al. (2010) proposed a Group-Testing based approach. It is deployed on the back end of the server. It creates virtual servers. A set of group client requests are distributed to each virtual server. It identifies an attacker by testing the client in the group using a dynamic threshold. Creating a virtual server can cause a large overhead on the server and it also increases response time.

Prasad et al. (2016) defined machine learning strategy called Anomaly based Real Time Prevention (ARTP) of under rated App-DDoS attacks. Features have to be extracted at absolute time interval rather than request level in order to identify whether the traffic contains attack packets by using the defined thresholds. The proposal is tested against benchmark dataset LLDOS dataset. The complexity of the process reduced and attained maximum detection accuracy compared to other existing machine learning approaches. The results are good but still it can be improved further.

Senthilnath et al. (2011) explored the use of firefly algorithm for clustering. Local Minima is obtained by using the k-means clustering and this drawback was overcome by the firefly algorithm. Global Optima is obtained by using the randomization parameter and nature of attractiveness in clustering process. The lesser brightness fireflies move towards the brighter ones and all the fireflies are assumed to be of same gender. Classification was done with the help of Classification Error Percentage (CER). The results are compared mainly with two other bio inspired approaches and also given the accuracy rate chart for almost other nine different approaches. The authors used k-means and firefly algorithm for clustering purpose which increases the time complexity and this is not applicable for detecting application layer DDoS attacks in real time analysis.

Hadoop based DDoS detection framework (Hameed and Ali, 2015) is adapted for detecting the DDoS flooding attacks. The proposed model devised using the network traffic capturing and log

generation, detection and result process, captures packets using Tshark and gathers necessary information. Based on information gathered, creates a log file and shares the information to detection server through traffic handler. Map Reduce algorithm conducts the task of filtering and sorting. The counter based algorithm detection uses the attacks based on threshold. The phase of capturing consumes more time in the process of attack detection time.

A novel HTTP flood attacks detection system (Choi et al., 2013) proposed for cloud environment. The system uses the method of misuse detection mechanism for identifying malicious packets. Then, it uses a threshold mechanism, followed by anomaly detection mechanism for attack detection. Both the processing and attack detection are powered using Map Reduce Processing.

A contemporary framework for Distributed DoS attack detection (Choi et al., 2014) proposed to detect DDoS attacks based on the rule engine that generates rules by the packet flow patterns learned. This system meant for cloud computing platform that uses Map Reduce algorithm for quick processing and detection of attacks. The model comprises packet and log collection, pattern analysis system and a detection module. Such method adapts entropy statistical method for detecting intrusions.

In Abdul et al. (2011), the model of Intelligent IDS proposed that based on ontology. The proposed system thwarts ontology based on encoding scheme, port number, system component, policies, and attack type. The proposed model extracts the HTTP packet and parses it. The system emphasize on usage of semantic rules for detecting DoS, buffer over row, XSS cross-site scripting and SQL injection kind of attacks. However, the model is vulnerable to DDoS attacks

The past HTTP transactions observed and labeled as Normal or Flood at server gateway are used to train the proposed model. The characteristics of these transactions are extracted as features to preprocess the dataset, which are used further to train and detect.

3. Bio inspired anomaly based real time App-DDoS attack detection

The past transactions in application layer observed and labeled as Normal or Flood at server gateway are used to train the proposed model. The characteristics of these transactions are extracted as features to preprocess the dataset, which are used further to train and detect.

3.1. The exploration of the features considered to train and test the model

In Prasad et al. (2017) metrics were defined to prepare stream level traffic into session level time intervals called absolute time intervals (ati) or time frames. The detailed exploration of the constraints observed in existing contemporary models, which are stated in related work (see Section 2), it is obvious to state that, in distributed environment, diversified packet flow is easy to achieve through minimal time frames and session time. The arrival rate based on human users, including a proxy server seems to constitute the non-pattern (random) cases. Hence, to challenge this constraint, this manuscript devised a novel set of metrics, which are derived from absolute time interval rather than the session time and packet patterns.

3.1.1. Absolute time interval

This denotes the absolute time taken by the set of sessions initiated at given threshold time frame. This feature considered as significant, as HTTP-flood is cumulative of multiple sessions and diversified packet flow. The features explored further for defined absolute time interval.

3.1.2. Absolute session count

This feature represents the average number of sessions found in an absolute time interval defined. This feature is considered since the load on any target webserver estimated by the number of sessions in a given time interval.

3.1.3. Absolute session interval

This feature represents the average time render each session in an absolute time interval defined. This feature is critical as the session time indicates the time spent by a source on the target webserver with an intension of fair use or an attack.

3.1.4. Absolute page access count

This feature represents the average number of requests in an absolute time interval defined. This feature also critical one among the considered features, since the page access count along with absolute session interval optimizes the detection of the load on target webserver.

3.1.5. Absolute page access time

This feature represents the average time spent on each page request in an absolute time interval defined. The motive to consider this feature is, load of requests with minimal access time of each page is suspicious.

3.1.6. Eminent source diversity ratio

This feature represents the average number of divergent sources those initiate the sessions in an absolute time interval defined. The request load from eminent sources is tolerable, hence this feature considered as significant.

3.1.7. Absolute bandwidth consumption

This feature represents the average bandwidth consumed by the requests found in absolute time interval defined. This feature also considered as significant since the estimation of bandwidth consumption is critical in load assessment.

3.1.7.1. The estimation of the absolute time interval and other features defined are as follows. The sessions initiated in a given time frame threshold are grouped, and then from each group, the time spent to complete all the sessions in that group will be considered as the time interval of the corresponding group. Then the sum of average of these time intervals and root mean square distance of the respective session groups considered as the absolute time interval.

The number of sessions rendered in each absolute time interval considered as absolute session count of the respective absolute time interval.

The sum of average session completion time of given absolute time interval and their root mean square distance denoted as absolute session completion of the corresponding absolute time interval.

Similarly, the average page access time for given absolute time interval and their root mean square distance aggregated, which denotes the absolute page access time of the corresponding absolute time interval.

Further the total number of pages rendered in a given absolute time interval considered as absolute page access count of the corresponding absolute time interval.

The ratio of eminent sources against the total number of divergent sources found in a given corresponding absolute time interval will be considered as the eminent source diversity ratio of the corresponding absolute time interval.

The total bandwidth consumed by the requests found in a given absolute time interval denoted as absolute bandwidth consumption of the corresponding absolute time interval.

All of these features estimated by ANOVA (Maciá-Fernández et al., 2007) standard, which is the particular the mean of the values adjusted with root mean square distance is used.

3.2. The dataset preparation

This section explores the dataset preprocessing to train the devised model. The labelled transactions given for training phase will be partitioned into Flood and normal transaction sets TF and TN. Then these partitioned sets are used further to extract the features considered for training phase. The absolute time interval will be defined for corresponding datasets TF and TN. Further these will be extracted from TF and TN which will be denoted as *RSF* and *RSN* in further discussion. Each record of the respective sets will represent an absolute time interval and respective values of the other dependent features. The record structure is as follows:

Absolute time interval id	Absolute session interval	absolute session count	absolute page access count	absolute page access time	eminent source diversity ratio	Absolute bandwidth consumption

These attributes will be referred as a set *al* in further draft of the article. The number of attributes in each record will be 7, which is the size of *al* that can be referred as $|al|$.

Further, these record sets *RSF*, *RSN* formed for respective transaction sets TF and TN are used to train the bio-inspired strategy called Cuckoo search. The BARTD process with CUCKOO, Firefly and Bat Algorithms are shown in Figs. 1 and 2.

4. Cuckoo search

Cuckoo Search is a search technique stimulated by the holoparasite act of some Cuckoo birds. The species of type Cuckoo unable to complete its reproduction cycle without proper host (nest of the other type of birds that contains eggs resemble to Cuckoo bird egg). The Cuckoo bird places its egg(s) in the nest of host. The strategy of search followed by a Cuckoo bird is adapted in numerous fields (Stevanovic et al., 2013), Cuckoo Search executes under three traditional rules and they are

(i) Cuckoo selects a host nest randomly to place the egg
(ii) The nest contains most compatible eggs that compared to Cuckoo egg enables the reproduction of the Cuckoo
(iii) The finite number nests (usually the 15) are adapted for Cuckoo search

The probability factor to notify the Cuckoo egg as artifact by a host bird is $\{P(a)\exists a \in (0,1)\}$. In order to optimize the initial nest of the search, usual search follows the techniques such as Levy flights and random walks .In regard to the proposed model, the features are deterministic and ordered by size in descending order, hence the levy flights technique is adapted for perch search.

4.1. Training phase

The Perch or nest formation that referred as the training phase of the Cuckoo search is the critical step. Each perch in our model is

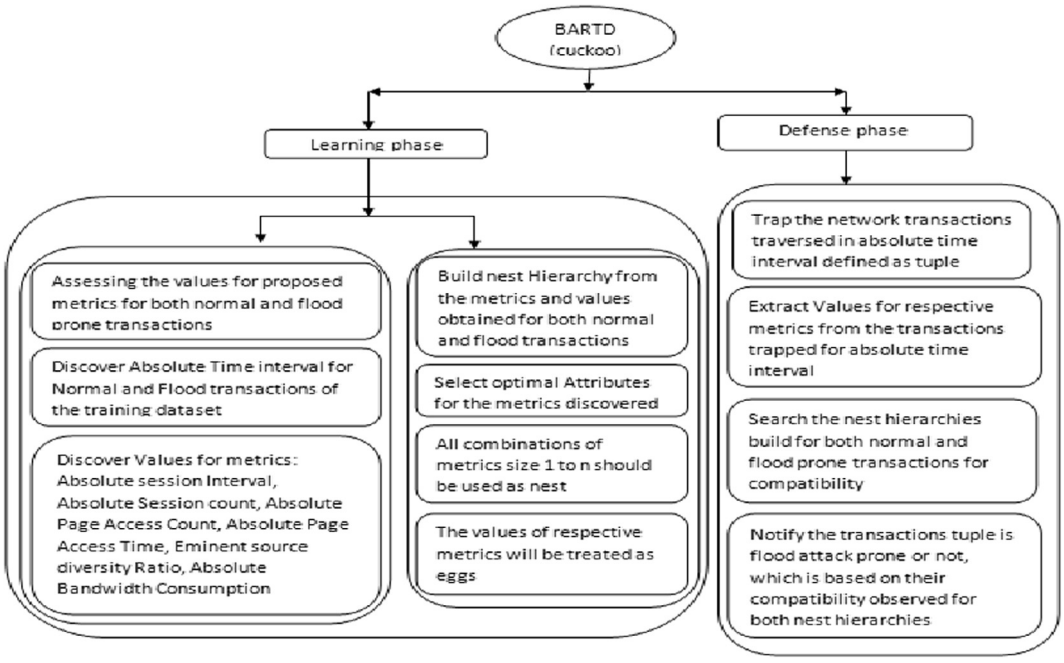

Fig. 1. The process flow of the BIFAD using CUCKOO search.

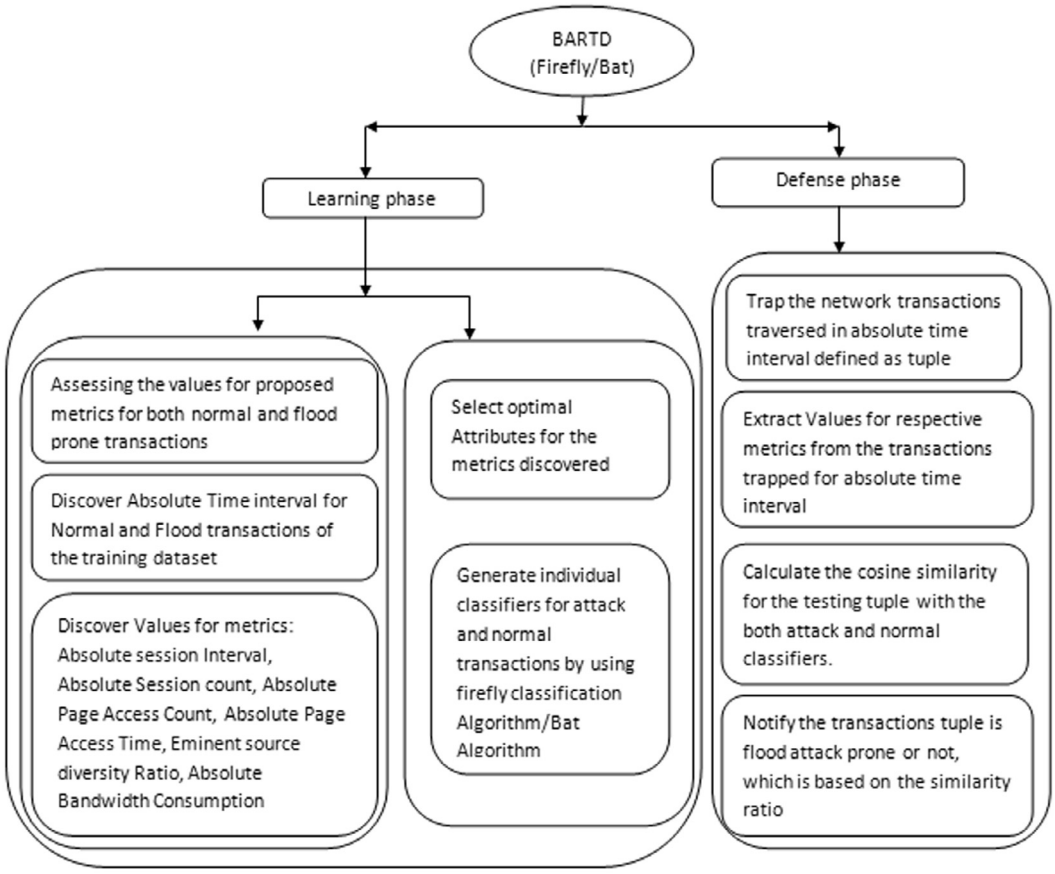

Fig. 2. The process flow of the BIFAD using Firefly and Bat algorithms.

represented by the unique subset (of the size between 1 to $|RSF_{ao}|$ in regard to *RSF* and in between 1 and $|RSN_{ao}|$ in regard to *RSN*) of respective optimal attribute sets.

4.1.1. The process that adopted to form the perches is explored in following steps

i Define the unique subsets of size 1 to $|RSF_{ao}|$ from the optimal attribute set RSF_{ao}, The count of all possible unique subsets of given set RSF_{ao} is $2^{|RSF_{ao}|}-1$. The unique subsets formed from the set RSF_{ao} are further referred as F_{ps} that denotes set of perches in respective to *RSF*. Each entry (which is unique subset of RSF_{ao}) in F_{ps} represents a perch. Similarly define set of perches N_{ps} for RSN_{ao}.

ii Let $RSF_e = \{ao_1e, ao_2e, \ldots ao_{|RSF_e|}e\}$ be the set that contains the elements, such that each element ao_ie is a set, which represents the values found for an optimal attribute ao_i with coverage of one or more records (in *RSF*).

iii Similarly, prepare $RSN_e = \{ao_1e, ao_2e, \ldots ao_{|RSF_e|}e\}$ for records set *RSN*.

iv Order the RSF_{ps} in descending order of the size of perch.

v Form a hierarchy H_F of perch set F_{ps}, such that each level of hierarchy contains the perches of same size.

vi The first level of the hierarchy H_F contains the perch represented by all optimal attributes RSF_{ao}, the second level contains perches represented by subsets of size $|RSF_{ao}|-1$ and so on the last level of the hierarchy contains perches represented by the optimal attribute subsets of size 1.

vii Each perch in the hierarchy is represented by the combination of level id and order id of the perch

viii In similar passion, the hierarchy H_N will be defined (as explained in steps iv to vii) for perch sets N_{ps} obtained for *RSN*.

ix For each perch p_{li} of the hierarchy H_F, the appropriate subsets of RSF_e (the values in sequence observed for optimal attribute representing the perch p_{li}) will considered as the set of eggs $p_{li}(e)$ of perch p_{li}

x In similar passion, add eggs (the values in sequence observed for optimal attribute representing the perch p_{li}) to each perch p_{li} in H_N.

4.2. Perch search: finding the flood status

For a stream of transactions observed in a given absolute time interval *tti* (target time interval), prepare the record $RF(tti)\&RN(tti)$ with the values observed in *tti* for the optimal attributes listed in F_{ao} and N_{ao} respectively. Then form the $2^{|ao|}$ number of unique subsets $RF(tti)_e = \{e_1, e_2, \ldots, e_{|RF(tti)_e|}\}$ and $RN(tti)_e = \{e_1, e_2, \ldots, e_{|Rn(tti)_e|}\}$ of all sizes in the range of 1 to $|F_{ao}|$ and $|N_{ao}|$ respectively. Afterwards, order the $RF(tti)_e\&RN(tti)$ in descending order of the subset size.

4.2.1. Parallel Cuckoo search on H_{RSF} and H_{RSN}

$PS(H_F) \leftarrow \phi//$ is an empty set of perch similarity scores for hierarchy H_F

$PS(H_N) \leftarrow \phi//$ is an empty set of perch similarity scores for hierarchy H_N

For each level $\{l \exists l = 1, 2, 3, \ldots n\}$ Begin

//l represents the level of the hierarchy
For each perch
$\{p_i \exists p_i \in H_F(l) \& p_i \in H_N(l) \wedge i = 1, 2, \ldots |H_F(l)| \exists |H_F(l)| \equiv |H_N(l)|\}$
//each perch p_i that exists in the level l of both hierarchies H_F and H_N
Begin
Let $p_i(e)_F$ be the set of eggs (see step ix in Section 4.1) exists in perch p_i of level l of hierarchy H_F
Let $p_i(e)_N$ be the set of eggs (see step x in Section 4.1) exists in perch p_i of level l of hierarchy H_N

$$PS(H_F) \leftarrow \frac{|RF(tti)_e \cap p_i(e)_F|}{|p_i(e)_F|} \quad (1)$$

// the above equation (Eq. (1)) is assessing the ratio of number of subsets from $R(tti)_e$ exists in perch pi of H_F, which is done by using jaccard index. $|R(tti)_e \cap p_i(e)_F|$ Represents the total number of eggs (subsets) exists in both $R(tti)_e$ and $p_i(e)_F$, $|p_i(e)_F|$ represents the total number of eggs in perch p_i of hierarchy H_F

$$PS(H_N) \leftarrow \frac{|RN(tti)_e \cap p_i(e)_N|}{|p_i(e)_N|} \quad (2)$$

// the above equation (Eq. (7)) is also similar to Eq. (2), which is assessing the ratio of number of subsets from $R(tti)_e$ exists in perch pi of H_N. Here $|R(tti)_e \cap p_i(e)_N|$ represents the total number of eggs (subsets) exists in both $R(tti)_e$ and $p_i(e)_N$, $|p_i(e)_N|$ represents the total number of eggs in perch p_i of hierarchy H_N
The resultant values of the Eqs. (1) and (2) is moved to the sets $PS(H_F)$ and $PS(H_N)$ respectively.
End

4.2.2. Assessing the state of live request stream

The mean of the similarities of the eggs (the values observed for optimal feature representing the respective perch) $RF(tti)_e$ with all available perches in the different levels of hierarchies in H_F will be considered as similarity ratio of respective hierarchy.

$$sr(H_F) = \frac{\sum_{j=1}^{|PS(H_F)|} \{s_j \exists s_j \in PS(H_F)\}}{|PS(H_F)|} \quad (3)$$

$$msd(H_F) = \frac{\sqrt{\sum_{j=1}^{|PS(H_F)|} (sr(H_F) - s_j)^2}}{|PS(H_F)|} \quad (4)$$

The similarity ratio $sr(H_F)$ of the request stream observed in an absolute time interval with H_F is measured (in Eq. (3)) as the mean of $PS(H_F)$, the s_j is the j^{th} similarity score in $PS(H_F)$. The Eq. (4) is assessing the mean square distance of the similarity ratio $sr(H_F)$.

Similarly, the mean of the similarities of the eggs (the values observed for optimal feature representing the respective perch) $RN(tti)_e$ with all available perches in the different levels of hierarchies in H_N will be considered as similarity ratio of respective hierarchy. Further the mean square distance (Choi et al., 2014) of the respective similarity ratio will be measured.

$$sr(H_N) = \frac{\sum_{j=1}^{|PS(H_N)|} \{s_j \exists s_j \in PS(H_N)\}}{|PS(H_N)|} \quad (5)$$

$$msd(H_N) = \frac{\sqrt{\sum_{j=1}^{|PS(H_N)|} (sr(H_N) - s_j)^2}}{|PS(H_N)|} \quad (6)$$

Table 1
Similarity ratios observed from the Cuckoo search on respective hierarchies and the relevant request stream state.

I.	$\begin{pmatrix} (sr(H_F) \cong sr(H_N)) \&\& \\ msd(H_F) < msd(H_N) \end{pmatrix}$	The request stream confirmed as Flood
II.	$(sr(H_F) > sr(H_N))$	The request stream confirmed as Flood
III.	$\begin{pmatrix} sr(H_N) > \\ sr(H_F) + msd(H_F) \end{pmatrix}$	The request stream is confirmed as absolutely normal
IV.	$\begin{pmatrix} (sr(H_N) > sr(H_F)) \&\& \\ (msd(H_N) < msd(H_F)) \end{pmatrix}$	The request stream is confirmed as absolutely normal
V.	None of the above case	Request stream said to be suspicious

The similarity ratio $sr(H_N)$ of the request stream observed in an absolute time interval with H_N is measured (in Eq. (5)) as the mean of $PS(H_N)$, the s_j is the j^{th} similarity score in $PS(H_N)$. The Eq. (11) is assessing the mean square distance of the similarity ratio $sr(H_N)$

Further these similarity ratios and their respective mean square distances are used to assess the state of the request stream observed in an absolute time interval as explored in Table 1.

The similarity ratio $sr(H_F)$ observed from the hierarchy H_F is approximately equal (case I in Table 1) to the similarity ratio $sr(H_N)$ observed from the hierarchy H_N and mean square distance $msd(H_F)$ of H_F is less than the mean square distance $msd(H_N)$ of H_N then request stream confirmed to be flood.

The similarity ratio $sr(H_F)$ observed from the hierarchy H_F is highly greater than (see case II in Table 1) the similarity ratio $sr(H_N)$ observed from the hierarchy H_N then request stream confirmed to be flood.

The similarity ratio $sr(H_N)$ observed from the hierarchy H_N is greater than (see case III in Table 1) the aggregate value of similarity ratio $sr(H_F)$ and mean square distance $msd(H_F)$ observed from the hierarchy H_F then request stream confirmed to be normal.

The similarity ratio $sr(H_N)$ observed from the hierarchy H_N is greater than (see case IV in Table 1) the similarity ratio $sr(H_F)$ observed from H_F and mean square distance $msd(H_N)$ observed from the hierarchy H_N is less than the and mean square distance $msd(H_F)$ observed from the hierarchy H_F, then request stream confirmed to be normal.

If none of the above conditions observed (see case V in table 1), then the request stream said to be suspicious.

5. Firefly approach

Various researches are made on the firefly algorithm to provide novel problem solving approaches. Most of the papers (Jasim Mohammed et al., 2015) proved that Firefly algorithm is used for clustering purpose by finding the global optima. Applications of firefly algorithm has been observed for solving problems with multi-modal functions, continuous and discrete search based problems, multi search problems, parallel computational problems and NP hard problems. In proposed approach firefly algorithm is used to classify the attack traffic and normal traffic. In this algorithm, selection of the best solution and randomization is the two main steps which help us to get accurate classifiers. For both normal and attack sets the classifiers are identified individually.

5.1. Nature of fireflies

In the summer sky, the flashing light of fireflies is an incredible sight within the tropical and temperate regions. Most of the fireflies produce short and swinging flashes. The pattern of flashes is commonly distinctive for a specific species. The flashing light is made by a method of luminescence, and therefore the true functions of such signaling are still debating. Draw in pairing partners

for communication and draw in potential prey are two elementary functions of flashing light .

To develop firefly-inspired algorithms, the flashing characteristics of fireflies are idealized. We tend to currently make the subsequent three perfect rules:

1. One firefly is going to be interested in different fireflies irrespective of their sex as all the fireflies are assumed to be unisex.
2. For any two flashing fireflies, the brighter one will attract the lesser brighter one. The attractiveness is proportional to the brightness and that they each decrease as their distance will increase. If there is no brighter one than a specific firefly, it will move randomly.
3. The brightness of a firefly resolve by the landscape of the target operates. For a maximization problem, the brightness will merely be proportional to the worth of the target operate.

5.2. Classification using firefly algorithm

An initial population of fireflies is generated. After this initialization, modify the parameters needed for fitness, and subsequently the fitness is evaluated for each firefly in the population. Subsequently, the fireflies may be ranked and best individuals of a solution may be taken forward for the next round of evaluation. Number of computations decided in advance can be helpful in controlling the iterations.

Step1: Generate initial population of firefly X_i, where i = 1,2,3,. . .n, n = number of fireflies
Step2: Define Objective function O(x)
Step3: Define Light Absorption coefficient γ = 1, Randomization parameter α = 0.2,Initial Attractiveness β_0 = 1.0
Step4: Define Light intensity I is determined by O(x).
Step5: while t < Number of Iterations
Step6: For i = 1 to NStep7: For j = 1 to N
Step8: If (I_i<I_j)
Step9: If (cosine similarity(i,j) $\geq$ 0.98)
Step 10: For each attribute
Step11: Calculate Cartesian Distance
$r_{ij} = \sqrt{(X_i - X_j)^2}$
Step12: Calculate Attractiveness using equation
$\beta = \beta_0 \exp^{(-\gamma r i j^2)}$
Step13: Move document i to j using equations
$X_i = X_i + \beta * (X_j - X_i) + \alpha\varepsilon_i$ where $\varepsilon_i = (rand - 1/2)$
Step14: End for j
Step 15: End for i
Step 16: End while
Step 17: Rank the fireflies and find the current best

5.2.1. Process for generating classifiers

Step1 Initial population of firefly has to be produced. One record (time capture) is treated as one firefly for which it has any number of features.
Step2 Define the objective function for calculating the fitness of the firefly. Here sum of scaled values of all features of each record is considered as fitness of that record.
Step3 Define the light absorption coefficient γ which characterizes the variation of attractiveness. γ is taken as one. Define the randomization parameter α which is helpful in reducing the number of iterations. Here we are considering $\alpha = 0$ in order getting accurate classifier. Define initial attractiveness β_0 at r = 0. It is assumed that the two fireflies are at same position initially so the distance between them is 0 and the attractiveness is 1. Later the attractiveness value can be calculated by using any of the following two equations.

$$\beta(r) = \beta_0 e^{-\gamma r^2} (or) \beta = \frac{\beta_0}{1 + \gamma r^2}$$

Step4 light intensity can be calculated with the help of objective function. Calculate the fitness value for all the records.
Step5 Number of iterations is user choice. Accuracy of the classifier is purely dependent on the number of iterations. If the number of iterations is high then the accuracy of the classifier will also be high. Here, for a given records we considered 20 iterations.
Step6 to step8: Now, consider the first firefly and compare its light intensity with next firefly's intensity. If it is less than the other, then the first firefly has to be moved towards the second firefly by checking the subsequent condition.
Step9 calculate the cosine similarity (Gomaa, 2013) for both the records and check whether it has more than 98 percent. If it has, then move the first firefly towards the second firefly. How much it has to be moved is measured in following steps.

$$similarity = \cos(\theta) = \frac{A.B}{\|A\|.\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Step10 For each attribute movement value has to be calculated and added to each attribute separately.
Step11 calculated the distance of each feature individually. Distance between the fireflies gives the value of attractiveness.
Step12 calculate the attractiveness based on the distance of individual feature.
Step13 Move the record i to j by using the equation in the algorithm. As the alpha value is 0, we can eliminate the third part of the equation.
Step14 Now the process has to be repeated with remaining fireflies and move the first firefly towards the remaining fireflies.
Step15 Above process has to be done for remaining all fireflies.
Step16 Do the next iteration until it reaches maximum number of iterations.
Step17 Once, maximum number of iterations done then rank the best firefly and consider it as classifier. Ranking is done based on the distance. Minimum distance is given the best rank.

This algorithm is applied on both the attack traffic and normal traffic individually to get both classifiers.

Prepared dataset of both normal and attack is given as input for the firefly algorithm individually. For normal dataset, each record is considered as one firefly and compare with the remaining fireflies to know how much distance it has to move towards the remaining fireflies. This has to be done for all the remaining fireflies. Updated records are carried over to next iteration or generation. Maximum number of iterations has to be performed for getting the accurate classifiers. Once all the iterations are completed, the normal classifier is extracted and marked as normal signature. The same process is carried for attack training records to get the attack signature.

6. Bat approach

Many bio-inspired algorithms exist; bat algorithm belongs to this class which is based on swarm intelligence. The bat algorithm uses the echo based location determining behavior of bats to solve

both single objective and multi-objective optimization problems. In proposed approach bat algorithm is used for classification (classify the attack traffic and normal traffic).

6.1. Nature of bats

Bats can find their prey and discriminate different types of insects even in complete darkness. The BAT algorithm is a population based evolutionary algorithm where each bat represents a solution. It is designed according to the echolocation behavior of the virtual bats. This property enables them to detect the position of their prey. Frequency modulated signals are used for echolocation. Bats listen to the sound pulse emitted by them when it bounces back from the prey or the surrounding objects. As the bat approaches near its prey, it reduces the loudness of their echo and increases the rate of the sound pulse.

The three generalized rules for bat algorithms:

1. All bats use echolocation to sense distance, and they also guess the difference between food/prey and back-ground barriers in some magical way.
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength and loudness A0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r depending on the proximity of their target.
3. Although the loudness can vary in many ways, it is assumed that the loudness varies from a large (positive) A0 to a minimum constant value Amin.

6.2. Classification using bat algorithm

An initial population of bats is generated. After initialization modify the parameters needed for fitness, and subsequently the fitness is evaluated for each bat in population.

Step1: for i = 1: Number Of Iteration
  delwt = wt
Step2: for j = 1: n
  read D_j
Step3:compute each bat_j frequency as f_j
  $f_j = c_1 * \text{Mean}(D_j)$
Step4: compute class(object) distance from bat_j as $S\ object_j$
  Sobject $_j$ = $f_j * D_j *$ delwt
Step5: compute for each class
  E_j = S$object_j - 1$, update $wt = wt - 2 * \mu * E_j$
Step6: compute the new position P_j and change pulse rate controller c_1 of Bat_j
  if $E_j < E_j - 1$
    $P_j = P_j + E_j$
    $c_1 = f_j + c2 * P_j * E_j$
  end
Step7: end
Step8: calculate the error and update wt
  err(i) = Mean(E), wt = wt- $delwt$
Step9:end
Step10: Plot the error as sigmoid(err)
Step11:Use the above wt and f with testing data
Step12: Print confusion matrix from S$object_j$
Step13: Compute percentage of accuracy

6.2.1. Process for generating the classifiers is as follows

Step1 2: Initialize the bat population has to be produced. One record is treated as one bat for which it has any number of features. Define the random number between m, T. Where m is the number of features, and T is the number of classes.

Step3 Calculate the frequency f_i of each bat
$f_j = c_1 * \text{Mean}(D_j)$
where, c_1 is the pulse rate controller initially taken as 0.6, it can change for every iteration.

Step4 calculate the class (object) distance from each bat.

Step5 calculate the E_j of each class and update the random number wt.
Update $wt = wt - 2 * \mu * E_j$
where, μ is the constant number 0.2.

Step6 now, consider the class of one bat and compare with the previous class of the bat, if it is less than the other, then calculate the new position P_j and pulse rate controller C_1 of that bat. Now the process has to be repeated with remaining bats.

Step7 Above process has to be done for remaining all bats.

Step8 calculate the error and update the random number.

Step9 Do the next iteration by using the above process until it reaches the maximum number of iterations.

Prepared dataset of both normal and attack is given as input for the Bat algorithm individually. For normal dataset, each record is considered as one bat and compare with the remaining bats to know how much distance it has to move towards the remaining bats. This has to be done for all the remaining bats. Updated records are carried over to next iteration or generation. Maximum number of iterations has to be performed for getting the accurate classifiers. Once all the iterations are completed, the normal classifier is extracted and marked as normal signature. The same process is carried for attack training records to get the attack signature.

7. Application layer DDoS attack detection using firefly and Bat algorithms

The detection of the testing absolute time intervals(ati) are given as the input for Firefly and Bat algorithms individually to calculate the individual weights for each ati. Testing dataset has to be preprocessed by using the dataset preprocessing process. Prepare the dataset with seven attributes as like in the dataset preparation (see Section 3.2). Calculate the total weight of the testing records individually. Calculate the cosine similarity of testing record with both normal and attack signatures and define whether the testing record is attack or normal by using the rules defined in Table 2.

8. Empirical study and performance analysis

In order to assess the performance of the BARTD, the experiments were carried out on transaction dataset generated by the JMETER (Kiran et al., 2015). The DDoS attack prone and normal transactions were collected for a period 60 min each. The statistics of the transactions collected from JMETER, which are used further in experimental study are explored in Table 3. The statistics obtained from the dataset preprocessing are explored in Tables 7 and 8 (see Appendix-A). The averages of the values obtained for devised metrics are explored in Table 4, which are approximately

Table 2
Rules define for attack detection.

Rule1	Weight of the testing time interval is less than the normal classifier weight and greater than the attack weight	$A(w) < T(w) \leqslant N(w)$	Normal
Rule2	2.1 similarity of testing record with the normal classifier is more than 98 percent	$similarity(test, normal) \geqslant 98\%$	Normal
	2.2 similarity of testing record with the attack classifier is more than 98 percent	$similarity(test, attack) \geqslant 98\%$	Attack
Rule3	Similarity of testing record with normal classifier is more than the similarity of testing record with attack classifier	$similarity(test, normal) > similarity(test, attack)$	Normal
Rule4	All the above conditions are failed	Suspicious	

Table 3
The statistics of the dataset generated by JMeter.

	Particulars of Intensified Load given as Flood	Normal Load
JMeter running time	60 min	60 min
No of sessions	4519	3597
No of requests	86749	51849
Bandwidth Consumed	4501.34 MB	2721.88 MB
No of sources	377	219

Table 4
Average values obtained for the features defined.

	From DDoS Attack Transactions	From Normal Transaction
Absolute Time Interval Observed	199.8 s	199.8 s
Average of Absolute Session Count observed for all absolute time intervals	63	37
Average absolute session interval	2.29 min	2.94 min
Average of absolute page access count	1205	720
Average of absolute page access interval	0.12	0.21
Average of eminent source diversity	3.4	0.34
Average absolute bandwidth consumption ratio	62.52	37.8

signifying the impact of the devised metrics towards flood detection.

The records obtained after dataset processing are considered as *RTF* and *RTN* sets that are representing the HTTP flood and normal transactions respectively. The total number of records defined is 72 each in flood and normal transactions. Among these 70% of the records from both *RTF&RTN* are used to train the given model and rest 30% of the records from both *RTF&RTN* are used to test the model.

A computer with i5 processor, 4 GB ram and Nvidia 4 GB graphics card (NVIDI, 2015) used. The implementation was done in CUDA (Nvidi, 2008), since the assessment metrics, computational and resource complexity also included in performance analysis, Statistical metrics analysis was done using explorative language R (Ross and Gentleman, 1996).

The scalability and significance of the proposed model is assessed through cross validation metrics like precision, accuracy, sensitivity and specificity) (Powers, 2011). The results obtained for these metrics listed in table 5.The values observed for detection accuracy is 0.955 with minimal false alarming rate that observed as 0.045. Hence, the BARTD is considerably significant to detect HTTP Flood Attacks.

Table 5
Prediction ratios and assessment metric values observed.

Total Number of Absolute time intervals	144 (72:RTF, 72:RTN)
The number of Absolute Time Intervals Used for Training	100 (50:RTF, 50:RTN)
The number of Absolute Time Intervals Used for Testing	44 (22:RTF, 22:RTN)
Total Number of records found as attacks	23
Total number of records found as normal	21
True Positives (truly predicted as attacks)	22
False Positives (falsely predicted as attacks	1
True Negatives (truly predicted as normal)	20
False Negative (Falsely predicted as normal)	1
Precision	0.957
Sensitivity	0.957
Specificity	0.909
Accuracy	0.955

Table 6
Prediction ratios and assessment metric values observed for Firefly and Bat Algorithms.

	Firefly Approach	Bat Algorithm
Total Number of Absolute time intervals	144 (72:RTF, 72:RTN)	
The number of Absolute Time Intervals Used for Training	100 (50:RTF, 50:RTN)	
The number of Absolute Time Intervals Used for Testing	44 (22:RTF, 22:RTN)	
Total Number of records found as attacks	24	26
Total number of records found as normal	23	21
True Positives (truly predicted as attacks)	23	24
False Positives (falsely predicted as attacks	1	2
True Negatives (truly predicted as normal)	22	19
False Negative (Falsely predicted as normal)	1	2
Precision	0.92	0.923077
Sensitivity	0.92	0.904762
Specificity	0.948333	0.923077
Accuracy	0.938776	0.914894
F-Measure	0.924539	0.914894

The robustness of the BARTD in regard to validation metrics are assessed through experimental study, The observations of these were compared to the other contemporary methods such as Firefly and Bat algorithms and these are shown in Table 6. However, the detection process of this model is conceptually different from the proposal but robust to detect HTTP flood through most traditional approaches called "Firefly and Bat algorithms".

The process complexity also assessed for Firefly approach, Bat algorithm and BARTD and compared (see Fig. 3). The process completion time observed for Firefly approach and Bat algorithm are complimenting against the increase in load, whereas in the case of BARTD, the process completion time is linear and these are shown in Figs. 4 and 5.

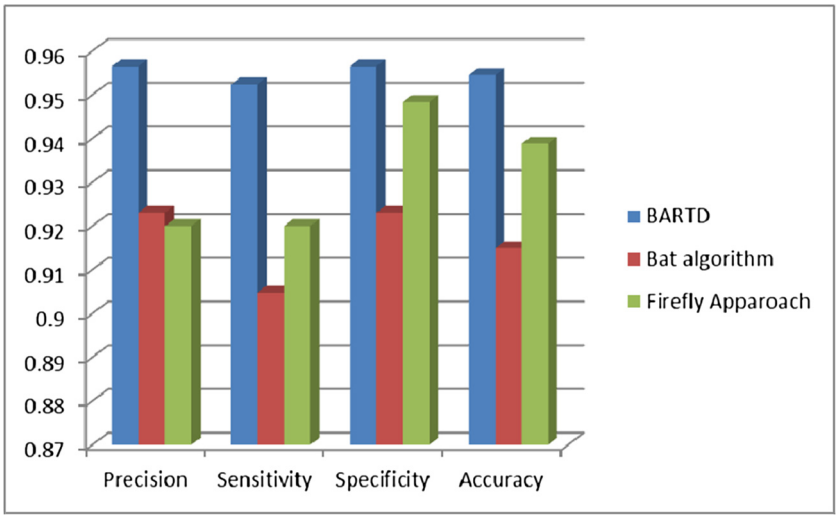

Fig. 3. Comparison of metrics using BARTD, Bat and Firefly algorithms.

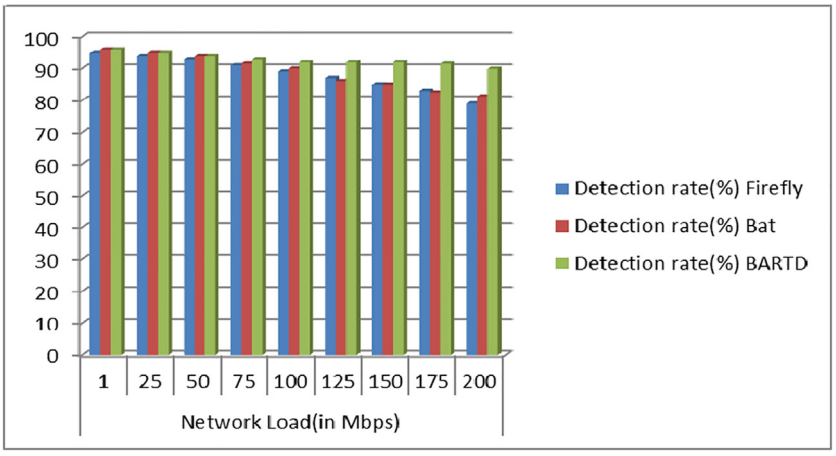

Fig. 4. The comparison of detection rate observed for BARTD and "Bat and Firefly algorithms" against load.

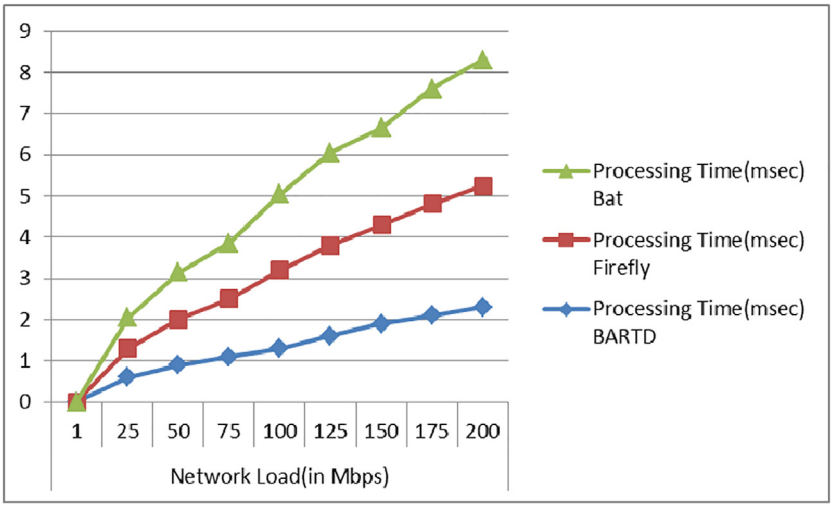

Fig. 5. The comparison of processing time observed for BARTD and "Firefly and Bat algorithms" against load.

9. Conclusion

Bio-Inspired Anomaly based Real Time Detection of under rated App-DDoS Attack on Web is devised in this article. In this, we adopted the Cuckoo search which is bio-inspired approach with magnified speed in search. The overall contribution of the paper is in three levels. The initial contribution is define feature metrics to identify the request stream behavior is of attack intension or not. Unlike traditional approaches, the assessment of feature metrics done on the stream of requests observed in an absolute time interval rather in a session. The Second contribution is to customize the Cuckoo search to train and search. The default approach of the cuckoo Search is random search, unlike that the proposal customized the cuckoo search to perform search in a hierarchical order rather than the random approach. Finally verified these process using Firefly and Bat algorithms for the same dataset. The devised Cuckoo Search amplified the detection accuracy with minimal process complexity. The experiments conducted using JMeater. Further, the training records used to define the Cuckoo Search strategy, which observes to be robust and is with minimal process complexity that compared other benchmarking models called Firefly and Bat algorithms. Hence the model devised here in this paper is significantly minimized the computational overhead and retains the maximal prediction accuracy. In future hybrid strategy that uses more than one bio-inspired technique to define a novel anomaly based divergent flood attack detection. In addition, the proposed model can be extended to identify the other significant attacks like Flash crowd.

Appendix-A

Table 7
The values obtained from attack prone date for the selected features.

Absolute Time Interval ID	Absolute session count	Absolute Session Interval	Absolute Page Access Count	Absolute Page Access Interval	Eminent Source Diversity	Absolute Bandwidth Consumption
1	58	1.9	986	0.11	5	67.28
2	69	2.67	1311	0.14	3	129.81
3	69	2.11	1242	0.12	2	27.61
4	59	2.15	1121	0.11	3	96.92
5	61	3.12	1159	0.16	3	110.22
6	63	1.74	1386	0.08	2	2.47
7	69	2.67	1173	0.16	3	87.15
8	69	2.11	1380	0.11	4	47.71
9	68	3.24	1292	0.17	5	85.3
10	63	3.2	1134	0.18	3	71.16
11	57	2.39	1026	0.13	4	79.19
12	61	1.58	1342	0.07	2	84.2
13	66	1.86	1188	0.1	4	104.03
14	61	3.16	1281	0.15	3	86.79
15	54	3.16	1188	0.14	5	57.64
16	59	2.59	1003	0.15	4	14.82
17	67	2.39	1206	0.13	2	95
18	69	2.15	1311	0.11	5	17.53
19	69	2.11	1173	0.12	2	90.55
20	61	1.86	1281	0.09	2	61.94
21	70	2.31	1190	0.14	2	98.45
22	67	1.78	1474	0.08	4	78.19
23	59	2.88	1121	0.15	4	73.13
24	59	3	1239	0.14	4	1.4
25	68	2.72	1428	0.13	4	35.91
26	67	1.54	1206	0.09	3	116.3
27	63	2.23	1197	0.12	5	73.2
28	70	2.15	1470	0.1	5	115.91
29	67	2.84	1407	0.14	2	136.79
30	54	1.58	1026	0.08	5	50.95
31	54	2.88	1134	0.14	2	93.01
32	65	2.8	1365	0.13	4	60.35
33	67	3.28	1273	0.17	2	65.04
34	60	2.63	1140	0.14	5	40.45
35	57	3.32	1140	0.17	4	99.61
36	70	1.66	1470	0.08	4	83.28
37	66	2.55	1452	0.12	5	85.39
38	65	3.24	1105	0.19	2	34.71
39	63	1.7	1197	0.09	4	43.57
40	54	1.74	1134	0.08	5	17.83
41	59	2.59	1298	0.12	2	13.56
42	58	2.39	1160	0.12	2	101.93
43	56	1.9	1008	0.11	2	70.97
44	66	2.43	1188	0.14	4	4.06
45	58	2.96	986	0.17	3	47.23
46	62	1.54	1364	0.07	2	23.6
47	61	2.51	1281	0.12	4	72.47
48	67	2.35	1407	0.11	4	139.43
49	67	1.74	1407	0.08	2	89.76
50	60	1.7	1140	0.09	4	71.97
51	54	2.31	972	0.13	4	86.83
52	66	1.42	1188	0.08	4	18.17
53	65	2.47	1235	0.13	4	54.42

Table 7 (*continued*)

Absolute Time Interval ID	Absolute session count	Absolute Session Interval	Absolute Page Access Count	Absolute Page Access Interval	Eminent Source Diversity	Absolute Bandwidth Consumption
54	61	1.62	1037	0.1	4	57.96
55	66	1.7	1122	0.1	3	13.11
56	54	3.08	1026	0.16	2	44.48
57	70	1.82	1190	0.11	5	26.01
58	69	1.9	1449	0.09	2	68.73
59	63	2.39	1071	0.14	3	2.67
60	66	1.78	1254	0.09	4	35.9
61	62	2.31	1054	0.14	4	80.73
62	62	3.08	1054	0.18	3	23.32
63	61	1.66	1159	0.09	2	18.9
64	68	2.47	1292	0.13	3	93.8
65	66	1.78	1122	0.1	4	96
66	56	2.11	1232	0.1	2	19.08
67	54	1.62	972	0.09	5	1.54
68	63	2.47	1197	0.13	4	39.57
69	57	1.34	1026	0.07	3	42.93
70	68	1.66	1224	0.09	5	72.06
71	58	1.82	1044	0.1	5	74.32
72	59	2.84	1239	0.14	2	75.02

Table 8
The values obtained from normal data for the selected features.

Absolute Time Interval ID	Absolute session count	Absolute Session Interval	Absolute Page Access Count	Absolute Page Access Interval	Eminent Source Diversity	Absolute Bandwidth Consumption
1	41	3.08	574	0.22	3	24.39
2	47	2.8	846	0.16	4	6.91
3	42	3.24	714	0.19	3	11.33
4	45	2.84	495	0.26	3	19.83
5	59	3.16	649	0.29	4	60.79
6	57	2.67	1026	0.15	4	55.91
7	42	3.32	462	0.3	2	19.17
8	43	3.08	602	0.22	5	33.4
9	47	2.59	799	0.15	4	39.19
10	60	2.84	1020	0.17	2	35.71
11	51	2.59	663	0.2	4	24.57
12	58	3.04	928	0.19	2	58.97
13	52	3.16	780	0.21	3	67.77
14	42	3.12	588	0.22	5	4.42
15	58	3.08	754	0.24	3	29.84
16	46	3.28	552	0.27	2	14.88
17	53	2.76	795	0.18	3	76.31
18	46	2.59	690	0.17	4	20.89
19	55	2.84	605	0.26	4	38.36
20	56	3.12	1008	0.17	2	11.99
21	50	2.67	750	0.18	4	64.94
22	59	2.88	1003	0.17	5	78.89
23	47	2.92	517	0.27	5	49.92
24	58	3.16	870	0.21	4	74.34
25	54	2.8	594	0.25	3	5.78
26	55	2.72	935	0.16	4	13.01
27	47	2.76	846	0.15	5	67.94
28	41	2.96	574	0.21	4	32.94
29	44	2.92	484	0.27	4	3.43
30	52	3.16	572	0.29	3	26.28
31	53	2.76	742	0.2	4	72.88
32	43	2.84	731	0.17	4	4.27
33	60	3.32	1020	0.2	2	51.83
34	41	2.67	697	0.16	4	10.54
35	50	3.2	850	0.19	2	42.68
36	55	3.28	990	0.18	2	15.21
37	54	2.63	702	0.2	2	30.05
38	43	3	559	0.23	5	12.94
39	48	3	624	0.23	3	28.18
40	43	2.51	645	0.17	3	15.44
41	41	3.04	492	0.25	3	5.48
42	57	2.76	684	0.23	2	38.38
43	46	3.04	736	0.19	3	49.08
44	60	2.72	720	0.23	4	17.07
45	58	2.84	812	0.2	5	28.17
46	46	2.67	598	0.21	5	48.52

(*continued on next page*)

Table 8 (*continued*)

Absolute Time Interval ID	Absolute session count	Absolute Session Interval	Absolute Page Access Count	Absolute Page Access Interval	Eminent Source Diversity	Absolute Bandwidth Consumption
47	48	2.76	576	0.23	4	3.48
48	41	3.2	697	0.19	3	57.49
49	47	2.76	705	0.18	2	35.53
50	58	2.88	696	0.24	2	62.48
51	41	3.12	451	0.28	2	42.26
52	58	2.92	754	0.22	4	38.08
53	46	3	506	0.27	5	30.26
54	52	2.84	884	0.17	3	3.39
55	42	2.92	546	0.22	2	45.23
56	51	2.76	918	0.15	2	88.2
57	45	2.92	540	0.24	2	49.7
58	49	3.08	784	0.19	4	50.5
59	52	2.92	936	0.16	4	90.61
60	58	3.24	696	0.27	5	51.33
61	49	3.12	637	0.24	3	9.62
62	59	3	826	0.21	3	42.68
63	43	3.24	559	0.25	5	30.47
64	53	2.88	795	0.19	2	56.12
65	50	3.2	800	0.2	3	27.79
66	57	3.24	855	0.22	2	53.32
67	52	2.67	728	0.19	4	19.91
68	46	2.84	598	0.22	4	53.16
69	45	3.08	765	0.18	4	62.13
70	50	2.84	700	0.2	3	61.02
71	50	2.84	800	0.18	3	57.4
72	47	3.24	800	0.19	2	56.90

References

Abdul, A. Razzaq et al., 2011. Foundation of semantic rule engine to protect web application attacks. In: 2011 Tenth International Symposium on Autonomous Decentralized Systems. IEEE.

Bhatia, S., Schmidt, D., Mohay, G., 2012. Ensemble-based DDoS detection and mitigation model. In: Proceedings of the Fifth International Conference on Security of Information and Networks. ACM, pp. 79–86.

Choi, Junho et al., 2013. Detecting web based DDoS attack using Map Reduce operations in cloud computing environment. J. Internet Services Inf. Secur. 3.3/4, 28–37.

Choi, Junho et al., 2014. A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment. Soft Comput. 18.9, 1697–1703.

Xia, Chun-Tao, Xue-Hui, Du, Li-Feng, Cao, 2012. An algorithm of detecting and defending CC attack in real time. International Conference on Industrial Control and Electronics Engineering, 1804–1806.

Gomaa, W.H., 2013. A survey of text similarity approaches. Int. J. Comput. Appl. 68 (13), 13–18.

Hameed, Sufian, Ali, Usman, 2015. On the Efficacy of Live DDoS Detection with Hadoop. arXiv preprint arXiv:1506.08953.

Mohammed, Athraa Jasim, Yusof, Yuhanis, et al., 2015. Determining Number of Clusters using Firefly Algorithm with Cluster Merging for Text Clustering. Springer International Publishing Switzerland.

Jung, J., Krishnamurthy, B., Rabinovich, M., 2002. Flash crowds and denial of service attacks: characterization and implications for cdns and web sites. In: Proceedings of the 11th International Conference on World Wide Web, WWW '02, New York, NY, USA, ACM, pp. 293–304.

Kambouakis, Georgios, Moschos, Tassos, Geneiatakis, Dimitris, Gritzalis, Stefanos, 2007. A fair solution to DNS amplification attacks.

Kiran, Sandhya, Mohapatra, Akshyansu, Swamy, Rajashekara, 2015. Experiences in performance testing of web applications with Unified Authentication platform using Jmeter. 2015 International Symposium on Technology Management and Emerging Technologies (ISTMET), IEEE.

Kumar, Raj, Nene, Manisha Jitendra, 2013. A survey on latest DoS attacks: classification and defense mechanisms. Proc. Int. J. Innovative Res. Comput. Commun. Eng. 1 (8).

Lee, S.M., 2004. Distributed denial of service: taxonomies of attacks, tools, and countermeasures. In: Proceedings of the International Workshop on Security in Parallel and Distributed Systems, San Francisco, pp: 543–550.

Liu, Huey-Ing, Chang, Kuo-Chao, 2011. Defending systems against tilt DDoS attacks. 2011 6th International Conference on . Telecommunication Systems, Services, and Applications (TSSA), IEEE.

Maciá-Fernández, G., Díaz-Verdejo, J.E., García-Teodoro, P., 2007. Evaluation of a low-rate dos attack against iterative servers. Comput. Netw. 51 (4), 1013–1030.

Macia-Fernandez, G., Diaz-Verdejo, J., Garcia-Teodoro, P., 2009. Mathematical model for low-rate dos attacks against application servers. IEEE Trans. Inf. Forensics Secur. 4 (3), 519–529.

Maciá-Fernández, G., Rodríguez-Gómez, R.A., Díaz-Verdejo, J.E., 2010. October). Defense techniques for low-rate dos attacks against application servers. Comput. Netw. 54 (15), 2711–2727.

Mehra, M., Agarwal, M., Pawar, R., Shah, D., 2011. Mitigating denial of service attack using captcha mechanism. In: Proceedings of the International Conference & #38; Workshop on Emerging Trends in Technology, ICWET '11, New York, NY, USA, ACM, pp. 284–287.

Mori, G., Malik, J., 2003. Recognizing objects in adversarial clutter: breaking a visual captcha. In CVPR 1, 134–141.

Nam, S.Y., Lee, T., 2009. Memory-efficient IP filtering for countering DDoS attacks. In: Proceedings of the 12th Asia-Pacific network operations and management conference on Management enabling the future internet for changing business and new computing services, APNOMS'09, Berlin, Heidelberg. Springer-Verlag, pp. 301–310.

Nvidia, C.U.D.A., 2008. Programming guide.

NVIDIA, 2015. PNY-NVIDIA-GeForce-GTX-960-4GB-XLR8.pdf.

Powers, David Martin, 2011. Evaluation: from precision, recall and F-measure to ROC, informed ness, marked ness and correlation.

Prasad, K. Munivara, Mohan Reddy, A. Rama, Venugopal Rao, K., 2016. Anomaly based Real Time Prevention of under Rated App-DDoS Attacks on Web: An Experiential Metrics based Machine Learning Metrics. Indian J. Sci. Technol. 9 (27). http://dx.doi.org/10.17485/ijst/2016/v9i27/87872.

Prasad, K. Munivara, Mohan Redy, A. Rama, Venugopal Rao, K., 2017. ,BIFAD: Bio-Inspired Anomaly Based HTTP-Flood Attack Detection. Wireless Pers Commun, 01 June 2017 (Online first). pp. 1–28.

Ranjan, S. et al., 2006. DDoS-resilient scheduling to counter application layer attacks under imperfect detection. Proceedings of IEEE INFOCOM, 23–29.

Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A., Knightly, E., 2009. DDoS-shield: DDoS-resilient scheduling to counter application layer attacks. IEEE/ACM Trans. Netw. 17 (1), 26–39.

Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A., Knightly, 2009. DDoS-shield: DDoS-resilient scheduling to counter application layer attacks. IEEE/ACM Trans. Netw. (TON) 17 (1), 26–39.

Ross, Ihaka, Gentleman, Robert, 1996. R: a language for data analysis and graphics. J. Comput. Graphical Stat. 5 (3), 299–314.

Salmen, Fadir, Galego Hernandes Paulo R., Jr., et al., 2015. Using Firefly and Genetic Meta heuristics for Anomaly Detection based on Network Flows. AICT: The Eleventh Advanced International Conference on Telecommunications.

Senthilnath, J., Omkar, S.N., Mani, V., 2011. Clustering using firefly algorithm: performance study. ELSEVIER, Swarm and Evolutionary Computation 1, 164–171.

Stevanovic, Dusan, Vlajic, Natalija, An, Aijun, 2013. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. Appl. Soft Comput. 13 (1), 698–708.

Tang, Y., 2012. Countermeasures on application level low-rate denial of-service attack. In: Proceedings of the 14th international conference on Information and Communications Security, ICICS'12, Berlin, Heidelberg. Springer-Verlag, pp. 70–80.

Udhayan, J., Anitha, R., 2009. Demystifying and rate limiting ICMP hosted DoS/DDOS flooding attacks with attack productivity analysis. IEEE International Conference on Advance Computing, 558–564.

Vijayalakshmi, M., Mercy Shalinie, S., 2012. IP traceback system for network and application layer attacks. Recent Trends In Information Technology (ICRTIT), 2012 International Conference on. IEEE.

Wang, Yadong, Liu, Lianzhong, et al., 2015. A survey of defense mechanisms against Application layer distributed denial of service (DDoS) attacks. IEEE Commun. Surveys Tutorials.

Wen, S., Jia, W., Zhou, W., Zhou, W., Xu, C., 2010. Cald: Surviving various application-layer DDoS attacks that mimic flash crowd. In: Network and System Security (NSS), 2010 4th International Conference on. IEEE.

Xiang, Yang, Li, Ke, Zhou, Wanlei, 2011. Low-rate DDoS attacks detection and traceback by using new information metrics. IEEE Trans. Inf. Forensics Secur. 6 (2), 426–437.

Xie, Y., Zheng Yu, S., 2006. A novel model for detecting application layer DDoS attacks. In Computer and Computational Sciences, 2006. IMSCCS '06. First International Multi-Symposiums on, Vol. 2, pp. 56–63.

Xie, Y., Zheng Yu, S., 2009. Monitoring the application-layer DDoS attacks for popular websites. IEEE/ACM Trans. Netw. 17 (1), 15–25.

Xuan, Y., Shin, I., Thai, M., Znati, T., 2010. Detecting application denial of-service attacks: a group-testing-based approach. IEEE Trans. Parallel Distributed Syst. 21 (8), 1203–1216.

Xuan, Ying, Shin, Incheol, Thai, My T., Znati, Taieb, 2010. Detecting application denial-of-service attacks: a group-testing-based approach. IEEE Trans. Parallel Distrib. Syst. 21 (8), 1203–1216.

Ye, C. Zheng K., 2011. Detection of application layer distributed denial of service. In Computer Science and Network Technology (ICCSNT), 2011 International Conference on, Vol. 1, pp. 310–314.

Yu, Jie, Li, Zhoujun, Chen, Huowang, Chen, Xiaoming, 2007. A detection and offense mechanism to defend against application layer DDoS attacks. ICNS. Third International Conference on Networking and Services, 2007, IEEE.

Yu, J., Li, Z., Chen, H., Chen, X., 2007. A detection and offense mechanism to defend against application layer DDoS attacks. In Networking and Services, 2007. ICNS. Third International Conference on, pp. 54–54.

Yu, J., Fang, C., Lu, L., Li, Z., 2010. Mitigating application layer distributed denial of service attacks via effective trust management. IET Commun. 4 (16), 1952–1962.

Zolotukhin, Mikhail, Hamalainen Timo, et al., 2016. Increasing Web Service Availability by Detecting Application-Layer DDoS Attacks in Encrypted Traffic. IEEE, 23rd International Conference on Telecommunications (ICT).