IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

HAP: detection of HTTP flooding attacks in cloud using diffusion map and affinity propagation clustering

ISSN 1751-8709
Received on 23rd July 2018
Revised 13th October 2018
Accepted on 6th November 2018
E-First on 29th January 2019
doi: 10.1049/iet-ifs.2018.5382
www.ietdl.org

Thankaraja Raja Sree[1] ✉, *Somasundaram Mary Saira Bhanu*[1]

[1]*Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, India*

✉ *E-mail: trajasree87@gmail.com*

Abstract: The extreme growth of Internet resources leads to several kinds of attacks. Cybercrime is one of the dominant threats apart from data defence mechanism, which enhances the economy, resource management, and service quality. Among them, HTTP flooding attacks in the cloud are one of the most prevalent threats as it depletes the cloud resources and services. It is difficult to distinguish the anomalous traffic by extracting the actual payload since most of the payload could not be accessed as they are encrypted and varies dynamically based on the user input. Hence, the proposed method uses web server logs that can be easily accessed to detect the attacks. This study highlights the detection methods by extracting the features from the web server logs and also deals with the reduction in the dimensionality of the features using diffusion map. The anomalies are detected by affinity propagation clustering technique and also by monitoring the status of the virtual machine. Furthermore, the Dempster–Shafer theory focuses on the identification of the suspicious user. It is inferred from the experimental results that the proposed method enhances the detection performance with very few false alarms than existing methods.

Nomenclature

$\boldsymbol{R}_{\mathbf{eq}}$	HTTP request of an instance
nReq	n-gram features of HTTP requests of an instance
N	normalised features obtained from GRPS, PS and RS of an instance
W_{AHP}	weight calculation using analytical hierarchical process of normalised features of an instance N
W_{DM}	diffusion map of n-gram features of HTTP request of an instance
W_{OW}	overall weight obtained from W_{AHP} and W_{DM}
S	pairwise similarity of an instance i from W_{OW}
C	affinity propagation clustering of similarity matrix $\boldsymbol{S}$
outlier_list	list of members in outlier
$T_c(i)$	threshold of cluster i
$D_o(c)$	Euclidean distance from outlier to cluster
m_c	minimum distance cluster
E_dist	Euclidean distance from member to cluster
IP_i	Internet protocol address of an instance i
W	non-zero vector (eigenvector)
CI	consistency index
$\boldsymbol{B}$	$n * n$ comparison matrix
W_{ni}	final weights of each IP
Bel(R)	belief of an instance R
Pl(R)	plausibility of an instance R
α, β	adjustment parameter to calculate weights
γ	damping factor
GRPS	Get Request Per Second of an instance
PS	Packet Size of an instance
RT	Response time of an instance
Req	HTTP request of an instance

1 Introduction

Nowadays, all over the world people are mostly subject to the internet resources. Its usage has been increasing tremendously over the years [1]. In order to provide services efficiently at low cost, cloud computing has emerged as one of the prominent technologies that offer on-demand services to the users. The availability of massive computation at very low cost motivates a malicious individual or an attacker to launch attacks either from inside or outside the cloud. This causes high resource consumption and also results in the limited access to cloud services. Denial of Service (DoS) is one of the predominant attacks since it ends up in the bankruptcy of the cloud resources and services, thereby causing DoS to the benign users. Distributed DoS (DDoS) is the distributed type of DoS attack that makes the online service unavailable by overwhelming all the resources with the heavy flow of traffic from various sources [2].

The DDoS attack is launched in various layers of the network. In the past, network and transport layers were the major targets for the attackers to deplete the resources by consuming network bandwidth. These attacks were detected by using network patterns, traffic analysis, Internet Protocol (IP) blocking, and network infringement method by collecting the information from various components viz., router, switches etc. [3]. Nowadays, the attackers shift their strategies towards the application layer to establish a large number of sophisticated attacks to exploit the resources and to damage the servers. The web application and web servers are the major targeted resources in an application layer since an attacker exploits the different number of vulnerabilities and intrusions to destroy the resources and services in cloud [4, 5].

HTTP flooding attack is a sort of DDoS attack that targets the cloud services by sending enormous malicious packets to the victim Virtual Machine (VM), thereby saturating all the resources running in cloud viz., bandwidth, memory, CPU cycles, I/O devices etc. The presence of an Intrusion Detection System (IDS) in an organisation plays a major role to monitor the suspicious activity and to alert the system. IDS is classified into two groups on the basis of detection principle viz., signature-based and anomaly-based. In signature-based attack detection, the rules are generated manually on the basis of known attack patterns. If there is a match, then an alarm is raised. It is used due to its computational simplicity. It can detect only previously known attacks. However, the unknown attacks remain undetected.

In the case of the anomaly detection method, any new type of misuse that differs from the normal behaviour in the network has been detected. If any deviation is found, an alarm is raised. The new and unknown attacks can be detected by using this method. The various approaches used in the existing literature for the detection of attacks are statistical methods [6–8], neural networks [9, 10], machine learning [10], Support Vector Machine (SVM)

[11] etc. Even though these methods detect the attacks effectively, they result in a large number of false positives.

Most of the anomaly detection methods assume the detection model as static in nature which does not possess the ability to adapt to the changes in normal behaviour which continuously evolve over time. It is crucial to analyse the large set of network patterns of an actual payload of HTTP traffic as most of the information is encrypted. So, web server logs are used to identify the traffic. The features that are extracted from the web server logs are huge and dynamic in nature. Hence, the dimensionality reduction technique is used to reduce the size of the features and also to represent as meaningful structures.

Dimensionality reduction technique converts the data of very high dimensionality into data of much lower dimensionality without altering the information content. Principal Component Analysis (PCA) [12], Diffusion Map (DM) [13], Independent Component Analysis [14], Random Projection (RP) [15] etc. are the frequently used techniques to detect DDoS attacks. Modified PCA is used to analyse the new incoming traffic by online updating technique [16] and this method supports large-scale problems. This method will not work well for non-linear data since the computational overhead is high. RP is used in web-based attack detection, which functions efficiently in the daily analysis of the huge volume of traffic. DM methodology is used to determine both the online [15, 17] and offline attacks [13, 18, 19]. The rule extraction algorithm is used to identify the web-related attacks online. The advantage of DM is that it can handle non-linear data, which preserves the local structures. In addition, DM provides the better visualisation of attacks with very few false alarms.

Clustering plays a vital role in the grouping of similar input patterns to determine anomalies and to filter out noisy data or outliers [20–22]. Several clustering techniques such as *k*-means [18], hierarchical [23], density based [21, 24] etc. are used to detect outliers by the grouping of input patterns based on similarities. Most of the clustering algorithms are static in nature, which is not applicable for identifying the dynamic variations of the traffic patterns. Moreover, most of the clustering-based detection approaches are highly sensitive to cluster centre initialisation. The number of clusters is to be defined in advance to determine the unknown attacks and it should not be stuck in local optima. Therefore, to detect the traffic online and also to improve the detection rate, Affinity Propagation (AP) clustering is used [15]. The AP clustering technique monitors the large stream of incoming traffic to detect the anomalies that emerge over time. The advantage of AP clustering is that it neither requires to specify the number of clusters beforehand nor prior knowledge to group the similar input patterns and also detects unknown attacks.

In order to mitigate these limitations, the proposed method (HAP) deals with the web server logs by extracting the significant features viz., Get Request Per Second (GRPS), Response Time (RT), and HTTP request [Uniform Resource Locator (URL)] and these input features are fed to the preprocessor to obtain the normalised features. DM is used to reduce the dimensionality of the normalised features without altering the information content. The attacks are detected by grouping the similar normalised features using AP clustering and also by monitoring the status of VM CPU usage and throughput. There are special cases where AP clustering cannot identify the traffic as either benign or malicious due to the insufficient information from the traffic data. Hence, Dempster–Shafer theory (DST) is used to identify this suspicious traffic since this uses the mass value of the clustered data [25]. The proposed method (HAP) is tested using real HTTP traffic of web server logs. The main inputs of the work are as follows:

- The n-gram preprocessed features are passed to the DM to reduce the size of the features without changing the information content.
- HAP is a dynamic clustering technique that solves the problem of determining unknown attacks without having the prior knowledge to group the relevant patterns using AP clustering.
- The suspicious traffic is identified from the mass value of the clustered information using DST of evidence.
- The performance of HAP is evaluated with the existing methods such as non-parametric Cumulative Sum (CUSUM) method [24], HADM [26], DM + K-Nearest Neighbour (KNN) [27], DM + statistical [15], DM + *k*-means [18, 20], in which HAP achieves high detection rate and reduces false positives.

The paper proceeds with the following sections: Section 2 conveys the existing literature survey. Section 3 deals with the proposed methodology for HTTP flooding attack detection. Experimental results and performance comparison of HAP with existing methods are discussed in Section 4. Section 5 describes the parameter analysis of packet rate, RT, throughput and Round Trip Time (RTT) for the generated dataset. Finally, Section 6 concludes the paper outlining possible future avenues for improvement.

2 Related work

The IDS monitors the traffic in real time and raises an alarm when a malicious user is detected [28]. In signature-based IDS, the rules are generated manually based on the known attack patterns (e.g. Snort, Bro IDS etc.) to determine known attacks whereas the anomaly based IDS detects the unknown patterns or unusual behaviour. The increase in usage of internet traffic degrades the performance of IDS in detecting the HTTP flooding attacks. Therefore, it is necessary to enhance the efficiency and accuracy of HTTP flooding attack detection and thereby reducing false positives.

2.1 Web-based attacks

Most of the researchers have focused on payload-based intrusion detection using user behaviour analysis. Wang and Stolfo *et al.* proposed a technique payload based anomaly detector (PAYL) using 1-g byte frequency distribution of network traffic payloads to detect anomalies [29]. It preprocesses the features by constructing a vector with relative 1-g frequency count in the packet payload. Mahalanobis distance is used for measuring the incoming traffic. The detection rate is 60% and the false alarm is <1%. The limitation of this approach is that it cannot fight against mimicry attacks, and it inspects entire payload for detection.

Wang *et al.* proposed ANAGRAM by extracting the n-gram byte sequence information. ANAGRAM uses supervised learning for the classification of n-gram sequences into benign and malicious packets [30]. These sequences are processed by two separate Bloom Filters (BFs). ANAGRAM does not work well in the tremendous usage of bandwidth and data rate networks since n-grams are stored within BFs. In the detection phase, the score is measured based on the unobserved behaviour and anomalous count of n-gram. Perdisci *et al.* proposed McPAD using the sliding window of 2-g to measure the traffic payload [31]. It leads to high false positives and low detection rate.

Rieck *et al.* extracted higher order n-gram features and words from connection payloads [32]. They used unsupervised learning for anomaly detection. Jamdagni *et al.* proposed payload based IDS named RePIDS using three-tier iterative feature selection engine for feature subspace selection [33]. The input data is preprocessed using PCA. Mahalanobis distance is used to find the hidden correlation between the patterns and features. Oza *et al.* proposed HTTP flooding attacks detection using statistical methods such as pattern counting, chi-square distance analysis, and ad-hoc distance measures and preprocessed each HTTP request by using n-gram analysis [34]. These methods take more time for the identification of traffic and also results in false positives.

2.2 Application layer attacks

In order to detect attacks at the application layer, Choi *et al.* adopted HTTP GET flooding attack detection by analysing the amount of traffic, source IP address, destination IP address, source port number and destination port number and their entropies are computed [35]. It results in false positives and the accuracy of the model is 88%.

Chwalinski *et al.* proposed HTTP-GET flooding attack detection by the grouping of similar categorical input patterns

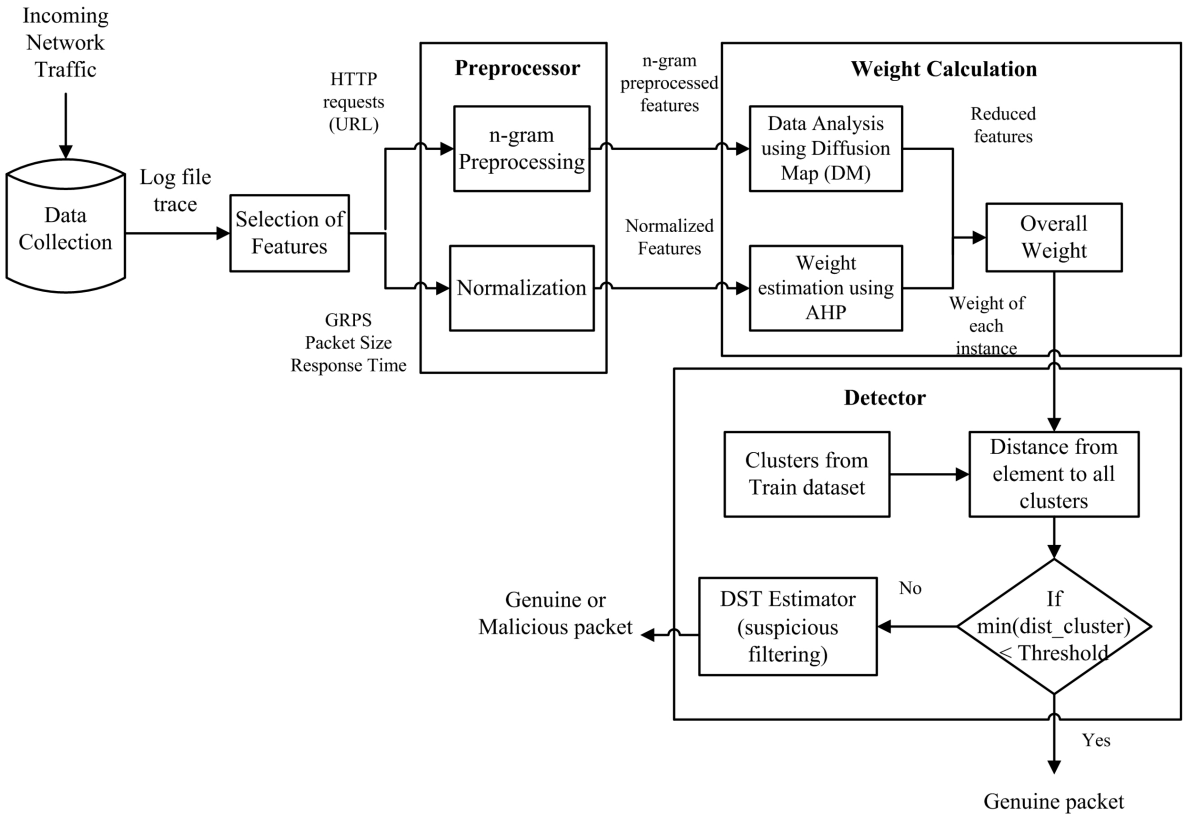

Fig. 1 *Flow diagram of HAP*

using clustering and information theoretic measures. This method distinguishes the benign and malicious sequences by analysing the behaviour of web request sequences [36]. This requires no prerequisite knowledge to identify the attack behaviour. It is difficult to find the number of clusters that had spread across the various entropy ranges because many requests sequence follow a uniform distribution.

Yatagai *et al.* proposed the HTTP GET flooding attacks detection method using correlation analysis or through the browsing behaviour of page [7]. This method cannot process huge data packets which results in false positives. Zhou *et al.* used the auto-regressive model for HTTP flooding attacks detection by analysing the real-time frequency vectors of the traffic and it is implemented at the backbone level [37]. The parameters are set on the basis of experience that leads to the time-consuming process.

Juvonen *et al.* proposed an online anomaly detection method by extracting the features from web server logs and reduces the dimensionality of the input features using PCA, RP and DM [15]. RP is used for the analysis of large volumes of traffic, whereas DM is mainly used for the accurate analysis of data and for better visualisation. Moreover, PCA identifies the incoming traffic as benign or malicious traffic but the identification of attack is not accurate. The fixing of the threshold is a challenging task to identify the attacks.

Frey *et al.* introduced the concept of identifying the exemplars by passing messages between data points (detecting patterns) through AP clustering [38]. Wei Wang *et al.* adopted a framework for autonomic intrusion detection using unlabelled data streams in networks [39]. It uses dynamic clustering technique for the grouping of similar streaming data. The autonomic framework is developed based on three aspects: self-managing, self-adapting and self-updating. The attack is detected from the real HTTP attack traffic and evaluated the effectiveness and efficiency against the statistical detection models. The limitation of this method is that it uses more features and it should adapt to the changes in detection parameters to improve the performance. This framework assumes that the benign and malicious traffic is in spherical shape.

Sreeram *et al.* proposed a bio-inspired detection algorithm for the identification of legitimate/malicious packets [22]. This model addressed the issue of session-based evolution models that are exposed to botnets. Jazzi *et al.* proposed non-parametric CUSUM-based application layer attacks in the presence of various sampling techniques [40]. The detection techniques may impact the detection performance and also the quality for the low rate of application layer attacks under different loads. Prasad *et al.* proposed a bio-inspired approach (cuckoo search) for the detection of application layer attacks with minimal complexity [41]. It cannot detect the suspicious source of attacks and flash crowds. Singh *et al.* proposed four different features to distinguish benign or malicious packets and the detection model is built using support vector machine-based classifiers for various HTTP GET flooding attack strategies [24]. The detection system requires historical web server logs to detect flooding attacks. It reduces complexity.

Agrawal *et al.* proposed a power spectral density based approach for the detection of low rate DoS attacks in the frequency domain [42]. The proposed method monitors the traffic in real time. This method is adaptive but the approach has a 3.7% False Positive Rate (FPR) and a 4.9% false negative rate. Vidal *et al.* proposed an approach for detecting and mitigating the DoS attacks on the basis of the emulation of the behaviour of an artificial immune system [43]. The proposed method gives better result in case of low-density traffic. However, as the congestion increases, the FPR also increases due to the large share of bandwidth. Moreover, the congestion problem is reduced by the activation of an adaptive response which helps to enhance the true positive rate of 13.7%.

In order to address the issues such as pattern matching, dynamic nature of network behaviour, and reduction in false positives, the HAP method is proposed.

3 Proposed method (HAP)

HAP consists of four major modules namely data collection, preprocessor, weight calculation, and detector module. Fig. 1 depicts the flow diagram of HAP. The notations and terminologies used in the algorithms are presented in Nomenclature.

HAP comprises four stages:

3.1 Data collection

HTTP requests are captured by using tools, scanning methods, and processed for each second and these packets are stored as the access log file of an Apache server. The log format of the access log file contains the information such as IP address, remote host, remote login name, remote user, timestamp (day/month/year:hr:min:s + zone), HTTP request (HTTP method, URL and HTTP version), HTTP status code, length of the data in bytes, referral URL and user agent. The sample access log is as follows:

*'10.20.28.157 - - [15/Feb/2016:08:58:34 + 700] 'GET/scripts/*root.exe?/c + dir HTTP/1.0' 200 378 'https://www.nitt.edu/OLCLD/view.php?q=book/' '-''

3.2 Preprocessor module

The input to the preprocessor module is the set of features viz., (GRPS, PS, RT and HTTP Request (URL)) that are extracted from the access log of an Apache server running in the cloud and the output of this stage is the normalised features. Depending on the feature, preprocessing is performed either by the normalisation process or through n-gram processing.

3.2.1 Preprocessing of GRPS, PS and RT: Let $\boldsymbol{y} = \{y_1, y_2, \ldots, y_m\}$ be an input feature vector collected at time t_1 to t_m. As the values of the input features are varying (both continuous and discrete values), it is normalised using min–max normalisation [15, 35, 44] and they are normalised in the range [0, 1] using the following equation:

$$\text{Normalisation} y_i^{\text{norm}} = \frac{y_i - \min(y)}{\max(y) - \min(y)}, \tag{1}$$

where y_i^{norm} represents the ith normalised feature value, y_i represents the ith actual feature value, min (y) is the minimum value of feature, i.e. min $\{y_1, y_2, \ldots, y_m\}$ and max (y) is the maximum value of feature, i.e. max $\{y_1, y_2, \ldots, y_m\}$.

For example, GRPS, PS and RT of an input instance are 18, 120, and 11, respectively. After substituting the values of an input instance (18, 120, 11) in (1), the normalised features of GRPS, PS and RT are obtained as $0.466, 0.5384$, and 0.2307, respectively. These normalised features are passed to the Analytical Hierarchical Process (AHP) module for the estimation of the weight of each feature.

The other feature HTTP request (URL) is preprocessed by n-gram analysis [34], which is illustrated in Section 3.2.2.

3.2.2 Preprocessing of HTTP requests (URL): The n-gram processing technique is used to extract and preprocess the HTTP request which reduces the noise and filters out the requests such as .html, .text, .pdf etc. The HTTP requests are American Standard Code for Information Interchange (ASCII) coded and converted into numerical vectors to form the feature matrix. When $n = 1$, n-gram analysis is performed on the character distribution with the maximum of 256 dimensions. Out of which, the ASCII code of 1-g appears from 33 to 127 (total 95 of ASCII printable characters, i.e. (from ! to DEL)) in the HTTP requests. Similarly, for higher dimensionality, the size of the feature is increased based on the values of n. The HTTP request (URL) is computed based on the frequency of ASCII code using n-gram analysis.

For example, the feature matrix of printable ASCII codes of HTTP request (URL) *'/scripts/root.exe?/c + dir'* for 2-g is computed as (see equation below). These preprocessed n-gram features are passed to the DM module for the reduction of the dimensionality of features to determine the attacks efficiently.

3.3 Weight calculation module

As the normalised features vary for each and every packet, it is necessary to compute the weight of each feature. In HAP, AHP is used for the estimation of weight. AHP is a multi-criteria decision-making process that breaks down a complicated problem into subproblems as decision factors and the weights are computed based on the relative importance to achieve goal [45, 46].

3.3.1 Weight estimation using AHP: AHP computes the weight of each IP address. The steps involved in AHP are described as follows:

- *Structuring features into hierarchical order* – it follows top-down approach. The objective is to select the weights (mass) of each request and it is represented at the top level. The key features of each packet are at the next level and an alternative IP address is at the bottom level.
- *Estimate the local weights of each feature* – the local weight of each feature is found by constructing the pairwise comparison matrix of features and the weight vectors of the matrices are calculated. The eigenvalues and eigenvectors are calculated for each pairwise comparison matrix. After the calculation of local weights of the feature, the consistency of comparison matrix is computed for all the features using Algorithm 1 (Fig. 2). Consistency Ratio (CR) is the ratio of the Consistency Index (CI), which is related to the eigenvalue method and the Random Index (RI). RI is the CI of a randomly generated pairwise comparison matrix [45]. RI depends on the number of elements being compared (i.e. size of the pairwise comparison matrix of features (GRPS, PS, and RT). According to Saaty [45], if the size of the pairwise comparison matrix is 3, then the corresponding RI value 0.58 is chosen for calculating the CR, which is described in Algorithm 1 (Fig. 2).
- *Synthesise the final weights (mass) of each request in the network* – this helps to estimate the overall weights (mass) value of all the requests (IP address) in the cluster. The weight of each IP address is computed as

$$Ws_{mi} = \sum_{j=1}^{m} Ws_{mi/j} * Ws_j, \tag{2}$$

where Ws_{mi} is the final mass value (weight) of feature i, $Ws_{mi/j}$ is the mass value of the ith IP address with the relevant feature j and Ws_j is the mass value of feature j. Table 1 indicates the standard value for RI. Weight calculation using AHP is depicted in Algorithm 1 (Fig. 2).

3.3.2 Data analysis using DM: The n-gram preprocessed features are relatively greater in size; hence the dimensionality reduction technique DM is used to reduce the size of the features. DM has the property of converting the features of high-dimensional input into the features of very low-dimensional input without having any variation in information content. It is a non-linear geometric method that maintains the diffusion distance as a Euclidean distance in lower dimensions [15, 18]. The diffusion matrix is computed by taking the pairwise distances between the input features and decomposed using Singular Value Decomposition (SVD). The steps involved in M are as follows:

(i) Let the feature be represented as $Y = \{y_1, y_2, \ldots y_N\}$, $y_i \epsilon R^n$, $i = 1, 2, \ldots N$, where N is the number of input features, and n is the dimension of the original features.
(ii) These input feature vectors are normalised by taking the logarithm to make the features more comparable. The features which have all zero rows and columns are removed.

URL:	/s	sc	cr	ri	ip	pt	ts	s/	/r	ro	oo	ot	t.	.e	ex	xe	e?	?/	/c	c+	+d	di	ir
Frequency:	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(iii) The affinity matrix ϕ is computed from these normalised features by taking the pairwise distance between input features. These distances are measured using Gaussian kernel function with Euclidean distance which is expressed as

$$\phi_{ij} = \exp\left(\frac{-\parallel y_i - y_j \parallel^2}{\epsilon}\right). \tag{3}$$

(iv) The diagonal matrix is obtained by adding each row of the affinity matrix ϕ, which is expressed by $D_{ii} = \sum_{i=1}^{n} \phi_{ij}$. These rows ϕ are normalised by the row sums, which is represented by $A = D^{-1}\phi$.

(v) The eigenvalues of $\boldsymbol{A}$ and $\tilde{\boldsymbol{A}}$ are the same with the conjugate matrix, which is expressed using $\tilde{\boldsymbol{A}} = D^{1/2}AD^{-1/2}$.

(vi) The symmetric matrix is obtained by substituting matrix $\boldsymbol{A}$ in step (v) and expressed using $\tilde{\boldsymbol{A}} = D^{-1/2}\phi D^{-1/2}$.

(vii) The above symmetric matrix is then decomposed using SVD [47]. The property of SVD is that it breaks down the matrix $\tilde{\boldsymbol{A}} = U\Lambda U^{\mathrm{T}}$ into U, i.e., $(U = u_1, u_2, \ldots u_N)$ that are almost matches with the N eigenvectors on its columns of $\tilde{\boldsymbol{A}}$ and also eigenvalues of the diagonal $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots \lambda_N)$ correspond to the eigenvalues of $\tilde{\boldsymbol{A}}$ since it is symmetric.

(viii) The decomposition matrix $\tilde{\boldsymbol{A}}$ and the eigenvalues of the transition matrix $\boldsymbol{A}$ are the same, but the eigenvectors are computed using $V = D^{-1/2}U$.

(ix) The low-dimensional features are obtained by multiplying each eigenvector column with the corresponding eigenvalue. The resulting matrix contains N rows that are exactly the same as the input features (data) points and k-columns that represent the new dimensions: $Y_{\mathrm{DM}} = V\Lambda$.

These low-dimensional features are passed to the AP clustering for the detection of flooding attacks.

3.4 Detector module

3.4.1 AP clustering: Clustering plays a significant role in a grouping of relevant features to identify the patterns. Most of the clustering algorithms are static in nature, which is not applicable for identifying the dynamic variations of traffic patterns. Moreover, the detection algorithms lack in the selection of optimal cluster centres and require specifying the number of clusters prior to determining anomalies. Hence, HAP uses AP clustering technique to dynamically monitor the large stream of incoming traffic to detect the anomalies that emerge over time [38].

AP clustering considers a set of real-valued similarities by passing a real-valued message between normalised features and the process is refined iteratively to attain a high-quality set of exemplars and also it evolves a large number of clusters that emerge gradually [39, 48].

Let $Y = \{Y_1, Y_2, \ldots Y_n\}$ denote the set of instances to be clustered and $d(Y_l, Y_m)$ represent the Euclidean distance between instance l and m. The sum of the Euclidean distance between the instances and their exemplars should be minimised. The fitness function of AP clustering is defined by

$$E(e) = \sum_{l=1}^{n} \mathrm{Sim}(Y_l, Y_{e(l)}), \tag{4}$$

where $e(l)$ represents the exemplar of an instance for the particular item Y in a cluster. Given an M input dataset, Y_l and Y_m are two features in it. The similarity $\mathrm{Sim}(Y_l, Y_m)$ indicates how well Y_m is suited to be an exemplar for Y_l and it is expressed by

$$\mathrm{Sim}(Y_l, Y_m) = \begin{cases} -d(Y_l, Y_m)^2 & l \neq m, \\ p & \text{otherwise}, \end{cases} \tag{5}$$

where $d(Y_l, Y_m)$ represents the Euclidean distance between l and m, p indicates how well an instance is chosen to be an exemplar. If there is no heuristic knowledge, self-similarities are called as preference (p), which is set as constant. For instance, preference (p) is indicated by

$$p = \frac{\sum_{l,m=1, l\neq m}^{M} \mathrm{Sim}(l,m)}{M*(M-1)}; \quad 1 \leq l \leqslant M\,. \tag{6}$$

The steps involved in the grouping of relevant features to identify anomaly using AP clustering are as follows:

(i) The responsibility $\mathbf{res}(l, m)$ and availability $\mathbf{av}(l, m)$ are the two matrices used to exchange the real-valued message between the features l and m. These matrices are initialised to zero.

(ii) The similarity matrix between these input features is computed by using (5) and (6).

(iii) Responsibility $\mathbf{res}(l, m)$ indicates how well the message is transferred between features l and the exemplary m and it checks how well it suits the feature m to the exemplar of a feature l, which is expressed as

$$\mathbf{res}(l,m) = \mathrm{Sim}(l,m) - \max_{\bar{m}\neq m} \{\mathrm{av}(l,\bar{m}) + \mathrm{Sim}(l,\bar{m})\}\,. \tag{7}$$

(iv) Similarly, availability $\mathbf{av}(l, m)$ indicates the data sent from the candidate exemplar m to the data feature l. The availability is measured using the following equation:

$$\mathbf{av}(l,m) = \begin{cases} \min\left\{0, \mathrm{res}(m,m) + \sum_{\bar{l}\neq l,m} \max\left\{0, \mathrm{res}(\bar{l},m)\right\}\right\}, & l \neq m, \\ \sum_{\bar{l}\neq l} \max\left\{0, \mathrm{res}(\bar{l},m)\right\}, & l = m\,. \end{cases} \tag{8}$$

(v) In order to avoid numerical oscillations while transferring messages between data points, the damping factor $\gamma\epsilon[0, 1]$ is added which is expressed as

$$\begin{aligned} \mathbf{Res}_{i+1} &= (1-\gamma)\mathbf{Res}_i + \gamma(\mathbf{Res}_{i-1}), \\ \mathbf{Av}_{i+1} &= (1-\gamma)\mathbf{Av}_i + \gamma(\mathbf{Av}_{i-1})\,. \end{aligned} \tag{9}$$

where $\mathbf{Res} = \mathrm{res}(l, m)$ and $\mathbf{Av} = \mathrm{av}(l, m)$ denote the responsibility and availability matrices, respectively, and 'i' indicates the number of iterations.

Input: Normalized Features N
Output: Weight of an instance i

function $W_{AHP} = Weight\_calculation\_AHP(N)$
  for each IP address IP_i **do**
    Assign priority matrix of each decision criteria
    Compute pairwise matrix $GRPS, PS, RT$
    Estimate weight vector $BW = \lambda_{max}W$
  //Check the consistency
    Consistency Ratio $(CR) = \frac{Consistency\ Index(CI)}{Random\ Index(RI)}$; (RI =0.58)
    Consistency Index $(CI) = \frac{\lambda_{max}-n}{n-1}$
    $\lambda_{max} = \left(\frac{1}{n}\right) * \sum_{i=1}^{n}\left(\frac{BW_i}{W_i}\right)$
    if CR $\leq$ 0.1 **then**
      Decision criteria $\{GRPS, PS, RT\}$ is consistent
    else
      construct new matrix and check until CR $\leq$0.1
    end if
  //Compute the final weights (mass) of each IP address
    $W_{ni} = \sum_{j=1}^{n} W_{\frac{ni}{j}} * W_j$
  end for
end function

Fig. 2 *Algorithm 1: Weight_calculation_AHP*

Table 1 Standard RI values

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.25	1.49

(vi) These responsibility and availability matrices are updated iteratively to determine the local decision for the constant number of iterations. The availability and responsibility of the data points are combined to identify the exemplars, which are represented by using the following equation:

$$c_m \leftarrow \underset{1 \le m \le M}{\arg\max} \left[\text{res}(l,m) + \text{av}(l,m)\right]. \quad (10)$$

The results obtained from the AP clustering are the number of clusters with data members and its associated exemplars. These clusters are checked for consistency to determine the traffic as either normal or anomalous.

Consistency estimation: The process involved in the consistency estimation is explained in Algorithm 1 (Fig. 2). The threshold is determined by using the average Euclidean distance of the particular cluster and it dynamically varies from one cluster to other clusters. The special cases persist where AP clustering algorithm cannot identify the benign or malicious traffic due to the lack of information in training data. So, DST is employed to determine the suspicious traffic using the mass value of the clustered data, which is described in our previous work [26]. Algorithm 5 (Fig. 3) uses DST of evidence to detect the suspicious user, i.e. to check whether the user is benign or malicious.

3.4.2 Detection process: The algorithms used in HTTP flooding attacks detection are described in this section. The detection process involves training and testing phase.

Training algorithm: The features such as (GRPS, PS, RT and HTTP requests) are extracted and also preprocessed by normalisation or through n-gram processing. The normalised features (GRPS, PS, RT) are passed to AHP for the calculation of weight. Similarly, the n-gram preprocessed HTTP requests are fed to DM module to obtain the reduced features without changing the information content. The overall weight is calculated by varying the values of α and β. The values of α and β are set as 0.5, since equal priorities are given to the features. AP clustering is used to group the similar input patterns to form clusters. Each cluster consists of both normal and attack elements. A cluster is said to be consistent, if all the members are of the same type. The consistency is checked for all the clusters. If the consistency is <1, then the corresponding element is removed from the cluster and the element is added to the outlier. The attack members are separated from the normal and moved to outlier by comparing the weight and threshold value. The threshold of each cluster $T_c(i)$ is computed by finding the average Euclidean distance between exemplars to all members in cluster and this threshold varies from one cluster to other. The elements of the outlier are moved to the existing clusters by comparing the Euclidean distance and threshold of each cluster. A new cluster is created if no match is found. The process is repeated until the outlier list becomes empty and all the clusters consistency become 1. Training algorithm of HAP is depicted in Algorithm 2 (Fig. 4). The algorithm is executed and checked for every second interval. If a new attack occurs, it is compared with the existing data items in training set. Algorithm 3 (Fig. 5) and Algorithm 4 (Fig. 6) are the sub-functions used for the calculation of threshold of cluster $T_c(i)$ and to determine the distance from member to cluster.

Testing algorithm: During testing, Euclidean distance is computed for the new incoming data instance to the clusters. The minimum distance cluster m_c is found by comparing the distance between the new incoming packet to the member of the cluster and the computed distance m_c should be less than the threshold. Now, m_c contains the number of normal/attack clusters. If the instance i belongs to the attack cluster then the corresponding instance is an attack otherwise it is normal. The suspicious instance is determined using DST of evidence by computing the belief and plausibility of evidence. The belief function gives the level of confidence that the given packet is either benign or malicious based on the hypothesis

Input: C: Clusters from training, I: Instances
Output: Normal or Attack
procedure $Testing\_data(I, C, T_c)$
for all instances i, $i \epsilon I$ **do**
$D_i(C)$ = Distance(i,C)
Read CPU usage, throughput
Find min_dist_clusters m_c; such that $D_i(C) < T_c(m_c)$
if $m_c \neq \phi$ **then**
i corresponds to m_c
if m_c is attack cluster **then**
i is attack
else
i is normal
end if
else
i is suspicious
//Use DS_theory
for each suspicious instance s, $s \epsilon I$ **do**
//Compute belief and plausibility
$Bel(R) = \sum_{S \subseteq R \& S \in 2^\theta} m(S)$
$Pl(R) = \sum_{S \cap R \neq \phi \& S \in 2^\theta} m(S)$
//Combination of Evidence
if $R == \phi$ **then**
$[m_1 \oplus m_2](R) = 0$
else
$[m_1 \oplus m_2](R) = \frac{\sum_{S \cap T = R} m_1(S) m_2(T)}{1 - \sum_{S \cap T = \phi} m_1(S) m_2(T)}$
end if
if $Bel(R) \geq 0.5$ && CPU usage is high && throughput decreases **then**
i is attack
else
i is normal
end if
end for
end if
end for
end procedure

Fig. 3 *Algorithm 5: HAP testing algorithm*

Input: Trace log file
Output: Clusters

procedure $Training\_Algorithm()$
$GRPS \leftarrow \{GRPS_1, GRPS_2, ...GRPS_n\}$
$PS \leftarrow \{PS_1, PS_2, ...PS_n\}$
$RT \leftarrow \{RT_1, RT_2, ...RT_n\}$
$Req \leftarrow \{Req_1, Req_2, ...Req_n\}$
$nReq = nGram(Req)$
$N = Normalization(GRPS, PS, RT)$
W_{AHP}=$Weight\_calculation\_AHP(N)$
$W_{DM} = Diffusion\_Map(nReq)$ //Diffusion Map of n-gram request
$W_{OW} = \alpha * W_{AHP} + \beta * W_{DM}$
$S = similarity\_matrix(W_{OW})$
$C = AP\_clustering(S)$ //AP clustering of input similarity matrix
Monitors VM status, throughput
repeat
$outlier\_list\ O = \phi$
for each clusters i, $i \epsilon C$ **do**
Calculate consistency
if $consistency < 1$ **then**
Remove outliers from cluster and add to outlier
end if
end for
for each clusters i, $i \epsilon C$ **do**
calculate $T_c(i) = cluster\_Threshold(i)$ ▷ Threshold of cluster
end for
for each outlier o, $o \epsilon O$ **do**
for each cluster c, $c \epsilon C$ **do**
calculate $D_o(c) = Distance(o, C)$ ▷ Distance from member to cluster
end for
$m_c = \phi$
Find min_dist_cluster and add to m_c such that $D_o(c) < T_c(m_c)$
if $m_c \neq \phi$ **then**
Assign o to m_c
else
Create new cluster and add as a member to new cluster
end if
end for
Until outlier list is empty and consistency of all cluster must be 1
Return C
end procedure

Fig. 4 *Algorithm 2: HAP training algorithm*

function $T_c = Cluster\_Threshold(c)$ ▷ Find the threshold of cluster
  $T_c = 0$;
  for each element i, $i \epsilon C$ & $i \neq j$ **do**
    for each element j, $j \epsilon C$ **do**
      $T_c = T_c + E\_dist(i, j)$ ▷ Euclidean distance between exemplar to all members in cluster
    end for
  end for
  $T_c = T_c/2$ // Need only upper or lower triangular matrix
  $T_c = \frac{T_c}{count_c(c)}$ // $count_c(c)$ is the total number of members in cluster
  return T_c
end function

Fig. 5 *Algorithm 3: threshold of cluster $T_c(i)$*

function $D_o = Distance(member, C)$ ▷ Finding distance from member to cluster
  $D_o = 0$;
  for each member i, $i \epsilon C$ **do**
    $D_o = D_o + E\_dist(i, member)$
  end for
  return $\frac{D_o}{count_c(c)}$
end function

Fig. 6 *Algorithm 4: finding distance from member to cluster*

of R. The plausibility represents the degree to which the hypothesis fails to disbelieve R and the hypothesis of R should be in the interval $[\text{Bel}(R), \text{Pl}(R)]$. The binary case of the belief function is given as

$$\text{Bel(Normal)} = m\text{(Normal)},$$

$$\text{Bel(Attack)} = m\text{(Attack)},$$

$$\text{Bel(Attack/Normal)} = m\text{(Attack)} + m\text{(Normal)} + m\text{(Attack/Normal)}.$$

DST provides a method to combine the measure of evidence from different sources using Dempster's rule of combination. This method identifies the true statement of the hypothesis that the attack has occurred from its particular source. For example, consider the two instances that contain the belief values of normal and attack: Instance 1: $m1(\text{Attack}) = 0.6864$, $m1(\text{Normal}) = 0.2011$, $m1(\text{Attack/Normal}) = 0.1125$; Instance 2: $m2(\text{Attack}) = 0.7045$, $m2(\text{Normal}) = 0.2851$, $m2(\text{Attack/Normal}) = 0.1019$.

The combined measure of evidence is used to identify the particular source is malicious or not. After applying DST, the value of k is calculated by using the equation: $k = 1\text{-}(0.1956 + 0.1416) = 0.6628$. The combined belief of evidence is given as

$$m(\text{Attack}) = (1.508) * (0.4835 + 0.0792 + 0.699) = 0.9539,$$

$$m(\text{Normal}) = (1.508) * (0.0573 + 0.0204 + 0.0320) = 0.1654,$$

$$m(\text{Attack/Normal}) = (1.508) * (0.0114) = 0.01719.$$

According to the results, it is evident that the belief of an attack is >0.5 and the CPU usage is high and the throughput decreases to a greater extent which leads to almost no change in throughput, hence the suspicious instance is identified as attack instance otherwise it is the normal instance. Testing algorithm of HAP is depicted in Algorithm 5 (Fig. 3). Table 2 details the Basic Probability Assignment (BPA) of an instance.

4 Experimental analysis

The information about the publicly available dataset [49] and the generated datasets for various test cases used for testing and the experimental results are presented in this section.

Table 2 BPA of instance

m_1/m_2	{Attack}:0.6864	{Normal}:0.2011	{Uncertainty: normal, attack}: 0.1125
{attack}:0.7045	0.4835	0.1416	0.0792
{normal}:0.2851	0.1956	0.0573	0.0320
{uncertainty: normal, attack}: 0.1019	0.0699	0.0204	0.0114

4.1 Datasets used

The datasets used for the experimental analysis in HAP are publicly available [Canadian Institute of Cybersecurity (CIC)] dataset and the generated dataset.

4.1.1 Publicly available dataset: The experiments were tested by using CIC DoS dataset [49]. It consists of enormous application layer requests (HTTP GET, domain name system queries, and Session Initiation Protocol (SIP) invite) transmitted to the victim machine. These requests were mixed with attack free traces from the ISCX dataset with 12 attack strategies and the traffic is collected for 24-h duration with the total size of 4.6 GB.

4.1.2 Generated dataset: In addition to public datasets, HAP was tested by using the generated dataset for various test cases to identify whether the user is benign or malicious.

4.2 Experimental setup

HAP is tested on a private cloud which is built using OpenStack open source cloud computing software [50]. The experiments are conducted on virtualised instances created by OpenStack running on a network of high-end systems [51]. The experimental testbed consists of 50 client machines with two VM instances running on each client and high-end server connected through switches and routers. The large amount of HTTP traffic is flowing through the web server captured during 1 week period and stored as log file of various sizes in MB. The attack was subsequently executed during this period. Attacks are conducted through VMs on the same host, VMs in the other hosts, and from systems external to the experimental setup by sending malicious packets to the victim VM. Snort is set up in the node controller to track the network activity of the VMs. The web server access log trace is used to identify the particular source of evidence. Fig. 7 depicts the OpenStack cloud setup. HTTP flooding attack is categorised into three different categories on the basis of various attacking strategies.

(i) An attacker and the victim VM instance are on the same hosts, then there is a chance that the co-resident VM instances will also get affected.
(ii) If an attacker is on different hosts and sends the malicious request to the VM instance of other hosts in the same cloud. In this case, all the resources such as bandwidth, throughput etc., will get affected.
(iii) An attacker is not a part of the private cloud environment. Suppose there are only a few entry points then the attacker will target these entry points by flooding malicious requests to the victim VM, which is the part of another cloud. In this case, all the resources get affected by flooding attack.

The scenarios used for generating the HTTP flooding attack using the different parameters are configured in the following ways:

1. *HTTP flooding attack using botnet:* Bots are malicious codes that run automated scripts as with computer virus. The attacker uses malicious code to modify the packet parameters and sends enormous malicious requests to the victim VM. The botnets such as IRC bot [52] and Dirt Jumper [53] are used to fire HTTP flooding attacks.

2. *Simple flooding attack:* The attacker randomly sends a large amount of malicious packets to the victim VM instead of sending the packets at regular interval. This attack overwhelms entire resources running in the cloud. The tool used for generating the HTTP flooding attack is HTTP DDoS attack.
3. *Vulnerabilities by using HTTP attack tools*: The HTTP flooding attack is generated using HULK and HOIC attack tools. It generates malicious requests to attack the victim VM instance by varying various parameters such as random URLs, Keep Alive Open etc. [54, 55].
4. *Slow rate HTTP flooding attacks*: The attacker sends partial request by opening a large number of connections on the victim VMs thereby exhausting the cloud server with minimal bandwidth. This attack keeps the connection open as long as possible so that the benign request cannot be processed by the server. This type of attack is generated by Slowloris tool [56].
5. *Cross-site scripting (XSS) attacks:* In addition to HTTP flooding attacks, XSS attacks are also injected into the network traffic. Whenever a user views the sequence of pages, the attacker tries to execute the foreign malicious scripts, which results in XSS attacks.

4.3 Results and discussion

Depending on the feature, the raw logs accessed from the Apache web server are preprocessed using normalisation or through n-gram processing. The overall weight is determined by computing the weight of each instance (GRPS, PS, and RT) using AHP and also the reduction in dimensionality of the feature (HTTP requests) using DM. The overall weight values are calculated by varying all the possibilities of α and β, which is between 0 and 1, i.e. α, $\beta \in -[0, 1]$ and $\alpha + \beta = 1$. If the α value is increased, then it takes more (priority) weights in AHP features, which are not enough to determine the attacks accurately. Similarly, the weights of DM features (β) also cannot be increased which can misclassify the attacks. Therefore, the values of α and β are set to 0.5, since the features obtained from AHP and DM influence weight and gives the better result.

Fig. 8 presents low-dimensional input features using DM [15]. This details how the input features are represented in the feature space for better visualisation of low-dimensional features in DM and to make the analysis easier. DM identifies the features accurately and the execution time is relatively higher in comparison with other dimensional methods. Table 3 describes the dataset considered for several test cases using the different attacking tools.

The subsets of the raw logs are taken into consideration for testing the speed and scalability of the execution for handling large datasets. The variations of the real traffic log files are encountered for the identification of traffic. The scalability is achieved on the basis of the size of the raw logs. Fig. 9 shows the computational times of DM. The speed and scalability of the approach are tested for different log instances of the generated dataset to measure the computational time of the reduced feature. It is observed from Fig. 9, the data grows linearly in terms of scalability aspects. As the number of entries in logs increases, the computational times grows linearly.

The data obtained from DM helps to construct the similarity matrix of the features. Once the real-valued pairwise similarity matrix is computed, AP clustering technique groups the similar input features. Each cluster is represented by an exemplar. For N input features there are $N^2 - N$ pairwise similarities. AP clustering automatically identifies the size of clusters from the input preference. This preference indicates how well input features suits an exemplar of a cluster. However, the preference p is set as the median of the similarities which gives the better results for the grouping of relevant data items. The output of AP is the number of clusters.

Fig. 10 presents the variations of members in cluster 1. The Euclidean distance is computed for the new incoming packets (test data) to all the members in the cluster. The figure demonstrates the

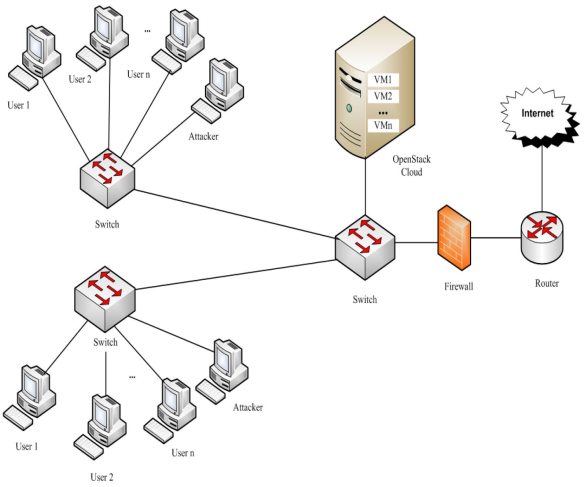

Fig. 7 *Experimental setup*

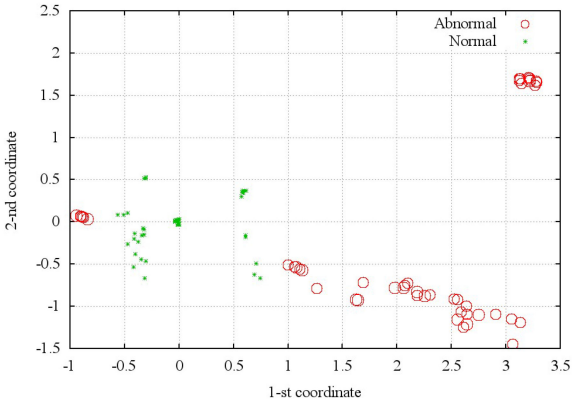

Fig. 8 *Low-dimensional points using DM*

Table 3 Tools used to generate datasets for various tests

Test cases	Tools used
1	HOIC
2	HULK, BOT, XSS
3	HTTP DDoS, Slowloris
4	HULK, HTTP DDoS, XSS
5	HULK, HOIC,BOT

Euclidean distance between all the members in cluster 1 along with the threshold and the thresholds vary from one cluster to other.

Fig. 11 shows the accuracy of HAP for various n-grams of the different test cases for flooding attack detection. The accuracy of the HTTP flooding attack detection is measured by the following equation:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{11}$$

where True Positive (TP) - Number of attack instances correctly classified as attack. False Positive (FP) - Number of normal instances incorrectly classified as attack.

True Negative (TN) - Number of normal instances correctly classified as normal.

False Negative (FN) - Number of attack instances incorrectly classified as normal.

The experimental results indicate that the HAP enhances detection accuracy with fewer false alarms for the different number of tests. It is inferred from the experimental results that the accuracy of 2-g for test case 1 is 7.02, 2.16, and 2.89% higher than 1-, 3- and 4-g, respectively. Similarly, the accuracy of 2-g for test case 2 increases to 8.16, 3.42, and 3.83%, respectively, and so on. It is observed that HAP achieves the best accuracy for 2-grams of all test cases because the actual feature size is less than the high

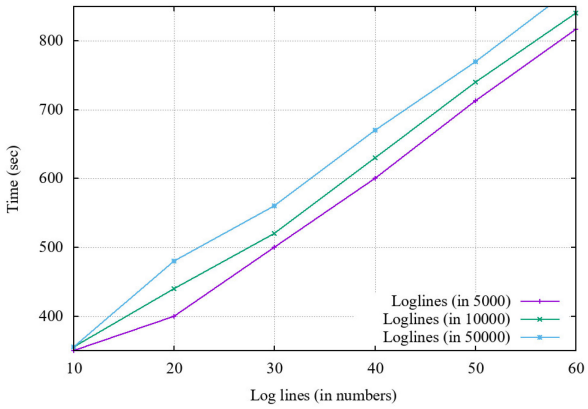

Fig. 9 *Computational times of DM*

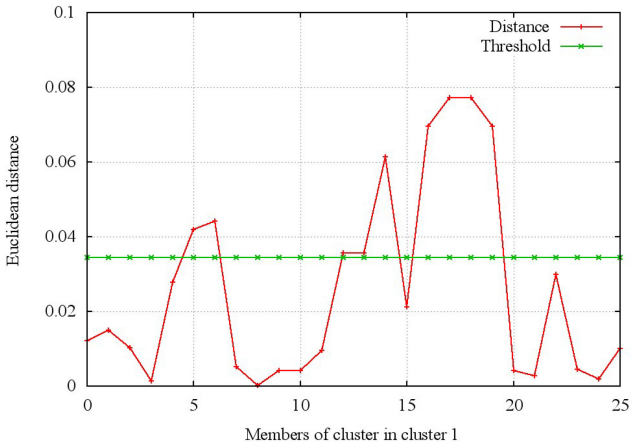

Fig. 10 *Variations of members in cluster 1*

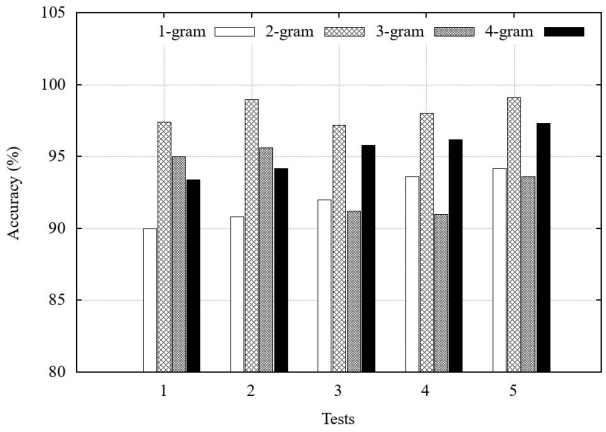

Fig. 11 *Accuracy of HAP for n-grams of various tests*

dimensional features and the high-dimensional feature would slow down the detection performance.

Fig. 12 shows the FPR of attack detection for different n-grams of various tests. The experimental results demonstrate that the HAP achieves fewer FPR for the different number of tests. Test case 1 gives large positive since the system assumed as normal traffic and processes all the requests in the server. In test cases 2 and 4, the server identifies the traffic as normal and it yields fewer false alarms.

4.4 Performance comparisons

KNN for attack detection: The low-dimensional data obtained from DM is passed to KNN for the identification of normal or abnormal behaviour. Let $X = \{x_1, x_2, \ldots, x_n\}$, $X \varepsilon R^D$ be the training data of ' D ' dimension and n represents the number of preprocessed instances of the training dataset. For a given k, determine the KNN for each

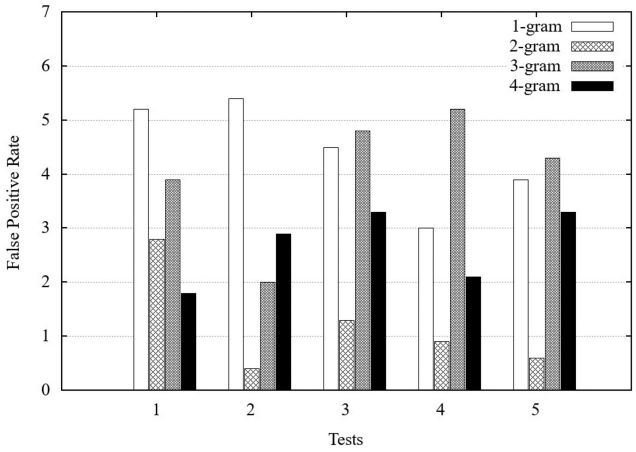

Fig. 12 *FPR of HAP for n-grams of various tests*

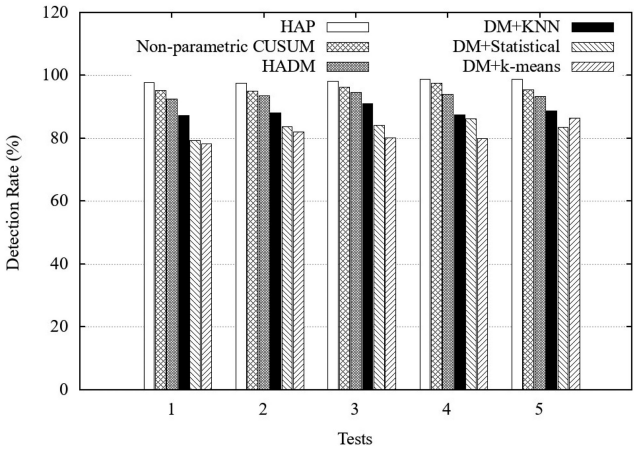

Fig. 13 *Detection rate of HAP with existing methods for generated dataset*

instance in the training set [27, 57]. The distance between the new data item (test) and KNN of the training data item are measured. The Euclidean distance scores are sorted and also the KNN is used to identify whether the new data item is benign or not. The anomaly index is computed by taking the average of the k-closest distance. If a given new data item falls above the threshold, then the given data item is categorised as anomalous otherwise the new data item is considered normal.

Fig. 13 depicts the performance comparison of the detection rate of HAP with the existing method such as non-parametric CUSUM, HADM, DM + KNN, DM + statistical, and DM + k-means method. It is observed from the experimental results that the detection rate of HAP for test case 1 is 5.23, 6.45, 9.57, 12.90, and 15.05% higher than non-parametric CUSUM, HADM, DM + statistical, DM + k-means, and DM + KNN method, respectively. Similarly, the HAP increases 7.36, 9.79, 12.63, 15.65, and 18.47% detection rate for test case 2 than non-parametric CUSUM, HADM, DM + statistical, DM + k-means and DM + KNN method, respectively. HAP analyse large amounts of packets with less processing time and also detects the suspicious user as either normal or anomalous than other existing methods. Similarly, a non-parametric CUSUM is used for the comparison with the HAP to identify the variations of the possible shifts due to its simplicity and low computational overhead.

Fig. 14 shows the performance comparison of the FPR of the HAP with the existing methods. It is inferred from Fig. 14, the FPR of the HAP is less than existing methods since HAP identifies the suspicious user using DST of evidence. The FPR is high for test case 1 for the DM + K-means method since it requires predefining the number of clusters in advance and also it leads to misclassify the benign or malicious behaviour. Similarly, the FPR of the HAP is less than the DM + statistical method [15] since the detection is found by analysing the mean and standard deviation of the training input data set to determine the patterns. So, there is a chance to

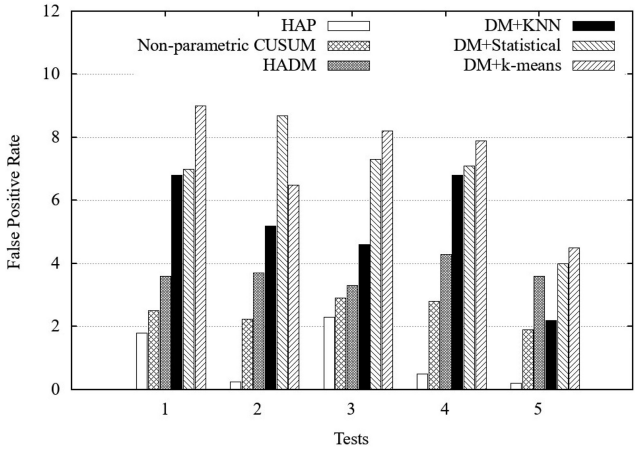

Fig. 14 *Performance comparison of FPR with various methods for generated dataset*

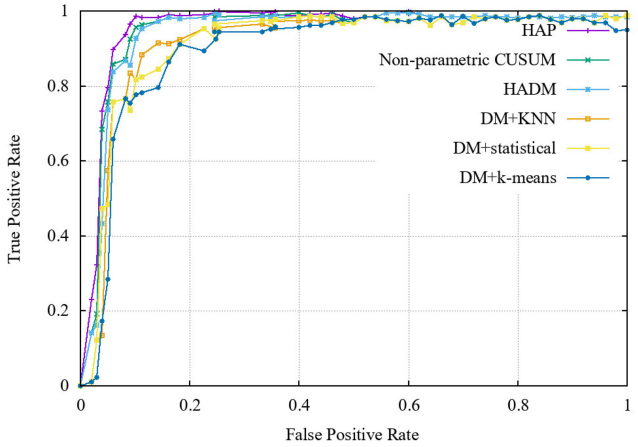

Fig. 15 *ROC curve of HAP with existing methods for generated dataset*

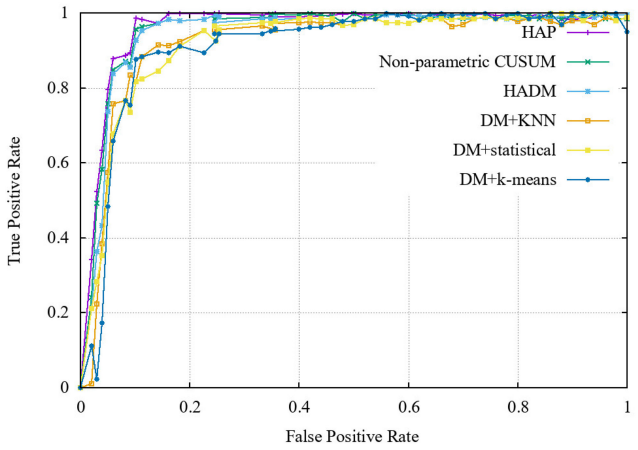

Fig. 16 *ROC curve of HAP with existing methods for CIC DoS dataset*

misclassify the attack behaviour. In addition, the FPR of HAP is less than the KNN method since it detects the attack iteratively and the suspicious user is identified using DST.

Table 4 illustrates the comparison results of HAP with the existing methods. The detection rate of HAP is 98.48% than the other method viz., non-parametric CUSUM, HADM, DM + KNN, DM + statistical method and DM + *k*-means, which has the detection rate of 96.4, 92.63, 88.4, 83.25 and 82.6%, respectively, since the AP clustering groups the patterns based on the input similarities of the features. The uncertainty of HAP is very less than the other methods since it determines the suspicious source using DST of evidence. The description of the uncertainty of the HAP is represented in Algorithm 5 (Fig. 3).

Table 4 Comparison between HAP and previous works

	Simulated/real environment	Detection rate, %	FPR, %	Uncertainty
Juvonen and Sipola [18]	simulated	82.6	9.35	high
Li and Zhang [27]	simulated	83.25	7.55	medium
Juvonen *et al.* [15]	simulated	88.4	6.32	medium
Sree and Bhanu [26]	simulated	92.63	4.23	low
Jazi *et al.* [40]	simulated	96.4	2.38	medium
HAP	**OpenStack cloud**	**98.48**	**0.9**	**very low**

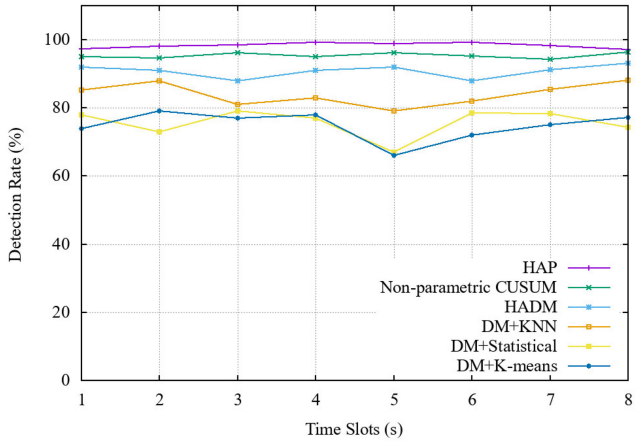

Fig. 17 *Detection rate of HAP with existing methods for CIC DoS dataset*

Fig. 15 shows the Receiver Operating Characteristic (ROC) curve of HAP with existing methods viz., non-parametric CUSUM, HADM, DM + statistical, DM + KNN and DM + *k*-means. It is observed from the figure, the true positive rate of HAP is higher than the existing methods. This is due to the autonomic clustering of the similar input patterns for the detection of benign or malicious behaviour. The suspicious source (unknown attacks) could not be identified by using existing methods and these methods fail to classify the benign as a malicious user. However, in HAP, DST is used to identify the suspicious source as either normal or anomalous. Moreover, the detection rate of the HAP is high and it achieves fewer false alarms than the existing methods.

In addition, the performance of the HAP was evaluated using publicly available CIC DoS datasets [49]. In order to label the web access log of CIC DoS dataset, a *k*-means clustering algorithm is used. The dataset is preprocessed by extracting the features such as GRPS, PS, RT, HTTP requests, throughput, and RTT. These normalised features are fed to the detector module to identify whether the user is legitimate or anomalous. Fig. 16 presents the ROC curve of HAP, non-parametric CUSUM, HADM, DM + statistical, DM + KNN, and DM + *k*-means. It is inferred from the figure that HAP achieves more accuracy than the existing methods since it identifies the suspicious source of evidence.

The detection rate of CIC DoS attack dataset for HTTP flooding attack is interpreted in Fig. 17. It is noticed from the experimental results that the detection rate of HAP for slot 1 is 4.32, 8.82, 16.98, 24.84, and 28.3% higher than non-parametric CUSUM, HADM, DM + KNN, DM + statistical method and DM + *k*-means method, respectively. Similarly, for slot 2, the detection rate of HAP are 4.97, 9.3, 13.32, 31.3, and 22.4% more than the detection rate of non-parametric CUSUM, HADM, DM + KNN, DM + statistical method, and DM + *k*-means, respectively, and so on. This is due to the fact that if an unknown attack occurs, the existing method fails to detect the anomalies and it predicts as benign traffic. The DM + statistical method performs more computation to detect attacks.

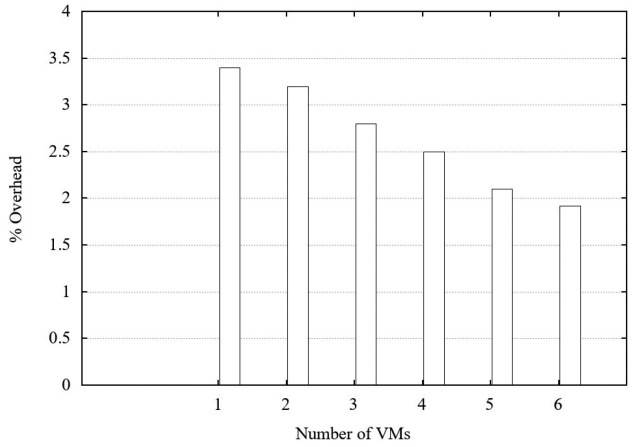

Fig. 18 *Performance overhead of HAP*

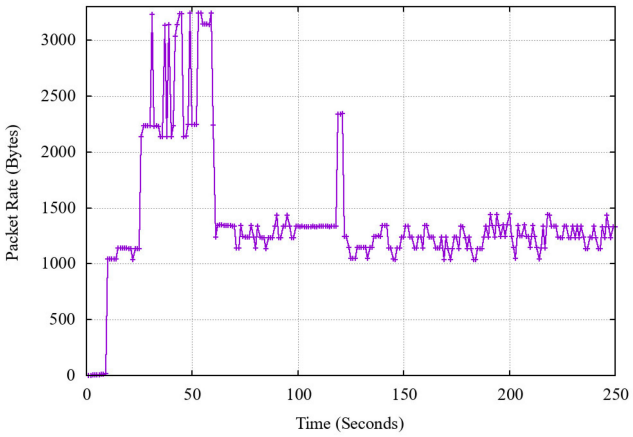

Fig. 19 *Average packet rate of generated dataset*

4.5 Performance overhead

Fig. 18 presents the performance overhead of the HAP framework. The performance degradation of the hosts running on the HAP framework is measured by monitoring the CPU performance of VM instance using Sysbench. The CPU performance of the host running on the system in a network is measured by hosting a different number of VM instances by executing the ping command from the VM instance running in the same hosts or through different hosts. It is inferred from the experimental results that the performance overhead of HAP is 3.4, 3.23, 2.88% and so on for the number of VM instances running in the cloud. As the size of the VM instances increases a huge number of packets are to be monitored for identifying the flooding attacks. This can be accomplished by preprocessing the packets using DM. DM uses the non-linear geometric method that preserves the diffusion distance as a Euclidean distance in lower dimensions to identify the traffic. Hence, HAP overhead is low.

5 Parameter analysis

The parameters such as packet rate, RT, throughput and RTT are estimated from the real traces of incoming traffic.

5.1 Packet rate

Fig. 19 shows the average packet rate of generated traffic. It is inferred from the figure that average packet rate is ∼1450 bytes/s, whereas in case of flooding attacks the average packet rate increases to a greater extent due to the heavy flow of traffic. There is a sudden increase in traffic that occurs due to attack from 27 seconds to 62 seconds.

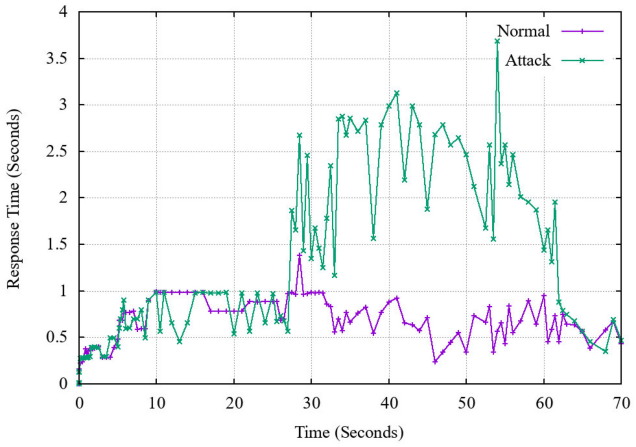

Fig. 20 *Average RT of generated dataset*

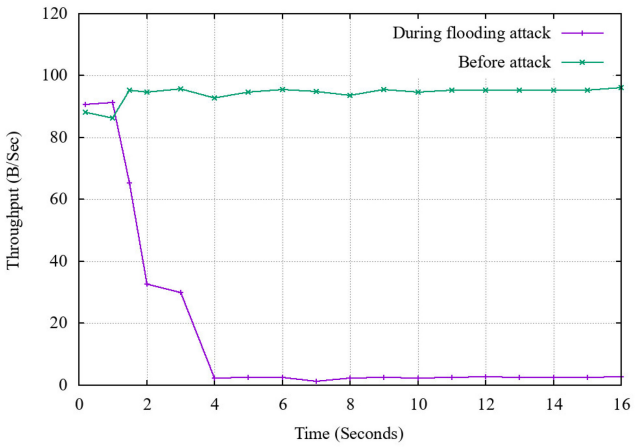

Fig. 21 *Average throughput of generated dataset*

5.2 Response time

Fig. 20 presents the average RT during normal and attack traffic. It is observed from the figure that RT is 1.1 second before an attack. When an attack is launched, the RT starts increasing with increase in attack strength at 27.5 second due to network load and congestion.

5.3 Throughput

Fig. 21 presents throughput values of before and during flooding attacks on web servers. From the figure, it is inferred that the throughput value decreases to the greater extent after HTTP flooding attacks. During HTTP flooding, throughput drops to 0.38 Mbps and it is almost constant after 4 second and it was between 0.38 and 0.47 Mbps whereas before flooding attacks occur, throughput value is almost constant at 95.3 Mbps. This is due to the fact that an attacker sends a huge volume of malicious packets to the victim thereby causing network congestion.

5.4 Round trip time

Fig. 22 depicts RTT for 16 runs before and during flooding attacks. The result shows that the average RTT value before flooding attack was 2.3 ms. During flooding attacks, the average RTT value is increased to 31.8 ms. This is due to the fact that the network is congested with the heavy flow of traffic.

6 Conclusion

The state-of-the-art clustering techniques such as *k*-means, hierarchical, density based etc. are highly sensitive to cluster centre initialisation. These methods require specifying the number of clusters beforehand and also stuck in local optima to detect the unknown attacks. This study discovers the various combinations of

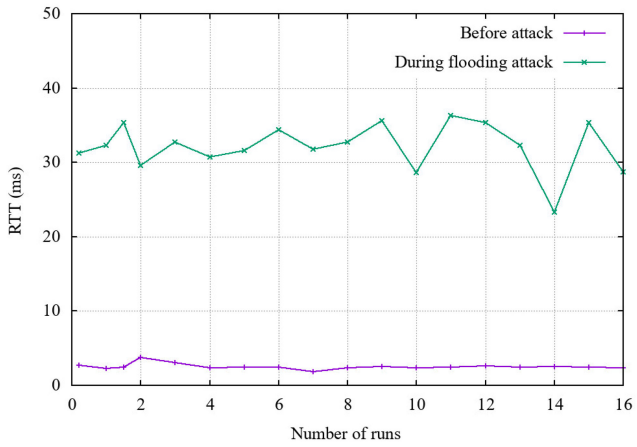

Fig. 22 *RTT 16 runs before and during flooding attacks*

attack features and also differentiates between the attacker, benign user, and suspicious user. The preprocessed relevant features are accessed from the web server log files, which are then fed to the DM module to obtain the reduced features without having any variations in information content. The AP clustering algorithm groups the reduced features obtained from DM into the number of clusters on the basis of input preference. The special cases persist where AP clustering could not identify the traffic as either normal or malicious. In this case, the suspicious traffic is identified from the mass value using DST of evidence. The experimental results demonstrate that the detection rate of the HAP is higher than the existing methods such as non-parametric CUSUM, HADM, DM + statistical, DM + *k*-means and DM + KNN method with fewer false alarms. In future, the incremental clustering techniques will be used to increase the efficiency and to further reduce the false alarms. Furthermore, this study will be extended to distinguish between HTTP flooding attacks from the flash crowd by analysing the several attack behaviours.

7 References

[1] Internet.: 'Internet statistics'. Available at http://www.internetlivestats.com/internet-users/ 2015 (accessed May 2018), 2015

[2] Bhuyan, M.H., Kashyap, H., Bhattacharyya, D., *et al.*: 'Detecting distributed denial of service attacks: methods, tools and future directions', *Comput. J.*, 2013, **57**, (4), pp. 537–556

[3] DDoSquick.: 'DDoS quick guide'. US-cert,' Available at https://www.us-cert.gov/security-publications/DDoS-Quick-Guide', 2015 1–4, 2014 (accessed 10 January 2016)

[4] Scarfone, K., Mell, P.: 'Guide to intrusion detection and prevention systems (IDPS)', *NIST Spec. Publ.*, 2007, **800**, p. 94

[5] Liao, H.J., Lin, C.H.R., Lin, Y.C., *et al.*: 'Intrusion detection system: a comprehensive review', *J. Netw. Comput. Appl.*, 2013, **36**, (1), pp. 16–24

[6] Patcha, A., Park, J.M.: 'An overview of anomaly detection techniques: existing solutions and latest technological trends', *Comput. Netw.*, 2007, **51**, (12), pp. 3448–3470

[7] Yatagai, T., Isohara, T., Sasase, I.: 'Detection of HTTP-GET flood attack based on analysis of page access behavior'. 2007 IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing, Victoria, BC, Canada, 2007, pp. 232–235

[8] Giralte, L.C., Conde, C., De Diego, I.M., *et al.*: 'Detecting denial of service by modelling web-server behaviour', *Comput. Electr. Eng.*, 2013, **39**, (7), pp. 2252–2262

[9] Zhang, Z., Li, J., Manikopoulos, C., *et al.*: 'Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification'. Proc. IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 2001, pp. 85–90

[10] Govindarajan, M., Chandrasekaran, R.: 'Intrusion detection using neural based hybrid classification methods', *Comput. Netw.*, 2011, **55**, (8), pp. 1662–1671

[11] Hu, W., Liao, Y., Vemuri, V.R.: 'Robust anomaly detection using support vector machines'. Proc. Int. Conf. on Machine Learning, United States, 2003, pp. 282–289

[12] Fodor, I.K.: 'A survey of dimension reduction techniques', 2002, pp. 1–18

[13] Sipola, T., Juvonen, A., Lehtonen, J.: 'Dimensionality reduction framework for detecting anomalies from network logs', *Eng. Intell. Syst.*, 2012, **20**, (1/2), pp. 87–97

[14] Hyvärinen, A.: 'Survey on independent component analysis', *Neural Comput. Surv.*, 1999, **2**, pp. 94–128

[15] Juvonen, A., Sipola, T., Hämäläinen, T.: 'Online anomaly detection using dimensionality reduction techniques for http log analysis', *Comput. Netw.*, 2015, **91**, pp. 46–56

[16] Lee, Y.J., Yeh, Y.R., Wang, Y.C.F.: 'Anomaly detection via online oversampling principal component analysis', *IEEE Trans. Knowl. Data Eng.*, 2013, **25**, (7), pp. 1460–1470

[17] Sipola, T., Juvonen, A., Lehtonen, J.: 'Anomaly detection from network logs using diffusion maps', in Iliadis, L., Jayne, C. (eds.): '*Engineering Applications of Neural Networks*' vol. **363** (Springer, Berlin, Heidelberg, 2011), pp. 172–181

[18] Juvonen, A., Sipola, T.: 'Adaptive framework for network traffic classification using dimensionality reduction and clustering'. 2012 4th Int. Congress on Ultra Modern Telecommunications and Control System and Workshops (ICUMT), St. Petersburg, Russia, 2012, pp. 274–279

[19] Lee, W., Stolfo, S.J., Mok, K.W.: 'A data mining framework for building intrusion detection models'. Proc. 1999 IEEE Symp. on Security and Privacy, Oakland, CA, USA, USA, 1999, pp. 120–132

[20] Bhaya, W., Manaa, M.E.: 'Review clustering mechanisms of distributed denial of service attacks', *J. Comput. Sci.*, 2014, **10**, (10), pp. 2037–2046

[21] Agrawal, S., Agrawal, J.: 'Survey on anomaly detection using data mining techniques', *Procedia Comput. Sci.*, 2015, **60**, pp. 708–713

[22] Sreeram, I., Vuppala, V.P.K.: 'Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm', *Appl. Comput. Inf.*, 2017

[23] Lee, K., Kim, J., Kwon, K.H., *et al.*: 'DDoS attack detection method using cluster analysis', *Expert Syst. Appl.*, 2008, **34**, (3), pp. 1659–1665

[24] Singh, K., Singh, P., Kumar, K.: 'User behavior analytics-based classification of application layer http-get flood attacks', *J. Netw. Comput. Appl.*, 2018, **112**, pp. 97–114

[25] Beyon, M.: 'DS/AHP method: a mathematical analysis, including an understanding of uncertainty', *Eur. J. Oper. Res.*, 2002, **140**, (1), pp. 148–164

[26] Sree, T.R., Bhanu, S.M.S.: 'HADM: detection of http get flooding attacks by using analytical hierarchical process and Dempster–Shafer theory with MapReduce', *Sec. Commun. Netw.*, 2016, **9**, (17), pp. 4341–4357

[27] Li, Y., Zhang, X.: 'Diffusion maps based k-nearest-neighbor rule technique for semiconductor manufacturing process fault detection', *Chemometr. Intell. Lab. Syst.*, 2014, **136**, pp. 47–57

[28] Wang, W., Guan, X., Zhang, X., *et al.*: 'Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data', *Comput. Secur.*, 2006, **25**, (7), pp. 539–550

[29] Wang, K., Stolfo, S.J.: 'Anomalous payload-based network intrusion detection', in Jonsson, E., Valdes, A., Almgren, M. (eds.): '*Recent Advances in Intrusion Detection*' vol. **3224** (Springer, Berlin, Heidelberg, 2004), pp. 203–222

[30] Wang, K., Parekh, J.J., Stolfo, S.J.: 'Anagram: a content anomaly detector resistant to mimicry attack'. *Recent Advances in Intrusion Detection*' vol. **4219** (Springer, Berlin, Heidelberg, 2006), pp. 226–248

[31] Perdisci, R., Ariu, D., Fogla, P., *et al.*: 'McPAD: a multiple classifier system for accurate payload-based anomaly detection', *Comput. Netw.*, 2009, **53**, (6), pp. 864–881

[32] Rieck, K., Laskov, P.: 'Language models for detection of unknown attacks in network traffic', *J. Comput. Virol.*, 2007, **2**, (4), pp. 243–256

[33] Jamdagni, A., Tan, Z., Nanda, P., *et al.*: 'Intrusion detection using geometrical structure'. Fourth Int. Conf. on Frontier of Computer Science and Technology, 2009 (FCST'09), Shanghai, China, 2009, pp. 327–333

[34] Oza, A., Ross, K., Low, R.M., *et al.*: 'Http attack detection using n-gram analysis', *Comput. Secur.*, 2014, **45**, pp. 242–254

[35] Choi, J., Choi, C., Ko, B., *et al.*: 'A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment', *Soft Comput.*, 2014, **18**, (9), pp. 1697–1703

[36] Chwalinski, P., Belavkin, R., Cheng, X.: 'Detection of http-get attack with clustering and information theoretic measurements'. Foundations and Practice of Security, 2013, pp. 45–61

[37] Zhou, W., Jia, W., Wen, S., *et al.*: 'Detection and defense of application-layer DDoS attacks in backbone web traffic', *Future Gener. Comput. Syst.*, 2014, **38**, pp. 36–46

[38] Frey, B.J., Dueck, D.: 'Clustering by passing messages between data points', *Science*, 2007, **315**, (5814), pp. 972–976

[39] Wang, W., Guyet, T., Quiniou, R., *et al.*: 'Autonomic intrusion detection: adaptively detecting anomalies over unlabeled audit data streams in computer networks', *Knowl.-Based Syst.*, 2014, **70**, pp. 103–117

[40] Jazi, H.H., Gonzalez, H., Stakhanova, N., *et al.*: 'Detecting http-based application layer DoS attacks on web servers in the presence of sampling', *Comput. Netw.*, 2017, **121**, pp. 25–36

[41] Prasad, K.M., Reddy, A.R.M., Rao, K.V.: 'BARTD: bio-inspired anomaly based real time detection of under rated App-DDoS attack on web', *J. King Saud Univ.-Comput. Inf. Sci.*, 2017

[42] Agrawal, N., Tapaswi, S.: 'Low rate cloud DDoS attack defense method based on power spectral density analysis', *Inf. Process. Lett.*, 2018

[43] Vidal, J.M., Orozco, A.L.S., Villalba, L.J.G.: 'Adaptive artificial immune networks for mitigating DoS flooding attacks', *Swarm. Evol. Comput.*, 2018, **38**, pp. 94–108

[44] Wang, W., He, Y., Liu, J., *et al.*: 'Constructing important features from massive network traffic for lightweight intrusion detection', *IET Inf. Sec.*, 2015, **9**, (6), pp. 374–379

[45] Saaty, T.L.: '*Fundamentals of decision making and priority theory with the analytic hierarchy process*' (RWS Publications, Pittsburgh, 2000), vol. **6**, pp. 1–475

[46] Vaidya, O.S., Kumar, S.: 'Analytic hierarchy process: an overview of applications', *Eur. J. Oper. Res.*, 2006, **169**, (1), pp. 1–29

[47] Sastry, C.S., Rawat, S., Pujari, A.K., *et al.*: 'Network traffic analysis using singular value decomposition and multiscale transforms', *Inf. Sci.*, 2007, **177**, (23), pp. 5275–5291

[48] Maggi, F., Robertson, W., Kruegel, C., *et al.*: 'Protecting a moving target: addressing web application concept drift', in Kirda, E., Jha, S., Balzarotti, D. (eds.): '*Recent Advances in Intrusion Detection*, 2009, pp. 21–40
[49] Chwalinski, P., Belavkin, R., Cheng, X.: 'Detection of HTTP-GET Attack with Clustering and Information Theoretic Measurements', in Garcia-Alfaro, J., Cuppens, F., Cuppens-Boulahia, N., *et al.* (eds.): *Foundations and Practice of Security*. FPS 2012. Lecture Notes in Computer Science, vol 7743 (Springer, Berlin, Heidelberg, 2013), pp. 45–61
[50] Openstack.: 'Openstack guide'. Available at https://docs.openstack.org/liberty/install-guide-ubuntu/ (accessed 10 January 2016), 2016
[51] Sree, T.R., Bhanu, S.M.S.: 'Detection of http flooding attacks in cloud using dynamic entropy method', *Arab. J. Sci. Eng.*, 2017, pp. 1–20
[52] IRC Bot.: 'IRC Bot'. Available at https://github.com/paulbarbu/IRC-Bot, (attack tool), (accessed 10 February 2016), 2016
[53] Dirtjumper.: 'Dirt jumper-kerbs on security'. Available at: https://Krebsonsecurity.com/tag/dirt-jumper/. (attack tool), (accessed 10 February 2016), 2015
[54] HULK.: 'HULK attack'. Available at http://github.com/grafov/hulk, (attack tool), (accessed 20 February 2017), 2015
[55] HOIC.: 'HOIC attack tool'. Available at http://www.thehackersnews.com/2012/03/another-ddos-tool-from-anonymous-hoic.html', (tool), 2015
[56] Slowloris.: 'Slowloris attack tool'. Available at https://sourceforge.net/projects/slowlorisgui/, (noted), (accessed 10 February 2016), 2016
[57] Liao, Y., Vemuri, V.R.: 'Using text categorization techniques for intrusion detection'. USENIX Security Symp., 2002, vol. 12, pp. 51–59