Nabbing absconding botmasters in multi cloud environment using robustive network traffic analyzer based on super intend ensemble-learning mechanism

Gautam M Borkar[1]

[1]Affiliation not available

May 5, 2020

Abstract

Even though multi cloud has been seen as a perfect path to prevent loss of data and avoid the vendor lock in problem, it still lacks in having comprehensive security mechanism due to its inherent nature. This inherent feature enables botnet attack where group of compromised nodes would do abnormal activities which are assigned by the bot master to degrade the quality of service in the environment of cloud. Thus, to recognize and remove the bot master, this work has designed a robust analyzer by proposing agglomerative-divisive based web usage mining, which classifies different types of attributes such as Access time, Destination IP address, port number, types of protocol used. Subsequently, clustered data are fed to the web structural mining based on WAP (Web Access Pattern-tree) which groups the network traffic information based on their topology. The preprocessed network traffic information would be subjected to the robust key identifier which decrypts the network traffic. Finally in order to nab the bot master, the decoded network traffic information would be given to the ensemble learner based on random forest algorithms.

1. INTRODUCTION

Cloud computing technology is widely used by providers and consumers to store and access data via user interfaces like, web browsers. The advantages of cloud computing includes, significant cost reductions for the services provided, minimizecost of cloud and maintenance, provided accessibility worldwide and produced flexible and highly automated processes. These cloud computing benefits provide customers with least concerns about the software up gradations.

The cloud environment's main services are Infrastructure as a Service (IaaS), Software as a Service (SaaS) and Platform as a Service (PaaS). In SaaS, the cloud providers host the complete application into the cloud and enable users to access them via internet.

Figure 1

The SaaS is either built directly onto the cloud infrastructure or on top of the facilities of IaaS and PaaS. Payment is based on pay-per-use grounds for SaaS apps. In IaaS, clients are supplied with computing resources, storage and network equipment as an internet-based service. In PaaS, consumers create, deploy and oversee the service providers ' instruments, frameworks, platforms and other company products.

Figure 2

Botnet is one of the major threats to cloud computing. It is a combination of two words, namely Bot and Net Robot is the word ' Bot ' and the network is the term ' Net. Botnet involves a network of computer systems that are infected and controlled by a human bot master or a bot herder.[1].with the assistance of C&C servers, the botmasters regulates the infected equipment remotely. Once a node is infected with malicious scripts or codes, the infected node joins the botnet as directed in the code and operates without user understanding for the botmasters. The Botnet also proliferates and infects much equipment. The attackers are carrying out numerous criminal operations such as DDoS, clicking fraud, phishing, spamming, sniffing traffic, and spreading botnet malware. The botnets are categorized as IRC, P2P botnet, HTTP botnet and hybrid botnet based on the C&C channel [2].

Thepresent botnet detection approaches use two primary groups to define the assaults triggered by the botnet, namely the honeynet-based detection method and the intrusion detection system[3]. Upon detection of the botnet, threat effect reduction methods such as proactive defense methods and reactive defense techniques [4] are used to heal botnet-caused infections. In order to safeguard the cloud network, it is essential to identify the botnets at their original phases and stop the assaults that they cause.

The botnet's conduct in network systems and the evaluation of their conduct helps predict their nature of network assault. Monitoring this data leads to a psychological distinction of two forms of active and passive analysis. The active assessment detects and deactivates the possible malware [5].

Honeypot and Honeynets are popular active techniques of assessment used to identify botnet. Honeypots are vulnerabilities that are deliberately implemented to detect intrusions and assaults in the cloud setting [6]. The honeypot is categorized into two kinds based on the emulation ability: high-interaction and low-interaction. In the high interaction honeypot, nearly all features of the actualworkingscheme are simulated. Only the recognized ports and protocols are answered [7]. The small honeypot interaction simulates only the actual operating system's main characteristics. High honeypot interaction enables the attackers to achieve complete control of the operating system, while low honeypot interaction restricts attackers from gaining complete control of the operating system as characteristics are restricted[8]. Different sizes of honeypots are merged into a honeynet in the cloud network. Honeynet's critical elements include, restricted scalability, inability to detect internet assaults, the ability to monitor only malicious operations when interacting with them, difficulties in the discovery of infected devices, and Systems that are impacted by honeybots are not monitored. In addition, only the expected infected machines are recorded [9].

The honeywall is the entry and departure point for all the netcirculation of the honeypot. The honeywall controls and analyzes all the network traffic that arrives and leaves the honeynet or honeypot system. The network traffic gathered assists the honeywall in their process of assessment. The C&C server IP address, port address, authentication password, connecting botnet channel name is determined by the honeywall's information capture capacity [10].Passive evaluation controls the botnet-created traffic without the message being corrupted or changed. It analyzes primarily the secondary impacts of botnet traffic. The advantage of the passive method of analysis is that detection activity cannot be perceived by the botmasters. The techniques of passive assessment are very difficult to deploy in the cloud setting [11]. Thus to overcome all the above mentioned issue, we proposed an robustive network traffic analyzer to efficiently analyze and identify botmaster who influence the botnet attack in cloud network.

Thus, the remaining of the paper is prepared as follows: section 2 describe about the papers deals with the security and attacks in cloud environment; our contribution over the paper i.e., proposed work has deliberate in section 3; section 4 follows the result and the output of our proposed work; finally, the overall work is concluded in the section 5.

2. Literature survey

[12] HammiBadis, Guillaume Doyen and RidaKhatoun had presented a Collaborative Approach for a Source based Detection of Botclouds. Each tenant is treated separately and for each one a tree is constructed. The objective of using a tree-based hierarchical architecture is to reduce the complexity of the time and cycles of execution of the detection algorithm. Indeed, the proposed approach is based on the idea of (1) applying the PCA-based detection algorithm, small sets of VMs after their data has been standardized, and (2) aggregating the result set to decide on the state of the tenant (attacker or not). However, it doesn't prevent against attacks in CSPs by cloud infrastructure.

[13] RémiCogranne, Guillaume Doyen, NisrineGhadban and BadisHammi had submitted a Decentralized and Robust Detection Method for Multi-Tenant Virtualized Environments This technique focused on developing and validating a solution to detect malicious operations by virtual hosts infected with botnets in the public cloud context. Although, this method doesn't prevent from data leakage problem thus leads to major security issues in cloud.

[14] GauravSomani, Manoj Singh Gaur Suggested a fresh "Inside-out Scale" approach that reduces the "Resource Utilization Factor" to a minimum value for fast assault absorption. The proposed strategy sacrifices victim service resources in relation to other co-located facilities and provides mitigation facilities with those resources to determine their accessibility during the assault. Eventhough, this method does not deter data collection assaults in the cloud processing.

[15]WesamBhaya, Mehdi EbadyManaa A combination of unsupervised data mining techniques was introduced with the introduction of an intrusion detection system. The entropy idea in terms of windowing incoming packets is implemented using information mining method using Clustering Using Representative (CURE) as a cluster analysis to detect the network flow DDoS attack. However, only the information theory was regarded to transform nominal data to numerical data using entropy windows is considered. It doesn't prevent from the attacks in frequency of packets during network flow.

[16]AmjadAlsirhani, SrinivasSampalli and Peter Bodorik the DDoS detection scheme provided those advantages from the resources of cloud computing. It entails of three notions: algorithms of classification, similarcalculating, besides a system of fuzzy logic. The notion of parallelism is used to effectively speed up the implementation of the classification algorithms used. It assessed the classification algorithm and the DDoS detection parallel processing by configuring a test-bed that consists of one master and three slaves. Although, this method only detect the attacks not prevent the system from the detection.

[17] Salman Iqbal et al. In terms of their cloud models, cloud-based attacks and vulnerabilities are collected and classified. We also present taxonomy of cloud security assaults and prospective mitigation strategies in order to provide a thorough knowledge of safety needs in the cloud setting. However, it fails to ensure integrity and confidentiality.

[18] Bo Li et al. New cluster-based intrusion detection system to track network traffic in the cloud environment to support intra-VM networks traffic monitor. A traffic deduction framework is also designed to eliminate redundant network traffic and reduce the burden of IDS clusters. It does not involve physical switch support and uses intra-VM network traffic to control virtual network traffic instead of virtualized IDS. Although in real cloud environments, this system is not efficiently suited.

[19] Mohamed Idhammad et al. had presented a distributed intrusion detection system for cloud settings based on machine learning. This system had been intended to be inserted side by side in the cloud with the cloud provider's edge network parts. It is used to intercept incoming network traffic to the physical layer's edge network routers. Even though, it is not well suitable for real world's IDS deployment.

Thus from the above discussion, it is revealed that, in [12] doesn't prevent against attacks in CSPs, [13] doesn't prevent from data leakage problem, [14] doesn't prevent from the attacks in frequency of packets, [15] does not deter data collection assaults, [16] only detect the attacks not prevent the system from the detection, [17] fails to ensure integrity and confidentiality, [18] not efficiently suitable in real cloud environments, [19] not well suitable for real world's IDS deployment. Hence in order to tackle such issues, there is a need to innovate a new strategy in the field of cloud environment.

3 Threat analysis in cloud computing

Cloud provides an ideal environment for botnets. First, cloud has rich and elastic computing resources, such as storage, bandwidth and processors, which are easy to access. Attackers can compromise a cloud-based server as their C&C server or lease high performance virtual machines with stolen credit card information. Second, hackers can easily deploy C&C servers and stepping-stones in cloud, dynamically change them to new domains, and delete their virtual images to remove evidence of their activities. Third party VPNs, proxies and SSH tunneling are commonly used stepping-stones, across which bot masters access C&C server and hide their identities. Finally, clouds are convenient for bot masters to select multiple cloud service providers in multiple domains and countries, and that makes the trace back very difficult.

Figure 3

Bot master establishes a botnet by deploying bots and launches attacks from the original host to victims through a number of stepping-stones and C&C server. Bot binaries are executed at victim machines, which become zombies and join the botnet. Bot master attempts to communicate with the victims for stealing sensitive information. It is different from another type of botnet attacks, such as DoS/DDoS attacks where an attacker has no interest in any communications with victims. Under the supervision of bot master through C&C server, victims periodically send data to receivers (i.e. drop-zones). The collected data are then sent back to bot master through stepping-stones.

A major difficulty for analyzing botnet attack traffic is that communication between bots and C&C servers are usually encrypted, and the encryption keys are to be identified first and also attackers usually hide behind stepping-stones from Web proxy, VPN and SSH tunneling.

For example, Zeus bot uses RC4 encryption scheme in which the string of key is transformed into a 256-byte S array. It is the S array that resides in memory to encrypt plaintexts and decrypt cipher texts. Therefore the key identification scheme looks for an S array in the memory image. RC4 utilizes important strings of varying length to generate a permutation of 256 bytes with different values, denoted as S arrays array constitutes the RC4 key for information traffic encryption and decryption.

Therefore in order to nab the bot master, systematic and reinforce analyzing of network traffic within the multi cloud environment is required. Therefore this work is mainly focused on designing a systematic network traffic analyzer which would find the exact origin of destruction and remove from the cloud.

3.1 Identifying Bot Masters Based On Ensemble Learning Classifiers

The proposed robust analyzer based on network forensic involves in processing vast amount of data that is being collected, stored and analyzed for ascertaining; how an attack was carried out or how an event occurred in a network and also for ensuring the overall integrity. Since network traffic information are volatile and dynamic, this work has proposed aagglomerative-divisivebased web usage miningwhichclassifies different types of attributes such as Access time, Destination IP address, port number, types of protocol used, inter arrival time, frequently requested data, packet length, number of request, Sender MAC address, Destination MAC address, Sending time in network traffic based on spatial and temporary data to prevent missing helpful data from unlabeled samples without requiring previous specification of the amount of clusters. Subsequently, clustered data are fed to theweb structural mining based on WAP (Web Access Pattern-tree)which groups the network traffic information based on their topology in which connected to the botmaster are compromised nodes who would change the topology once the task is completed by the compromised nodes.The preprocessed network traffic information would be subjected to the robust key identifierwhichdecrypts the network traffic.

Finally in order to nab the botmaster the decoded network traffic information would be given to theensemble learner based on random forest algorithms which has the capacity to explore assaults by tracing the attack back to the source and finding the nature of the attacker if he or she is a individual, host or network, as well as predicting future assaults with elevated precision by correlating attack patterns with prior traffic data records.

Figure 4

3.1.1 Algorithmic steps for Agglomerative Hierarchical clustering

Let $X = \{x_1, x_2, x_3, \ldots\ldots, x_n\}$ be the set of data points.

1) Begin with the disjoint clustering having level $L(0) = 0$ and sequence number $m = 0$.

2) Find the least distance pair of clusters in the current clustering, say pair $(r),\ (s)$, according to$d[(r),(s)] = min\ d[(i),(j)]$ where the minimum is over all pairs of clusters in the current clustering.

3) Increment the sequence number: $m = m + 1$.Merge clusters (r) and (s) into a single cluster to form the next clustering m. Set the level of this clustering to $L(m) = d[(r),(s)]$.

4) Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The distance between the new clusters, denoted (r,s) and $old\ cluster(k)$ is defined in this way:$d[(k),\ (r,s)] = min\ (d[(k),(r)],\ d[(k),(s)])$.

5) If all the data points are in one cluster then stop, else repeat from step 2).

If points (objects) i and j are agglomerated into cluster$i \cup j$, then we must simply specify the new dissimilarity between the cluster and all other points (objects or clusters). The formula is:

$$d\,(i \cup j,k) = \alpha_i d\,(i,k) + \alpha_j d\,(j,k) + \beta d\,(i,j) + \gamma|d\,(i,k) - d\,(j,k)\,|$$

Where $\alpha_i,\alpha_j,\beta\ and\ \gamma$ define the agglomerative criterion. In the case of the complete link method, using$\alpha_i = \alpha_j = \gamma = \frac{1}{2},\ \ and\ \beta = 0$ given us

$d\,(i \cup j,k) = \frac{1}{2}d\,(i,k) + \frac{1}{2}d\,(j,k) + \frac{1}{2}\,|d\,(i,k) - d\,(j,k)|$ (1)

3.1.2 Construction of WAP-Tree and Mining of WAP-Tree

Only frequent 1-sequences are considered for constructing WAP-tree, as they only are useful in generating k-sequences where k>1. Common prefixes are shared in the WAP-tree to save space. WAP-tree records, labels and counts two parts of data. The tree's root is a virtual special node with an empty label and count 0.

The construction of WAP-tree is done as follows: for each series of accesses, frequent subsequence is entered into the tree, starting from the root node. While inserting the first event e, if the current node has already a child e, then Increase the child node count with event e by 1, otherwise a new child with label e and count 1 will be created. Then insert the remaining subsequence into the sub-tree rooted in the present e node recursively. All nodes with same label e are linked by shared label linkages into a queue, called event-queue. Event queue for$e_i$is called e_i-queue. Head of each event-queue is registered in a header table H.

A simple web access sequence database obtained after preprocessing, with the set of access events E = {a, b, c, d, e, f} is shown in Table 1 and WAP-tree with linkage header for the WASD is given in Fig. 5.

Table 1

Figure 5 .

All pattern information related to a frequent event $e_i$can be accessed by following all the branches in WAP-tree linked by$e$ $_{\text{i}}$queue only once. All nodes from the root of the tree to the e i(excluded) node form a prefix series of e iand this node count is the prefix sequence count. Let G and H be two prefix sequences of$e_i$and G is also formed by the sub-path from root that H is formed by, H is called a super- prefix sequence of G, and G is a sub-prefix sequence of H. The un-subsumed count is the count for sequence ofe_i prefixes without any super-prefix sequences. For a prefix sequence of $e_i$with some super-prefix sequences, the un-subsumed count is the count of that sequence minus un-subsumed counts of all its super-prefix sequences. Access patterns with same suffix are now used for searching all web access patterns.

Mining is done as follows: For each event $e_i$in the header list, conditional sequence base is found. A suffix event's conditional sequence list is obtained by following the event's header connection and reading the root route to each node (excluding the node). The prefix sequence count is identical to the node count. For each prefix sequence inserted into the conditional sequence base with count c, all its sub-prefix sequences are inserted with –c as count, to get the un-subsumed count. Then find the set of conditional frequency occurrences. If it is not empty, recursive mining is done on the conditional WAP-tree of each frequent event. When there is only one branch in the conditional WAP-tree, all unique combinations of nodes in that branch are generated as patterns.

3.1.3 Entropy Background

Let w be a word composed of N characters over alphabet Σ = (a0, a1…am-1). For each ai[?]Σ, we can count its occurrences, denoted as ni, and then the frequency fiof aiis calculated as ni/N. The entropy of w is estimated as:

Where MLE is the maximum likelihood estimator. When the characters are bytes, the maximum entropy is 8 (bits per byte). The above estimator is a little different from the original definition of entropy, which is:

However, the approximation$H_N^{\mathrm{MLE}}(w) \sim H(p)$is validwhen N >> m. In practice, the condition N >> m couldbe an issue because some packets are short and less than m.[9] introduces an approach to solve this problem. Firstly, theauthors estimate the average sample entropy HN(p), which isdefined as average of $H_N^{\mathrm{MLE}}(w)$over all words w of lengthN. When p is uniform distribution µ, HN(µ) can be calculatedas:

With c = N m and o (1) is less than 0.004. Similar to [9], we use the Monte-Carlo method to estimate the standard deviation SD $(^{HMLEN}).Finally,weuseHN(m)-4*SD(^{HMLEN)astheentropythresholdtodecidewhetherthedataunderinvestigationishighorlowentropy.Notethattheabovemetho}$

B. HE Flow Detection

We now present the flow- and packet-based algorithms to designate a flow as HE. We will then compare these algorithms by applying them on data of known, popular types. While they encrypt user data, many encryption protocols exchange packets at the beginning of a flow that are not encrypted (and thus low entropy). After the initial exchange,

Figure 6

Figure 7

Subsequent packets are encrypted. Figure 6 illustrates this scenario. To prevent such LE packets from skewing entropy calculation our algorithms wait until N Sequential High EntropyPackets have been detected before calculating entropy. Unfortunately, there is no clear way to estimate N, so we determine the value of N experimentally. For our datasets N = 2 seems to work best.

We now describe briefly the flow-based and packet-based algorithms. Recall that both algorithms aim at labeling a flow as HE or LE, but the former does so by examining the entire flow data, where the latter examines each packet separately.

1) Flow-based Entropy:

After detecting N Sequential High Entropy Packets we capture the payload of all subsequent packets and then calculate the cumulative entropy of the resulting data (including the initial HE packets). We then compare the cumulative entropy with the threshold, as described earlier. If the cumulative entropy is greater than the threshold, then the flow is identified as HE, else it is LE.

2) Packet-based Entropy:

After detecting N Sequential high entropy packets, we calculate the entropy for each packet and classify it as HE or LE. At the end of the flow we count the number of HE and LE packets, denoted as N (HE) and N (LE). If N (HE)/(N(HE)+N(LE)) is greater than our threshold, which is named High Entropy Packet Percentage Threshold, then we consider the flow as HE. Figure 7 illustrates this approach.

Table 2

2) Test 2: Encrypted Traffic: In this test we repeat the above methodology, but this time using real network traffic, specifically the trace Lab1. We isolate traffic on ports 22 (SSH) and 443 (HTTPS). The results are

shown in Table 3. As we can see from the table the two algorithms identify 95% and 97% of the flows in the trace as HE, a very good result.

Table 3

By using Agglomerative Hierarchical clustering algorithm, the data are clustered into a group to avoid missing of useful information as well as WAP-tree methodology is used to linearize the data in a tree form. However, there is a need for get an accurate prediction for finding the botmaster.

3.1.4 Ensemble model

Here linearized data which is derived from the WAP tree would be subjected to the training phase based on random Poisson forest. The Random forest model is made up of large set of decision trees and combined them to get an accurate prediction. In decision tree each internal node indicates a test on an attribute. In a decision tree, each branch shows the result of the test. If the node does not have any children then that node is called a leaf node. Every leaf node in the decision tree shows a class label.

The main contibution of this model is that Rather than hunting down the best feature while part a hub, it scans for the best feature among an irregular subset of features. This procedure makes a wide decent variety, which for the most part brings about a superior model.

The random forest algorithm takes less time to train but more time to predict since enormous number of decision trees would cause the model to slow down. In order to speed up the entire process of random forest model, Poisson distribution function is adapted.

Bagging

To reduce its variance

Suppose is a classifier, such as a tree, generating a predicted class label at point x on the basis of our training data S. We take bootstrap samples to bag Cwe bring samples of bootstrap to bag C. Then

Bagging can dramatically decrease the variance in volatile (like trees) processes, leading to better forecast. In C (e.g. a tree), however, any easy structure is lost.

Boosting

Average many trees, each grown to re-weighted versions of the training data.

Final Classifier is weighted by calculating average of classifiers:

Figure 8

AdaBoost

1. Initialize the observation weights

2. For m = 1 to M repeat steps :

(a) Fit a classifier to the training data using weights.

(b) Compute weighted error of newest tree

(c) Compute

(d) Update weights for

And renormalize to to sum to 1.

3. Output

Therefore, our proposed work is efficient in analyzing the network traffic as well as in recognizing and removing the bot net attack which is influenced by a botmaster .

4. Result analysis

In order to evaluate the implementation of the proposed scheme, following performance measures are taken into account they are packet delivery ratio, packet lost ratio, throughput and two attacks are analyzed they are distributed denial of service and spam attack.

4.1. Performance analysis

4.1.1Packet delivery ratio

The estimation of Packet Delivery Ratio (PDR) depends on the received and created bundles as recorded in the trace document. All in all, PDR is characterized as the proportion between the number of packets sent by the source node and the number of packets received by the destination node.

Figure 9

4.1.2 Packet lost ratio

Packet loss happens when at least one packets of information traversing a computer network neglect to achieve their goal. Packet loss is estimated as a level of packets lost concerning packets sent. The below figure depicted the packet lost ratio of IoT based network during normal time and attack time.

Figure 10

4.1.3 Throughput

In data transmission, network throughput is the amount of data transferred successfully in a specified time period from source node to destination node and typically measured in bits per second (bps), as in megabits per second (Mbps) or gigabits per second (Gbps).

Figure 11

Table 4

4.2 Clustering of botnet of distributed denial of service attack

In this type of botnet attack, group of attackers would send the request for resource to the same destination address for specified time continuously so an authenticated user cannot get that resource for a particular time. The proposed algorithm would cluster those nodes based on the similarity vale of packet sending time, destination address and the resource which they requested continuously and the distance between source nodes and destination node is calculated in order to efficiently group the attacks.

In existing systems, hierarchical based clustering has been incorporated to cluster the devices of the attackers in the IoT based network, the main problem with hierarchical based clustering is thatif the decision is taken once to join two clusters, it cannot be cancelled but in this work mixture model is used for clustering so it has both matrices distance as well as similarity based so the clustering ratio is high when compared with an existing techniques.

Table 5

Figure 12

4.3 Clustering of botnet of spam attack

Here the botnet would send the email to the spam box instead of sending to the inbox of the mail application. It includes sending undesirable messages, regularly spontaneous publicizing, to countless. Spam is a genuine security worry as it can be utilized to convey Trojan stallions, infections, worms, spyware, and focused on phishing attacks. The proposed system would cluster this type of botnet based on the behavior that sending file to spam box instead of sending to the inbox of the mail. Existing techniquesdid not cope with different sized cluster and irregular shapes and need of breaking large clusters since they are based on hierarchical based clustering. In this work mixture based clustering is incorporated so it manages all shapes of clustering.

Table 6

Figure 13

Table 7

Figure 14

4.4 Comparison Of Proposed System With Existing Techniques

In this section, the proposed system is compared with existing classifiers like decision tree, Random forest, RBF. In order to evaluate the proposed system following parameters are considered Precision, Recall, F-measure and Accuracy.

4.4.1 Precision

Precision is a degree of what fraction of test data is detected as attack are actually from the attack classes.

Where TP represents the true positive value, FP indicates the false positive.

4.4.2 Recall

Recall measures the fraction of attack class that was correctly detected

Where TP indicates the true positive value and FN indicates the false negative

4.4.3 F-Measure

F-measure is a degree of a measure of test's accuracy, which measures the balance between precision and recall.

Where P represents the precision and R denotes the Recall value

4.4.4 Accuracy

Accuracy is defined as the proportion of number of correctly classified botnet attacks to the total number of botnet attacks

Where Ic Bindicates the correctly identified botnet attack, TB denotes the total number of botnet attack.

Figure 15

Fig 15 describes about the comparison results of proposed vs. prior methodologies such as: ABE, IBE and PBE. Here for the proposed work, the obtained encryption time is 0.18 sec, whereas for ABE, IBE and PBE the obtained encryption time will be 0.24sec, 0.5sec and 0.43 respectively.

Figure 16

Fig 16 describes about the comparison results of proposed vs. prior methodologies such as: ABE, IBE and PBE. Here for the proposed work, the obtained decryption time is 0.18 sec, whereas for ABE, IBE and PBE the obtained decryption time will be 0.25sec, 0.4sec and 0.38sec respectively.

Figure 17

Fig 17 describes about the comparison results of proposed vs. prior methodologies such as: GA, PSO and ABC. Here for the proposed work, the obtained sensitivity is 0.2, whereas for GA, PSO and ABC, the obtained sensitivity will be 0.6, 0.7 and 0.68 respectively.

Figure 18

Fig 18 describes about the comparison results of proposed vs. prior methodologies such as: GA, PSO and ABC. Here for the proposed work, the obtained specificity is 0.95, whereas for GA, PSO and ABC, the obtained specificity is 0.0.985, 0.998 and 0.99 respectively.

Figure 19

Fig 19 describes about the comparison results of proposed vs. prior methodologies such as: GA, PSO and ABC. Here for the proposed work, the obtained FDR is 0.03, whereas for GA, PSO and ABC, the obtained FDR is 0.35, 0.2 and 0.3 respectively.

Figure 20

Fig 20 describes about the comparison results of proposed vs. prior methodologies such as: GA, PSO and ABC. Here for the proposed work, the obtained accuracy is 0.87 sec, whereas for GA, PSO and ABC, the obtained accuracy is 0.97, 0.98 and 0.98 respectively.

Figure 21

Fig 21 describes about the comparison results of proposed vs. prior methodologies such as: Decision tree, Random forest and RBF. Here for the proposed work, the obtained precision ratio is 0.968, whereas for Decision tree, Random forest and RBF the obtained precision ratio is 0.968, 0.976 and 0.966 respectively.

Figure 22

Fig 22 describes about the comparison results of proposed vs. prior methodologies such as: Decision tree, Random forest and RBF. Here for the proposed work, the obtained precision is 0.93, whereas for Decision tree, Random forest and RBF the obtained precision ratio is 0.933, 0.924 and 0.987 respectively.

Figure 23

Fig 23 describes about the comparison results of proposed vs. prior methodologies such as: Decision tree, Random forest and RBF. Here for the proposed work, the obtained F-measure ratio is 0.949, whereas for Decision tree, Random forest and RBF the obtained F-measure ratio will be 0.95, 0.95 and 0.976 respectively.

Figure 24

Fig 24 describes about the comparison results of proposed vs. prior methodologies such as: Decision tree, Random forest and RBF. Here for the proposed work, the obtained accuracy is 96.5, whereas for Decision tree, Random forest and RBF the obtained accuracy will be 96.51, 96.5 and 99 respectively.

Figure 25

Fig 25 describes about the comparison results of proposed vs. prior methodologies such as: SVM, SVR, ANN and RNR. Here for the proposed work, the obtained accuracy is 0.99 sec, whereas for ABE, IBE and PB SVM, SVR, ANN and RNR the obtained accuracy will be 0.95, 0.96, 0.97 and 0.95 respectively.

Figure 26

Fig 26describes about the comparison results of proposed vs. prior methodologies such as: SVM, SVR, ANN and RNR. Here for the proposed work, the obtained overall workload estimation is 0.049, whereas for ABE, IBE and PB SVM, SVR, ANN and RNR the obtained overall workload estimation is 0.087, 0.078, 0.053 and 0.062 respectively.

Figure 27

Fig 27 describes about the comparison results of proposed vs. prior methodologies such as: SVM, SVR, ANN and RNR. Here for the proposed work, the obtained FAR is 0.03, whereas for ABE, IBE and PB SVM, SVR, ANN and RNR the obtained FAR will be 0.059, 0.049, 0.038 and 0.069 respectively.

Figure 28

Fig 28 describes about the comparison results of proposed vs. prior methodologies such as: SVM, SVR, ANN and RNR. Here for the proposed work, the obtained infection rate is 0.012, whereas for ABE, IBE and PB SVM, SVR, ANN and RNR the obtained infection rates are 0.025, 0.028, 0.035 and 0.025 respectively.

Figure 29

Fig 29 describes about the comparison results of proposed vs. prior methodologies such as: SVM, SVR, ANN and RNR. Here for the proposed work, the obtained number of iteration is 238, whereas for ABE, IBE and PB SVM, SVR, ANN and RNR the obtained number of iterations are 360, 350, 320 and 250 respectively.

Figure 30

Fig 30 describes about the comparison results of proposed vs. prior methodologies such as: SVM, SVR, ANN and RNR. Here for the proposed work, the obtained efficiency is 0.99 sec, whereas for ABE, IBE and PB SVM, SVR, ANN and RNR the obtained efficiency is 1.03, 0.98, 1.03 and 1.07 respectively.

Thus, from above results it has clearly shown that our proposed system has efficiently detect the botmaster by whom botnet attack is influenced in the cloud environment as well as , the result obtained for the proposed system has exposed better performance when compared to existing systems.

5. CONCLUSION

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. Therefore, there is a lot of chances for security attacks especially botnet attack to degrade the quality of cloud service influenced by botmaster. Thus the proposed robustive network traffic analyzer based on superintend ensemble-learning mechanism, has clustered the each type of botnet attack such distributed denial of service, spam botnet attack and maintain the reliability and quality of service in cloud based applications. Thus, the result obtained for the proposed system has exposed better performance when compared to existing systems. The proposed system has taken optimal precision value of0.961 and recall value of 0.986. It accomplished high F-measure value of 0.976 and high detection accuracy value of 99.04.

References

[1]Anselmi D, Boscovich R, etal. Security intelligence report, Security in the cloud. *Communications of the ACM* , 2010;53(11):16–18.

[2] Armbrust M, Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, et al. A view of cloud computing. *Communications of the ACM,* 2010;53(4):50–58.

[3] Chandrashekar J. The Dark Cloud Understanding and Defending Against Botnets and Stealthy Malware. *IntelRTechnology Journal,*2009;13(2).

[4] Clayton R. Stopping spam by extrusion detection. *In First Conference on Email and Anti-Spam* , 2004.

[5] Dagon D, Gu G, Lee C, and Lee W. A taxonomy of botnet structures . *In acsac, IEEE Computer Society,* 2007;325–339.

[6] Haddadi H. Fighting online click-fraud using bluff ads.*ACM SIGCOMM Computer Communication Review,* 2010;40(2):21–25.

[7] Ianelli V, and Hackworth A. Botnets as a vehicle for online crime. *CERT Coordination Center* , 2005;1–28.

[8] Jing L, Yang, X., Kaveh G, Hongmei D, and Jingyuan Z. Botnet: Classification, attacks, detection, tracing, and preventive measures.*EURASIP journal on wireless communications and networking* , 2009.

[9] Kshetri N . The economics of click fraud . *IEEE Security and Privacy* , 2010;45–53.

[10] Maggi F, and Zanero S. Rethinking security in a cloudy world. *Politecnico di Milano, Tech. Rep. TR-2010-11* , 2010.

[11] Mirkovic J, and Reiher P. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review* , 2004; 34(2):39– 53.

[12] Badis, Hammi, Guillaume Doyen, and RidaKhatoun. A collaborative approach for a source based detection of botclouds. *In Integrated Network Management (IM), IFIP/IEEE International Symposium on,* 2015; 906-909.

[13] Cogranne, Rémi, Guillaume Doyen, NisrineGhadban, and BadisHammi. Detecting Botclouds at Large Scale: A Decentralized and Robust Detection Method for Multi-Tenant Virtualized Environments. *IEEE Transactions on Network and Service Management,* 2018;15(1):68-82.

[14] Somani, Gaurav, Manoj Singh Gaur, DheerajSanghi, Mauro Conti, and MuttukrishnanRajarajan. Scale Inside-out: Rapid Mitigation of Cloud DDoS Attacks. *IEEE Transactions on Dependable and Secure Computing* , 2018;15(6):959-973.

[15] Bhaya, Wesam, and Mehdi EbadyManaa. DDoS attack detection approach using an efficient cluster analysis in large data scale.*In New Trends in Information & Communications Technology Applications (NTICT), Annual Conference on,* 2017;168-173.

[16] Alsirhani, Amjad, SrinivasSampalli, and Peter Bodorik. DDoS Attack Detection System: Utilizing Classification Algorithms with Apache Spark. *In New Technologies, Mobility and Security (NTMS), 2018 9th IFIP International Conference on* , 2018; 1-7.

[17] Iqbal S, Kiah MLM, Dhaghighi B, Hussain M, Khan S, Khan MK, and Choo KKR. A taxonomy and intrusion detection and prevention as a service. *Journal of Network and Computer Applications,*2016;74:98-120.

[18] Li B, Liu P, and Lin L, June. A cluster-based intrusion detection framework for monitoring the traffic of cloud environments.*In 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud),* 2016; 42-45.

[19] Idhammad M, Afdel K, and Belouch M. Distributed intrusion detection system for cloud environments based on data mining techniques. *Procedia Computer Science,* 2018;127: 35-41.

Figure 1: Cloud Architecture

Figure 2 :Cloud Services

Figure 3: Structure of Botnet

Figure 4 :Flow diagram of proposed work

Figure 5 :WAP-tree with linkage (dotted line) for the frequent sub-sequences in Table1

Figure 6: Protocol Format 1

Figure 7: Protocol Format 2

Figure 8: Weighted samples for final classifier

Figure 9: shows the ratio of packet delivery during normal and attack period

Figure 10: shows the packet loss ratio of the network during normal flow and attack

Figure 11: shows that throughput of the network under normal and attack period

Figure 12: shows the clustering of botnet attack which leads to Distributed DoS attack

Figure 13: Threat analysis of the proposed system

Figure 14: shows the clustering of spam type botnet attack

Figure 15: Comparison regarding encryption time

Figure 16: Comparison regarding decryption time

Figure 17: Comparison regarding sensitivity

Figure 18: Comparison regarding specificity

Figure 19: Comparison regarding FDR

Figure 20: Comparison regarding accuracy

Figure 21: Comparison regarding precision ratio

Figure 22: Comparison regarding precision

Figure 23: Comparison regarding F-measure

Figure 24: Comparison regarding accuracy

Figure 25: Comparison regarding accuracy

Figure 26: Comparison regarding workload estimation

Figure 27: Comparison regarding FAR

Figure 28: Comparison regarding infection rates

Figure 29: Comparison regarding No. of iterations

Figure 30: Comparison regarding efficiency

Table 1: A database of web access sequences

User ID	Web access sequence	Frequent sub-sequence
100	abdac	abac
200	eaebcac	abcac
300	babfaec	babac
400	afbacfc	abacc

Table 2: offline encrypted files

Encryption Algorithm	Key Length (bit)	HE (flow-based) out of 600	HE (packet-based) out of 600
None	N/A	6	32
DESX	128	599	599
Blowfish	448	600	600
Rijndael (AES)	256	600	600
CAST	256	600	600
Triple-DES	192	600	600
RC2	1024	599	600
Diamond 2	2048	598	600
Tea	128	599	600
Safer	128	599	600
3-Way	96	600	600
GOST	256	600	600
Shark	128	599	600
Square	128	598	600
Skipjack	80	598	600
Twofish	256	598	600
MARS (IBM)	448	600	600

Encryption Algorithm	Key Length (bit)	HE (flow-based) out of 600	HE (packet-based) out of 600
Serpent	256	600	600

Table 3: Online Encrypted Traffic

Traffic	SSH	HTTPS
Total Flows	3618	1717
HE (flow-based)	3440	1638
Rate (flow-based)	95.1%	95.4%
HE (packet-based)	3517	1676
Rate (packet-based)	97.2%	97.6%

Table 4: Log details of each virtual machine in the cloud environment

Node	IP Address	Arrival time(sec)	Packet Delivery ratio	Packet Loss	Throughput
n1	151.142.255.1	2.256	88.025	2.2835	56.895
n2	151.142.255.2	1.267	93.211	1.4756	54.742
n3	151.142.255.3	8.278	94.723	1.8629	55.315
n4	151.142.255.4	1.289	94.601	1.8687	56.889
n5	151.142.255.5	1.314	94.783	1.4756	53.895
n6	151.142.255.6	5.311	89.5404	1.8629	56.888
n7	151.142.255.7	4.322	93.031	2.1905	53.895
n8	151.142.255.8	3.333	95.5216	2.1905	51.2
n9	151.142.255.9	2.344	88.0122	1.4756	60.235
n10	151.142.255.10	1.355	90.5028	2.1905	53.895
n11	151.142.255.11	1.366	92.9934	1.8629	51.2
n12	151.142.255.12	9.377	95.484	2.1905	51.2
n13	151.142.255.13	8.388	87.9746	2.2835	51.2
n14	151.142.255.14	7.399	91.4652	1.9597	51.895
n15	151.142.255.15	6.441	92.9558	1.8629	50.96
n16	151.142.255.16	5.421	89.937	1.8629	53.895
n17	151.142.255.17	4.432	90.4276	1.57717	56.889
n18	151.142.255.18	3.443	92.9182	1.9598	56.888
n19	151.142.255.19	2.454	95.4088	1.8629	51.221
n20	151.142.255.20	1.465	89.8994	1.9598	56.38

Table 5: shows that list of virtual machines clustered under DoS botnet attack

Node	Source IP Address	Packet Sending Time(sec)	destination IP Address	Spam box
N11	151.142.255.11	0.214	151.142.255.1	mail
N12	151.142.255.12	0.214	151.142.255.1	mail
N13	151.142.255.13	0.214	151.142.255.1	mail
N14	151.142.255.14	0.214	151.142.255.1	mail
N15	151.142.255.15	0.214	151.142.255.1	mail
N16	151.142.255.16	0.214	151.142.255.1	mail
N17	151.142.255.17	0.214	151.142.255.1	mail

Node	Source IP Address	Packet Sending Time(sec)	destination IP Address	Spam box
N18	151.142.255.18	0.214	151.142.255.1	mail
N19	151.142.255.19	0.214	151.142.255.1	mail
N20	151.142.255.20	0.214	151.142.255.1	mail

Table 6: Threat analysis of the proposed system

Overall workload estimation	0.04761
false-alarm rate (FAR)	0.02941
various infection rates	0.0119
efficiency	1.1904
number of iterations	234
accuracy	0.99

Table 7: shows that list of nodes clustered under botnet spam attack

Figure 1

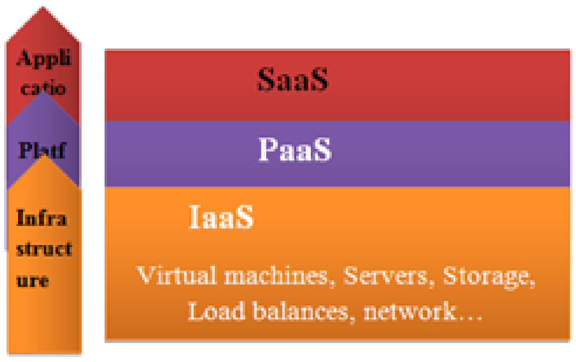

Figure 2

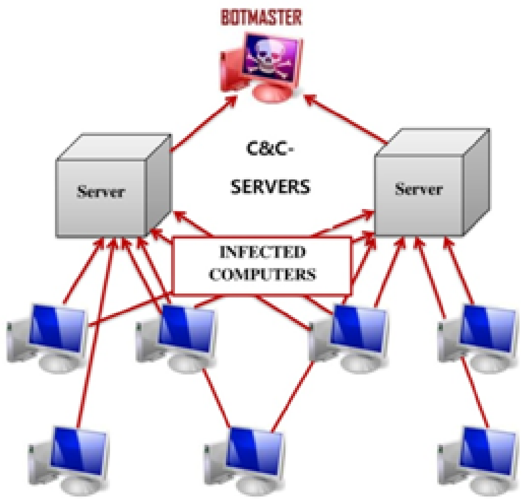

Figure 3

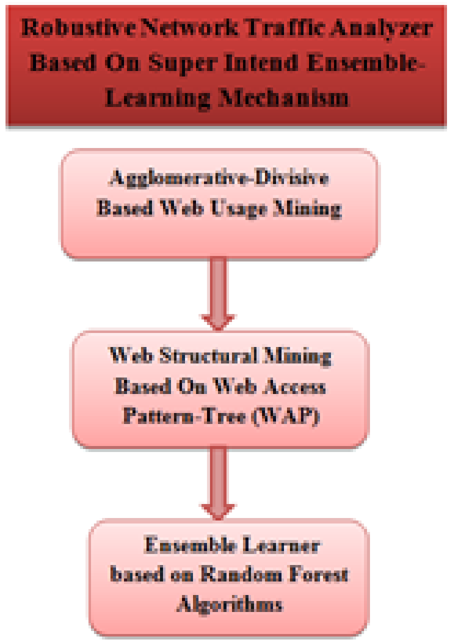
Robustive Network Traffic Analyzer Based On Super Intend Ensemble-Learning Mechanism
Agglomerative-Divisive Based Web Usage Mining
Web Structural Mining Based On Web Access Pattern-Tree (WAP)
Ensemble Learner based on Random Forest Algorithms

Figure 4

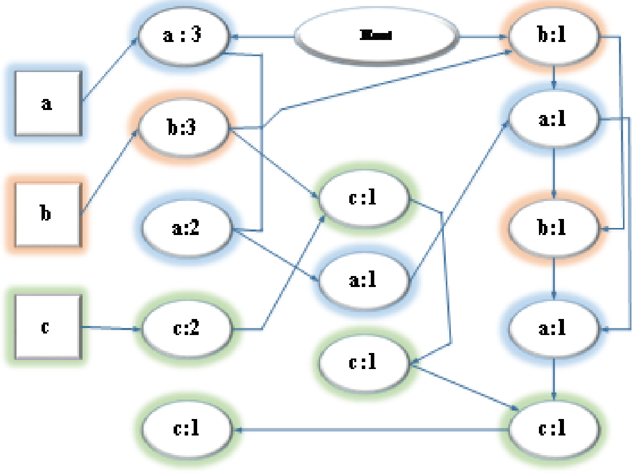

Figure 5

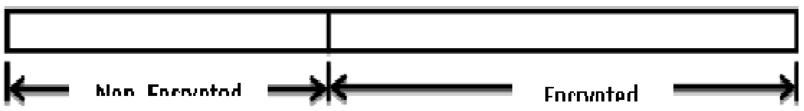

Figure 6

Figure 7

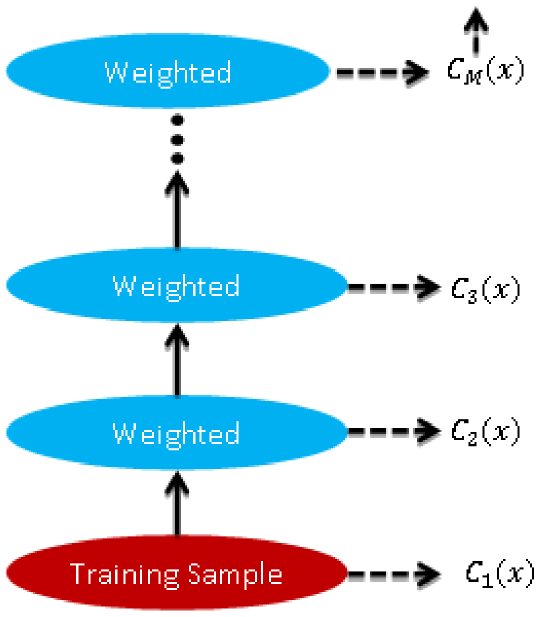

Figure 8

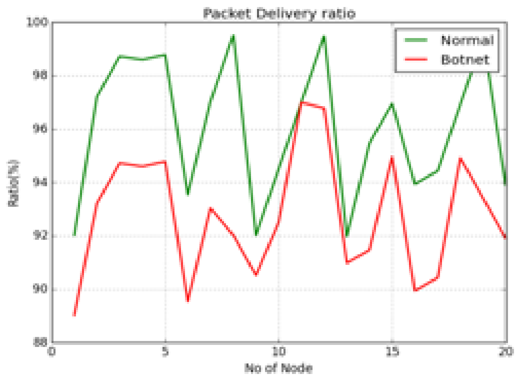

Figure 9

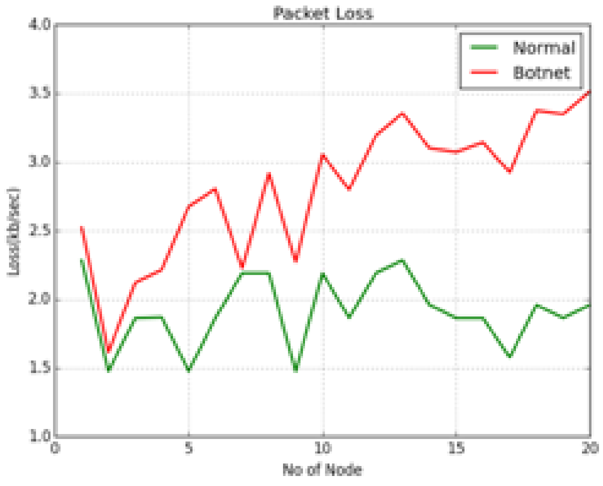

Figure 10

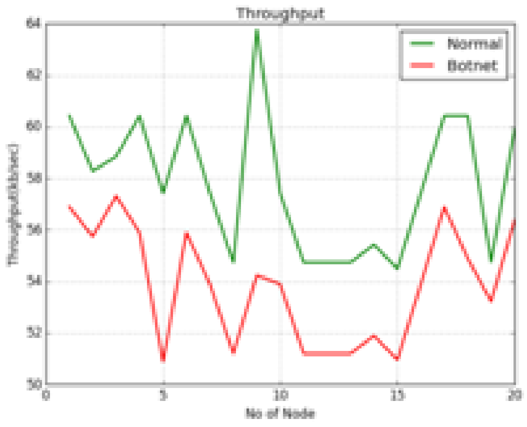

Figure 11

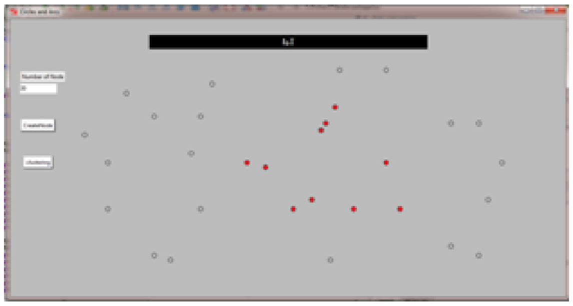

Figure 12

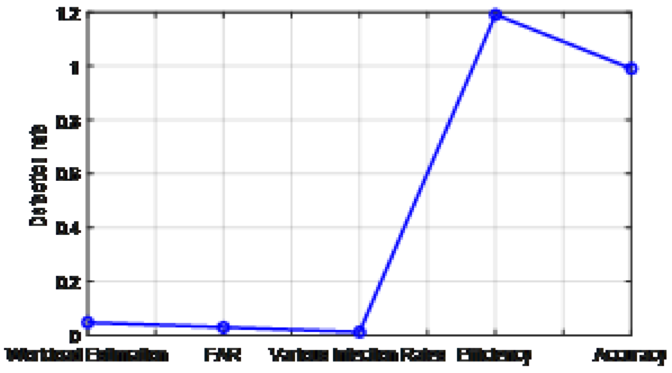

Figure 13

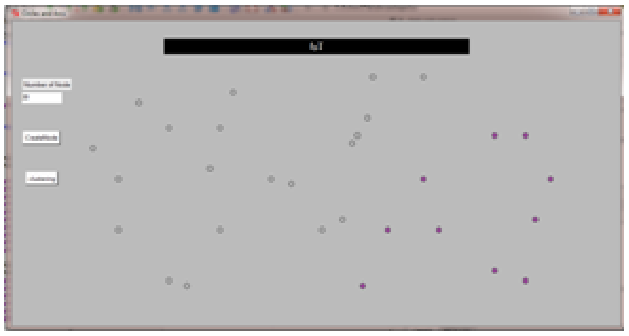

Figure 14

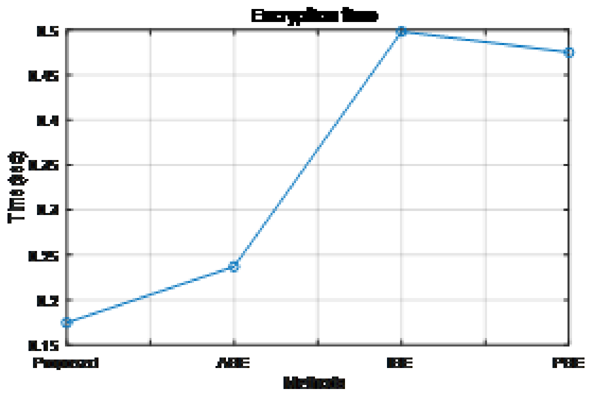

Figure 15

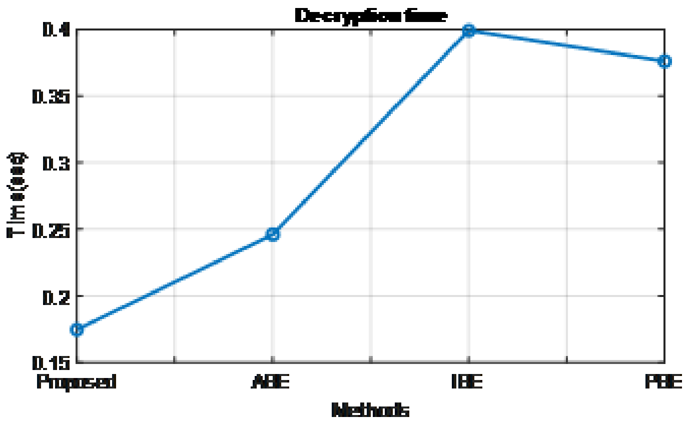

Figure 16

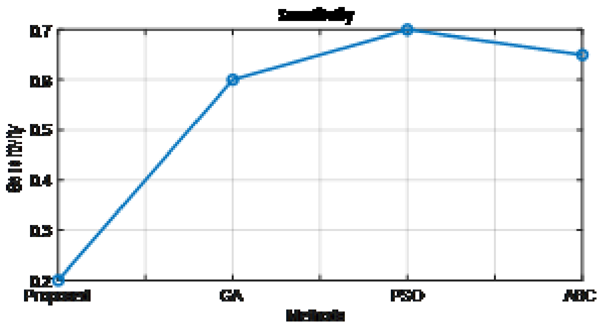

Figure 17

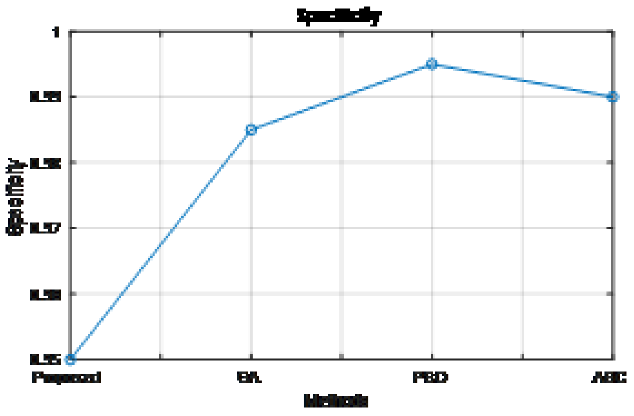

Figure 18

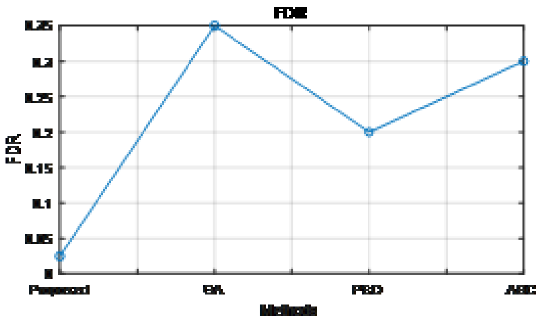

Figure 19

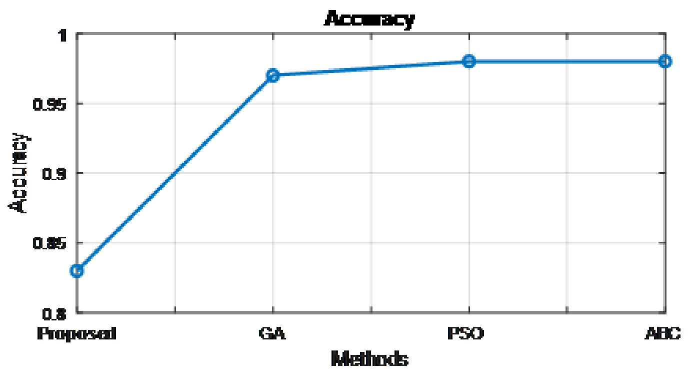

Figure 20

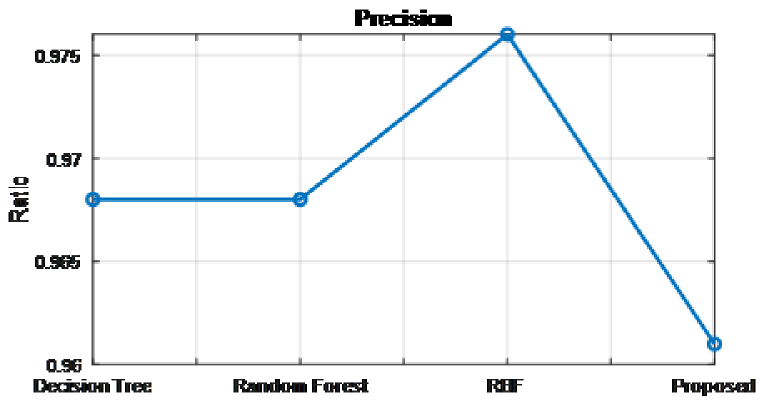

Figure 21

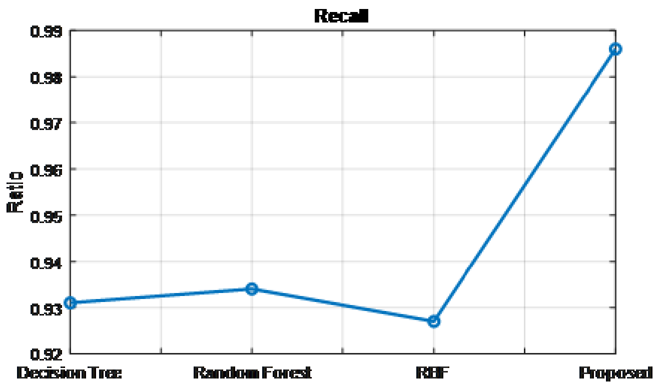

Figure 22

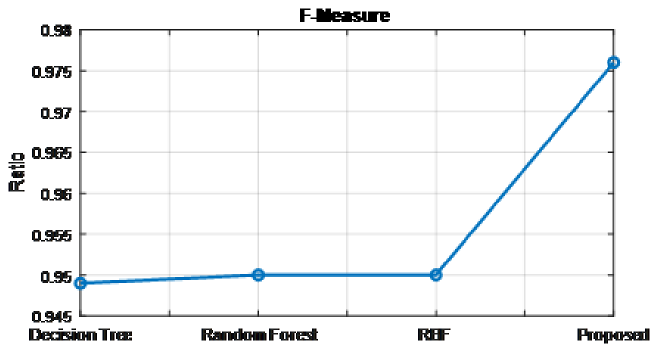

Figure 23

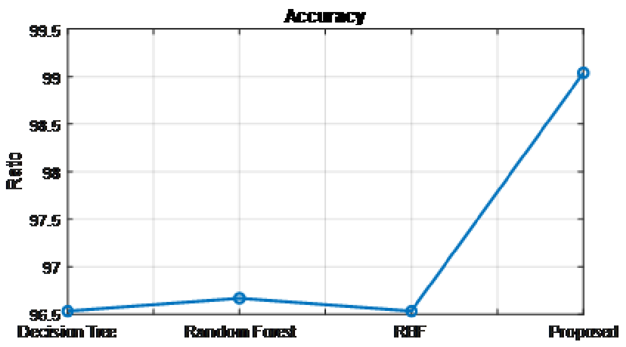

Figure 24

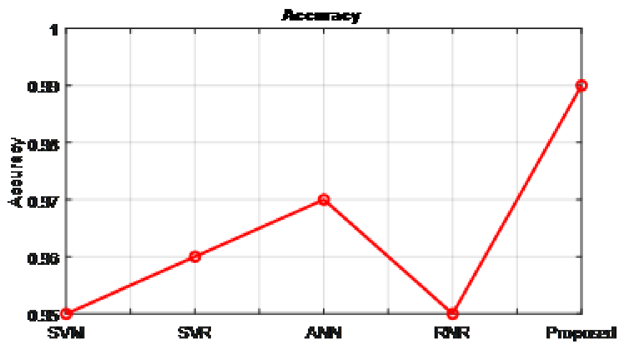

Figure 25

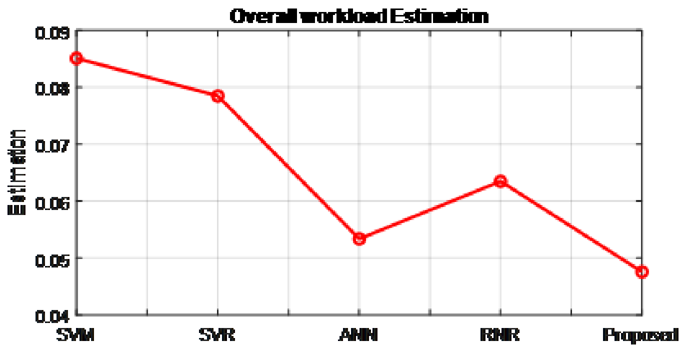

Figure 26

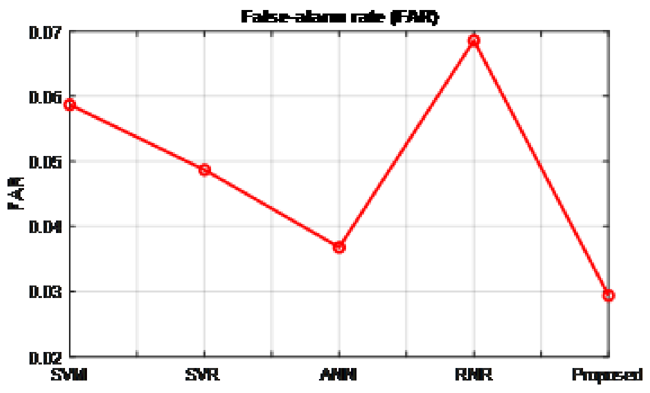

Figure 27

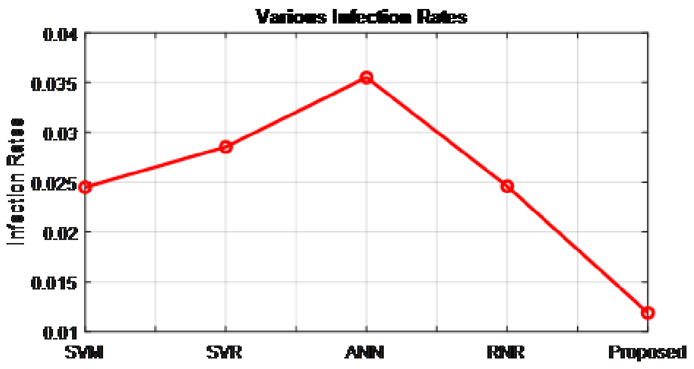

Figure 28

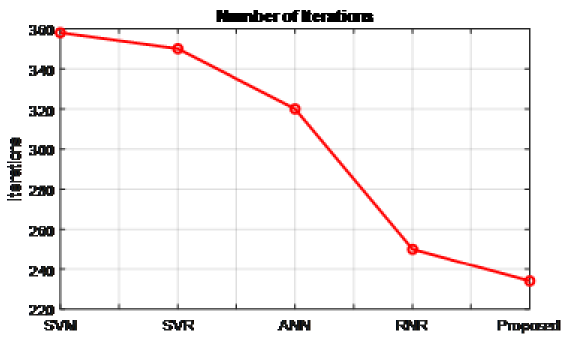

Figure 29

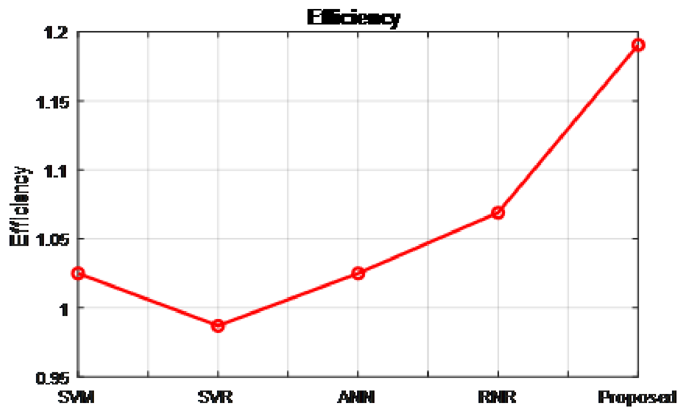

Figure 30