Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.018257
Article

Tech Science Press

Prevention of Runtime Malware Injection Attack in Cloud Using Unsupervised Learning

M. Prabhavathy[1,*] and S. UmaMaheswari[2]

[1]Department of CSE, Coimbatore Institute of Technology, Coimbatore, 641014, Tamilnadu, India
[2]Department of ECE, Coimbatore Institute of Technology, Coimbatore, 641014, Tamilnadu, India
*Corresponding Author: M. Prabhavathy. Email: prabhavathy.phd@gmail.com
Received: 02 March 2021; Accepted: 29 June 2021

Abstract: Cloud computing utilizes various Internet-based technologies to enhance the Internet user experience. Cloud systems are on the rise, as this technology has completely revolutionized the digital industry. Currently, many users rely on cloud-based solutions to acquire business information and knowledge. As a result, cloud computing services such as SaaS and PaaS store a warehouse of sensitive and valuable information, which has turned the cloud systems into the obvious target for many malware creators and hackers. These malicious attackers attempt to gain illegal access to a myriad of valuable information such as user personal information, password, credit/debit card numbers, etc., from systems as the unsecured e-learning ones. As an important part of cloud services, security is needed to protect business customers and users from unauthorized threats. This paper aims to identify malware that attacks cloud-based software solutions using an unsupervised learning model with fixed-weight Hamming and Mexiannet. Different types of attack methodologies and various ways of malicious instructions targeting unknown files in cloud services are investigated. The result and analysis in this study provide an evolution of the unsupervised learning detection algorithm with an accuracy of 94.05%.

Keywords: Security; malware; Hammingnet; Mexiannet; SaaS; PaaS

1 Introduction

Cloud computing is a major development field in the computer sector. Cloud is the space for storage, whereas cloud computing, based on internet usage per hour, helps deliver on-demand services. Utilizing a network to access servers that are hosted on the internet is one alternative to accessing a local server to perform operations like retrieving, storing, managing, and processing data. The objective of cloud computing is to share resources between the client and a server through cloud providers, cloud consumers, partners, and vendors. The resource sharing is achieved at various levels like infrastructure cloud (IT infrastructure management, i.e., platform as a service), business cloud (business as a service), software cloud (software as a service), and application cloud (application as a service). Currently, the market for mobile phones is increasing. Consequently, mobile cloud computing will be an inevitable future trend. As a matter of fact, statistical data shows that the number of internet-based mobile users increases 10% every year [1,2]. To create and use the web services [3], cloud service providers (CSP) offer cloud platforms and various cloud-based services. As such, customers are able to access those services anywhere, through a high-speed network. The assent in cloud computing is bringing about novel and simple ways to overcome limitations of restricted energy and the potential of adding on resources [4,5]. However, privacy protection, security, and interoperability bring major concerns to accessing cloud computing platforms and services. To avoid system fragility and defend the system against vulnerabilities from cyber attackers, various cybersecurity measures, tools, and techniques have been developed. Protecting the data from unauthorized access in a cloud platform is quite complicated. Nowadays, the data of private organization personals are more sensitive and need to be protected with more security on cloud platforms. One of the issues is that the authentication framework is difficult to extend to the cloud. Therefore, modifying the existing framework to support the cloud is crucial.

The meaning of security is abundant. As the key measure of security, the CIA triangle (Confidentiality, Integrity, and Availability) is a combination of various terms such as confidentiality (preventing data from unauthorized access), integrity (preventing unauthorized modification or deletion of the data), and availability (preventing unauthorized conceal of the data). Cloud computing service can be divided into three models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

Software-as-a-Service (SaaS): It is a process in which different service providers provide online services for different types of software applications. The customers can run the application without installing operating systems, application-specific software like databases, and word processing software. Instead, service providers (e.g., Dropbox, Google Apps, Salesforce.com) provide everything, which eliminates the maintenance of software by individuals and provides service on-demand.

Platform as a Service (PaaS): It is a process in which cloud service providers (e.g., Apache Stratos, Microsoft Azure, Force.com) provide a platform to run applications of users, such as developers or IT managers. It provides a high-level infrastructure integration to test and implement cloud applications. The user does not need to manage the infrastructure like operating systems, networks, servers, and storage.

2 Literature Survey

Maria and Eugene proposed a user-level security threat model for e-learning systems based on the notion that the attacks on e-learning systems are mainly through entry points (points used by an attacker to gain access). Characterization of the attackers is made, together with the identification of the assets and entry points, as well as the identification and classification of the threats [6]. The major security issues in computing systems are availability, integrity, confidentiality, and authentication. So, to provide a secure e-learning technology, it is important to identify the threats at the design phase. Therefore, the system designer should have complete knowledge about who, what, and how a system can be attacked.

Ateeq Ahmad and Mohammed Ahmed Elhossiny introduced various forms of e-learning security threats and discussed the tools and motivations to prevent security attacks. The seven kinds of risks identified to be directly or indirectly related to e-learning are virus threats, spyware threats, hackers, phishing threats, viral websites threats, adware, advertising and Trojans threats, and online social network site threats. Some of the common approaches for securing the e-learning system include recovering from viruses, worms, and Trojans, reducing spam, evaluating the web browser's security settings, protecting user privacy, supplementing passwords, and exercising caution with USB. According to the paper, e-learning systems can be protected against unauthorized security attacks by using some proper security tools and best practices [7].

R. Priya and J. Jayanthi have discussed different types of malware attacks over the layers of the e-learning system. They classified the operations of e-learning systems are into synchronous and asynchronous modes. Also, the implementation of availability, integrity, and confidentiality and the pillars of information security in these systems were discussed. Besides, emphasis was put forward to ensure the information is transferred over the network without any security breach. In addition, the e-learning systems were divided into three portions, including user, internet, and database server. The common security threats identified over e-learning websites are viruses, worms, Trojan horse, malware, adware, spyware, rootkit, etc. There are specified security threats over the layers. For example, on the user side, the security attack can be phishing, cross-site scripting, clickjacking, content spoofing, brute force, and authentication attack; on the internet side, the security attacks can be sniffing, DOS, DDOS, spoofing, replay attack, etc.; on the database server side, the attack can be SQL injection, LDAP injection, and weak authentication, etc [8,9]. The authors insisted that the e-learning progress must be executed with highly accepted security standardization. The e-learning application requires security measures such as proper authorization of user input, permissions, privileged access, and authentication to build the application with high security. Throughout the communication, i.e., the exchange of information between user and server, data must be encrypted by secure communication channels. Te-Shun Chou discussed cloud computing and thought that developing customized computer software, business applications, and online storage are the services provided by cloud computing platforms [10]. Many hackers attempt to make use of security vulnerabilities of the cloud architecture by data breaches in cloud platforms [11]. The cloud service models that are provided by many providers are SaaS, PaaS, and IaaS. The security threats arise due to misuse of cloud computing resources, including data breaches, wrapping attacks, malicious code, cyber theft through online mode, cloud security attacks, malware injection attacks, etc. As a countermeasure, the security policy with strict rules and regulations can help to manage the cloud more effectively. Access management must ensure that the users are authorized, and distrustful access to the resources is forbidden. Sometimes, intentional or unintentional insiders cause data breaches.

Some of the security breach countermeasure tools include malware scanners, system firewalls, vulnerability scanners, data integrity and security services, intrusion detection system, active domain name system and passive domain name system, encryption tools, user behavior capturing, and anomaly detection. Among these, the structure based on the file allocation table is exploited to prevent malware-based injection attacks. For XML Signature wrapping attacks, XML schema hardening technique is proposed to strengthen XML schema declarations [12]. Rajendra Patil, Harsha Dudeja, and Chirag Modi have discussed the malware detection method that is assisted by a virtual machine based on lightweight agents in a cloud computing environment. The method consists of two components: one for anomaly detection at the hypervisor level and the other for the agent at the virtual machine layer. Meanwhile, signature-based detection was used to generate optimal static features. Based on two fitness functions, features are derived following the extended binary bat algorithm. The profile is transferred to the hypervisor, where the anomaly detection using a random forest classifier is applied. The method classifies the executable as either normal or malware and generates an alert to the VM user. The functionality of the proposed framework was validated on a cloud testbed at NIT Goa and the latest malware datasets. Also, the VM security requirements fulfilled by the proposed framework were analyzed.

3 Delineation of Unsupervised Fixed Weight Competitive Net Learning Used to Analysis the Unknown File Instruction Sets

To find the injected code in the malicious code, the Fixed Weight Competitive Nets (FWCN) are used in the training process. In the FWCN, the competition is used in every training process of the activation function to find the transformation of normal code to a malicious file. In the implementation, three nets, including Maxnet, Mexican hat, and Hamming net, are exploited to handle the malicious activity against the normal file and malicious file with the fixed weight to find malware present in the unknown file [13–16]. One of the competition-based networks with fixed weight is Maxnet. During training, Maxnetact as a subnet to find the largest value in the node pool and the maximum value of the injected code variation. In the process of training Maxnet, all the nodes are interconnected, and identical weight values are assigned to the interconnected nodes. Fig. 1 shows the architecture of Maxnet, where fixed identical weights are assigned over the weighted interconnections [17].

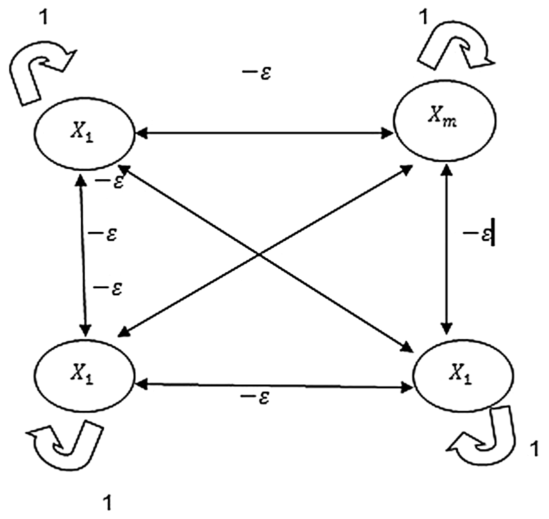

Figure 1: Maxnet structure

3.1 Pseudo Code of the Maxnet to Find the Unknown File Pool Instruction Set with Maximum Input Value Instruction Sets

The activation function applied on the maximum net is:

$$f(x) = \int_0^x \begin{matrix} \text{if} \quad x > 0 \\ \text{if} \quad x \le 0 \end{matrix} \tag{1}$$

Step 0: Initial weights are set for the Maxnet. The weight is set as [0 < ε < 1/m], where m is the total number of nodes taken for unknown file instructions. Let $x_j(0)$ be the input to the instruction set node x_j, and the weight of the consequent nodes is trained by

$$w_{ij} = \int_{-\varepsilon}^{1} \begin{matrix} \text{if} \quad i = j \\ \text{if} \quad i \ne j \end{matrix} \tag{2}$$

Step 1: Perform Steps 2–4 until the unknown file instructions set nodes meet the stopping criteria.

Step 2: Activate the training of the Maxnet, and the weight value of the interconnected nodes is updated with the activations of each node. For j = 1 to m,

$$x_{i(new)} = f\left[x_{i(old)-\varepsilon \sum_{i \ne j} x_{k(old)}}\right] \tag{3}$$

Step 3: Calculate the weight deviations of the unknown file instructions set during the training in the next iteration to save the activations obtained for use. For j = 1 to m,

$$x_{j(old)} = x_{j(new)} \tag{4}$$

Step 4: Finally, the stopping condition is tested as follows to determine the convergence of the network,

If (more than one node having a non-zero activation in the unknown file instruction set)

continue;

else

stop;

From the above algorithm, the input given for the above function f(-) is calculated by total input to node X_j taken away every other node together with its input

3.2 Mexican Hat Net Analysis of Unknown File Instruction Set with Competition

In this Mexican hat net used analyses the unknown file instruction with the nodes of interconnected and also to the connection with a specific layer of neural instruction set net, the unknown instruction set also receives some other external input weight values about the malicious instruction during the training of the Mexican hat net [18]. The architecture of the Mexican hat net is shown in Fig. 2, with each unknown instruction net node X_i for the interconnection pattern. The unknown instruction net nodes are put in proper places and linearly organized. In this way, there exist progressive connections between X_i and the neighboring units. Also, there are deleterious connections between X_i and aside neighboring units. The cooperation between the neighboring nodes and the deleterious connection region is of competition with the unknown instruction net node having a progressive connection region. The relative magnitudes between the progressive and deleterious weights consider the regions of the set that includes hexagonal. Linear grids depend on the size of the region. In Mexican Hat, there exist two symmetric regions around each neuron. In Fig. 3, X_i denotes the individual unknown instruction net node.

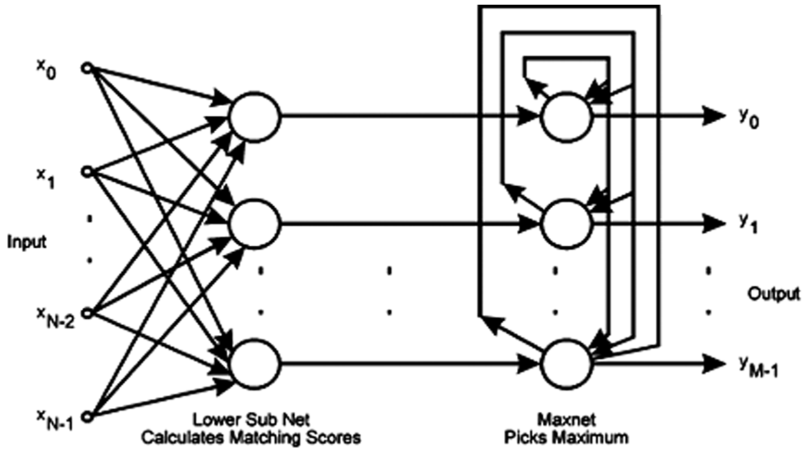

Figure 2: Mexican hat network

This unknown instruction net node is surrounded by other unknown instruction net nodes $x_{i+1}, x_{i-1}, x_{i+2}, x_{i-2},\ldots$ The nearest neighbors to the individual unknown instruction net node X_i are $x_{i+1}, x_{i-1}, x_{i+2},$ and x_{i-2}. Here, w_1 and w_2 are the weights associated with that to be considered progressive [19,20].

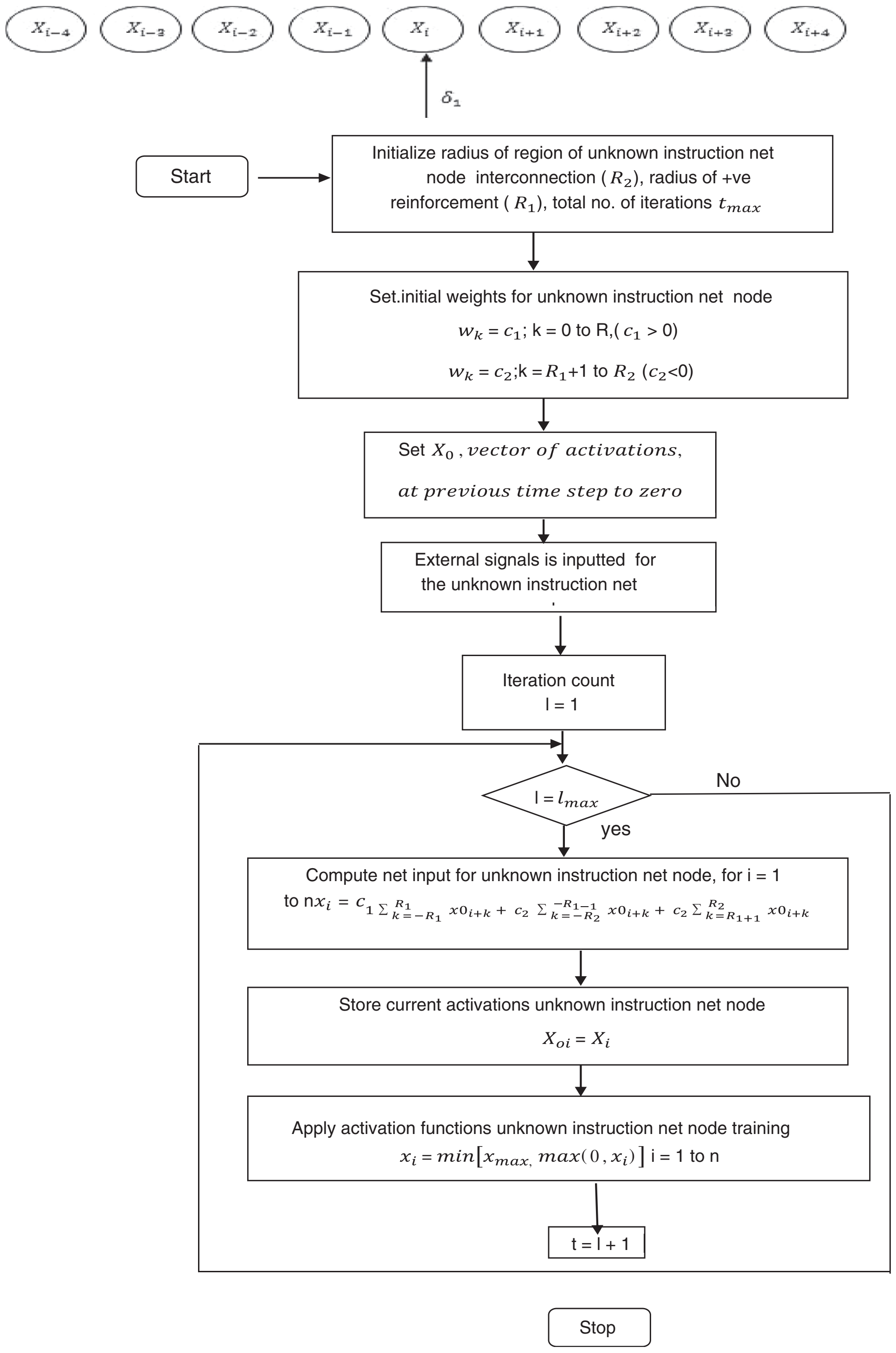
X_{i-4} X_{i-3} X_{i-2} X_{i-1} X_i X_{i+1} X_{i+2} X_{i+3} X_{i+4}
δ_1
Start
Initialize radius of region of unknown instruction net node interconnection (R_2), radius of +ve reinforcement (R_1), total no. of iterations t_{max}
Set.initial weights for unknown instruction net node
$w_k = c_1$; k = 0 to R,($c_1 > 0$)
$w_k = c_2$;k = R_1+1 to R_2 (c_2<0)
Set X_0, vector of activations, at previous time step to zero
External signals is inputted for the unknown instruction net
Iteration count I = 1
I = l_{max}
No
yes
Compute net input for unknown instruction net node, for i = 1 to n$x_i = c_1 \sum_{k=-R_1}^{R_1} x0_{i+k} + c_2 \sum_{k=-R_2}^{-R_1-1} x0_{i+k} + c_2 \sum_{k=R_1+1}^{R_2} x0_{i+k}$
Store current activations unknown instruction net node
$X_{oi} = X_i$
Apply activation functions unknown instruction net node training
$x_i = min[x_{max}, max(0, x_i)]$ i = 1 to n
t = I + 1
Stop

Figure 3: The process performed in Mexican hat network

The farthest neighbors to neuron x_i which are individuals that are taken as x_{i+3} and x. The weights associated with that to be considered as deleterious are denoted as W_3. From the observation, x_{i+4} and x_{i-4} are not connected to the individual neuron x_i, and there is no weighted interconnection between these connections. For easier understanding, the units within a radius of 2 to the unit x_i are assigned with progressive weights, and the units within a radius of 3 to unit x_i are assigned with deleterious weights, while the units farther away from the radius of 3 are not connected in any way to the neuron x_i. The flowchart of the Mexican hat is shown in Fig. 3. This figure depicts the processing of unknown file malicious weight analysis by the Mexican hat network during the training.

The various parameters used in the training algorithm for unknown instruction net node analysis are shown as follows.

R_2 is the radius of the interconnection regions in the unknown instruction net node;

Individual units X_i are connected by X_{i+k} and $X_{i-k}X_i$ for k = 1 to R_2.;

R_1 is the radius of the region along with progressive reinforcement ($R_1 < R_2$);

W_k is the weight between W_i and the units X_{i+k} and X_{i-k};

$0 \leq k \leq R_{1,}$ W_k = progressive

$R_1 \leq k \leq R_{2,}$ W_k = deleterious

δ is the external input signal for unknown instruction net node;

x is the vector of activation for unknown instruction net node;

X_0 is the vector of activation at the previous time step;

t_{max} is the contrast enhancement for the total number of iterations.

Once the external signal is input to the network, the iteration begins.

Step 0: Initializing the parameters, including R_1, R_2, and t_{max} accordingly.

Then Initialize the weights as

$w_k = c_1$ for k = 0,…, R_1 (where $c_1 > 0$)

$w_k = c_2$ for k = R_1 + 1,…, (where $c_2 > 0$)

Initialize $x_0 = 0$.

Step 1: External signal δ in of the input is given:

$$X = \delta$$

In the array x_0, the activations are saved as

$$For\ i = 1\ to\ n,$$

$$x_{0i} = x_i$$

Once the activations are stored, the iteration counter t is set to l.

Step 2: Perform steps 3–7, when t is less than t_{max}.

Step 3: For i = 1 to n, calculate the net input

$$x_i = c_1 \sum_{k=-R_1}^{R_1} x0_{i+k} + c_2 \sum_{k=-R_2}^{-R_{1-1}} x0_{i+k} + c_2 \sum_{k=R_{1+1}}^{R_2} x0_{i+k} \tag{5}$$

Step 4: Apply the activation function to the unknown instruction net node. For i= 1 to n,

$$x_i = \min[x_{max,}\ \max(0,\ x_i)] \tag{6}$$

Step 5: Save the current unknown instruction net node activations in x_0, i.e., for i= 1 to n,

$x_{0i} = x_i$

Step 6: Increment the iteration counter:

$t = t + 1$

Step 7: Test the stopping criterion that is shown as follows.

If $t < t_{max,}$then continue

Else stop

The progressive supplement used in this method can increase the activation units of unknown instruction net node with larger initial activations and reduce the activation of unknown instruction net node units with smaller initial activations by deleterious reinforcement. The unit unknown instruction net node x_i are used as an activation function at a definite time instant t is given by

$$x_i(t) = \text{f}\left[s_{i\,(t)} + \sum_k w_k x_{i+k} + k(t-1)\right] \tag{7}$$

The weighted signals that arrive at the previous time step from other unknown instruction net node units are the terms present within the summation symbol.

3.3 Hamming Network for the Further Analysis of Noisy Instruction Sets Which are Present in the Unknown File

The Hamming network selects the stored classes of unknown instruction net node that is at noisy vector presented as input to unknown instruction net nodes at a maximum Hamming distance (H). The bipolar and binary vectors are involved in this case. The maximum likelihood classifier in a Hamming network determines the maximum exemplar vector in the unknown instruction net node. In a clustering net, the weight vector for output is a codebook vector or exemplar vector for the same set of inputs, and a unit placed on the net is similar to an input unknown instruction net node vector. The exemplar vectors are determined by the weights of the net. The measure of the similarity between the net input unknown instruction net node vector and the stored malicious exemplar instruction set vector provides the difference between the total number of components and the corresponding Hamming distance between the vectors [21,22].

Considering two bipolar vectors a and y, the relationship between the two vectors is as follows

$x \cdot y = a \cdot d$

where a is the number of components in the unknown instruction net node vector, and it has a relationship with the bipolar vector; d is the number of the components with which the unknown instruction net node vectors disagree. Here a-dis the Hamming distance that exists between two unknown instruction net node vectors. The total number of components is n.

$n_1 = a + d$

i.e., $d = n_1 - a$

For simplification, we have

$$x \cdot y = a - d$$
$$x \cdot y = a - (n_1 - a)$$
$$x \cdot y = 2a - n_1$$
$$2a = x \cdot y + n_1$$
$$a = \frac{1}{2}(x \cdot y) + \frac{1}{2}(n_1)$$

By obtaining the value of a, it is stated that the weights that are assigned a value of 1½ (one-half) about the exemplar unknown instruction net node vector and $n_1/2$ can be initialized as bias. The largest net input can be used to calculate the unit and locate the net or unit that is particularly close to the exemplar vector. The net unknown instruction net node along with the largest unit obtained by Hamming net are input to the Maxnetto find the unknown instruction net node with malicious activity. In Fig. 4, the architecture based on the Hamming network is shown. There are two layers in the Hamming network. The first layer calculates the difference between the total number of unknown instruction net node components and the Hamming(H) distance between the input unknown instruction net node vector(x) and the vector pattern stored in the feedforward path of the unknown instruction net node. The response to a neuron in this layer is the sign for the minimum Hamming distance value between the category and the input, which is the representation of the unknown instruction net node. The second layer is composed of an unknown instruction net node Maxnet (can be expressed as a subnet) or a Winner-take-all-network (WTA). The WTA is a recurrent neural network of the Maxnet, and it is exploited to suppress the Maxnet output nodes, excluding the initial highest output node of the first layer.

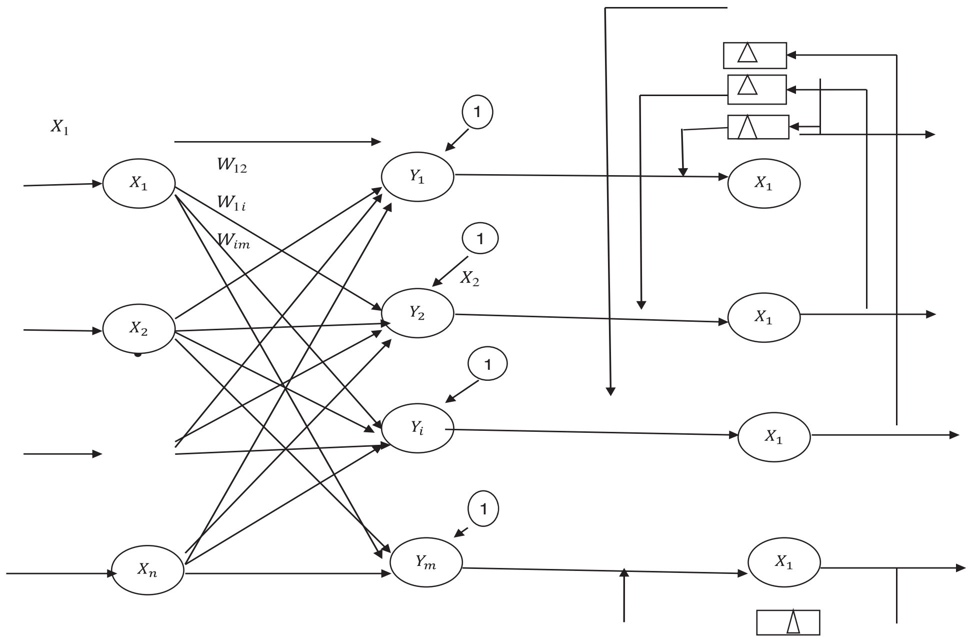

Figure 4: The structure of Hamming network

The purpose of Maxnet is to improve the response of the initial dominant unknown malicious instruction net node and suppress others nodes. The Maxnet performs repeated possessing so that the positive response is obtained by the jth node, and many responses of the remaining nodes are decayed to zero slowly. After the self-feedback positive connection is established, a negative feedback lateral is suppressed by the connection.

The value x is an oscillating input vector, and m represents the number of bipolar exemplar vectors that are denoted as e(1),…, e(j),…, e(m). To determine an exemplar vector, the Hamming net is used because it is closest to m in the input vector x.

The similarity between the input vector and exemplar vector is given by the input entering unit of the y_j net. The parameters used here are as follows:

n is the number of input unknown instruction net node units (the total number of I/O vector components)

m is the number of output malicious instruction net node units (the total number of exemplar vector components)

$e_1(j)$ is an exemplar vector of the jth component.

$e_1(j) = [e_{11}(j),\ldots,e_{1i}(j),\ldots,e_{1n}(j)]$

For the Hamming network, the testing algorithm is as follows:

Step 0: Initialize the weights of the unknown instruction net node.

For i = 1 to n & j = 1 to m,

$$w_{ij} = \frac{e_{1i(j)}}{2}$$

To store the values of the exemplar vector m, the bias value is initialized as follows

For j= 1 to m,

$$b_j = \frac{n}{2}$$

Step 1: Repeatedly perform steps 2–4 to calculate each input vector unknown instruction net node.

Step 2: The net input to each unknown instruction net node unit y_j is calculated by

$$y_{inj} = b_j + \sum\nolimits_{i=1}^{a} x_i w_{ij},\ j = 1 \text{ to m} \tag{8}$$

Step 3: Iteratively execute the Maxnettofind the exemplar vectors that are matched with the input unknown instruction net node patterns.

The Hamming network cannot find the entire unknown instruction net node vector. Instead, it retrieves only the closest class index. Meanwhile, the Hamming network is based on classification rather than associative memory. So, the Hamming network can be altered as an associative memory by including an additional property over the Maxnet to trigger a Maxnet structure stored in the weight vector winner unit.

4 Results and Discussions

In the experiment, the virtual machine is created in the Windows operating system, and Anaconda Python, Keras, and TensorFlow are installed in the VM. Meanwhile, the benign malware samples are taken and unpacked with EXE explorer. The training input is assigned with a fixed weight between 0 and 1, and the Hamming and Mexican algorithms are applied to process unknown samples. The detection accuracy of the instruction set analyzed from unknown files and against the existing algorithm is listed in Tab. 1. Cloud-based attack vectors were analyzed in the unknown instruction set.

Table 1: Comparisons of the proposed unsupervised learning (Hamming and Mexian) method with the existing methodologies on malware

Comparisons of proposed DSTM-AA over malware with existing methodologies	Detected the number of malware images	No malware files correctly identified as malware files [Ratio for TP %)]	No malware files wrongly identified as malware files Detected FP	No malware files wrongly identified as malware files [Ration for FP (%)]
Maria	957	90.02	75	0.09
Lei Cen	735	91.88	61	0.08
Proposed Unsupervised Learning (Hamming and Mexian)	**929**	**94.28**	**32**	**0.05**

The open-access and distributed structure of cloud computing offer a variety of services. Meanwhile, it becomes a vulnerable and attractive target for intruders. As shown in Figs. 5 and 6, the unsupervised learning with a fixed weight that is exploited to train the learning of unknown files using the Hamming and Mexian can maximize the detection accuracy of the malicious activity in the unknown file. The training of Hamming and Mexiannet by using unsupervised learning systems ensures the safety of cloud-based e-learning and other Paas/SaaS services.

For analysis, the total number of malware files used is 497.

For analysis, the total number of normal files used is 497.

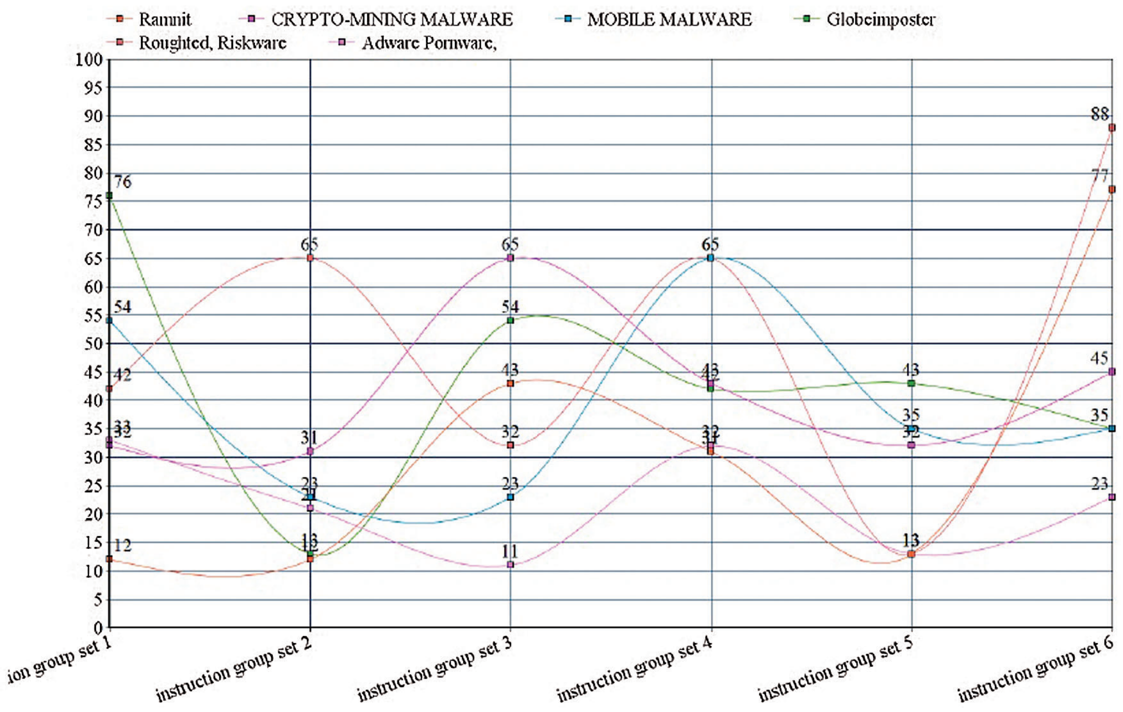

Figure 5: Unsupervised learning algorithm for Hamming and Mexian net analysis over the unknown instruction sets

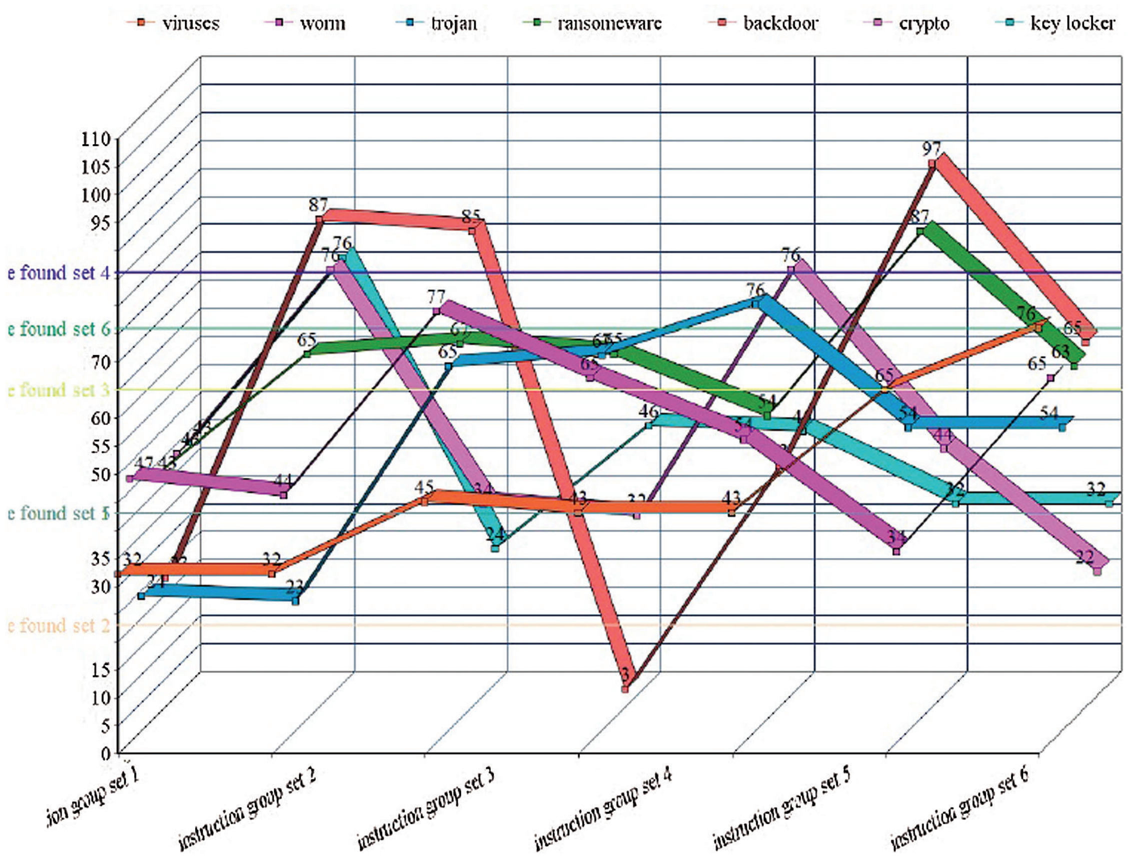

Figure 6: Unsupervised learning algorithm for Hamming and Mexiannet analysis over the unknown instruction sets

5 Conclusion

Internet penetration is increasing every day even in the developing world. Cloud-hosted services such as virtual machines pose a challenge to security researchers due to their complex architecture design. The unknown file intrusion set analysis is an important mechanism to protect the cloud data from unauthorized attacks. Meanwhile, unsupervised learning techniques have many advantages because of their adaptability, learning ability, and uniqueness for identifying malicious activity through an instruction set in the unknown file. The unsupervised learning methodologies of Hamming and Mexican neural network analysis help to identify malicious behaviors in the unknown file by the fixed weight and competition between the neurons to learn the unknown file. It can be adaptively trained to detect unknown attacks even when under environmental changes. In this work, unsupervised learning is exploited to identify the malware in the cloud computing models of SaaS and PaaS. In the future, the proposed method will be extended to identify the possible process attack on various cloud services.

Acknowledgement: We would like to thank TopEdit (www.topeditsci.com) for the English language editing of this manuscript.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] C. Yin, S. Huang, P. Su and C. Gao, “Secure routing for large-scale wireless sensor networks,” in *Proc. Int. Conf. on Communication Technology*, Beijing, China, vol. 2, pp. 1282–1286, 2003.

[2] Z. J. Hass, “Design methodologies for adaptive and multimedia networks,” *IEEE Communications Magazine*, vol. 39, no. 11, pp. 106–107, 2001.

[3] N. Krishnaraj, P. K. Kumar and K. S. Bhagavan, “Conceptual semantic model for web document clustering using term frequency,” *EAI Endorsed Transactions on Energy Web and Information Technologies*, vol. 5, no. 20, pp. 1–4, 2018.

[4] P. P. Bonissone, “Soft computing: The convergence of emerging reasoning technologies,” *Soft Computing—A Fusion of Foundations, Methodologies and Applications*, vol. 1, no. 1, pp. 6–18, 1997.

[5] A. Shabtai, L. T. Chekina, D. Mimran, L. Rokachand, B. Shapiraet *et al.*, “Mobile malware detection through analysis of deviations in application network behaviour,” *Computers & Security*, vol. 43, pp. 1–18, 2014.

[6] M. Nickolov and E. Nickolov, “Threat model for user security in e-learning systems,” *International Journal Information Technologies and Knowledge*, vol. 1, pp. 341–347, 2007.

[7] T. S. Chou, “Security threats on cloud computing vulnerabilities,” *International Journal of Computer Science & Information Technology*, vol. 5, no. 3, pp. 79–88, 2013.

[8] R. Priya and J. Jayanthi, “Security attacks and threats in e-learning,” *International Journal of Emerging Technology in Computer Science & Electronics*, vol. 21, no. 3, pp. 1–12, 2016.

[9] A. Jahan and M. A. Alam, “Intrusion detection system based on artificial intelligence,” *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, pp. 705–708, 2017.

[10] A. Bedi, N. Pandey and S. K. Khatri, “Analysis of detection and prevention of malware in cloud computing environment,” in *Proc. Amity Int. Conf. on Artificial Intelligence (AICAI)*, Dubai, United Arab Emirates, pp. 918–921, 2019.

[11] I. Shhadata, B. Bataineha, A. Hayajnehaand and Z. A. A. Sharifb, “The use of machine learning techniques to advance the detection and classification of unknown malware,” in *Proc. Int. Workshop on Data-Driven Security (DDSW 2020)*, Warsaw, Poland, vol. 170, pp. 917–922, 2020.

[12] J. Singh and J. Singh, “Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms,” *Information and Software Technology*, vol. 121, pp. 1–18, 2020.

[13] H. Rathore, S. Agarwal, S. K. Sahay and M. Sewak, “Malware detection using machine learning and deep learning,” in *Proc. Int. Conf. on Big Data Analytic 2018*, Springer, LNCS, vol. 11297, pp. 402–411, 2019.

[14] K. Demertzis, L. Iliadis and I. Bougoudis, “Gryphon: A semi-supervised anomaly detection system based on one class evolving spiking neural network,” *Neural Computing and Applications*, vol. 32, pp. 4303–4314, 2020.

[15] L. Cen, C. S. Gates, L. Si and N. Li, “A probabilistic discriminative model for android malware detection with decompiled source code,” *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 400–412, 2015.

[16] M. Siddiqui, M. C. Wang and J. Lee, “Detecting internet worms using data mining techniques,” *Systemics, Cybernetics and Informatics*, vol. 6, no. 6, pp. 1–6, 2009.

[17] Z. Aung and W. Zaw, “Permission based android malware detection,” *International Journal of Scientific & Technology Research*, vol. 2, no. 3, pp. 228–234, 2013.

[18] G. Kapse and A. Gupta, “Detection of malware on android based on application features,” *International Journal of Computer Science and Information Technologies*, vol. 6, no. 4, pp. 3561–3564, 2015.

[19] M. Siddiqui, M. C. Wang and J. Lee, “A survey of data mining techniques for malware detection using file features,” in *ACM-SE 46, Proc. of the 46th Annual Southeast Regional Conf. on XX*, Auburn Alabama, vol. 1, pp. 509–510, 2008.

[20] R. Patil, H. Dudeja and C. Modi, “Designing in-vM-assisted lightweight agent-based malware detection framework for securing virtual machines in cloud computing,” *International Journal of Information Security*, vol. 19, pp. 147–162, 2020.

[21] T. Thomas, A. P. Vijayaraghavan and S. Emmanuel, "*Machine Learning Approaches in Cyber Security Analytics*," I Edition. Springer, United States, 2020.

[22] R. Priya and J. Jayanthi, "Security attacks and threats in e-learning," *International Journal of Emerging Technology in Computer Science & Electronics*, vol. 21, no. 3, pp. 1–12, 2016.