Malwares Attack Detection Using Ensemble Deep Restricted Boltzmann Machine

K. Janani[1*] and **R. Gunasundari** [2**]
jananikumar6@gmail.com
Karpagam Academy of Higher Education
Coimbatore, India
gunasoundar04@gmail.com
Karpagam Academy of Higher Education
Coimbatore, India

Abstract

In recent times cyber attackers can use Artificial Intelligence (AI) to boost the sophistication and scope of attacks. On the defense side, AI is used to enhance defense plans, to boost the robustness, flexibility, and efficiency of defense systems, which means adapting to environmental changes to reduce impacts. With increased developments in the field of information and communication technologies, various exploits occur as a danger sign to cyber security and these exploitations are changing rapidly. Cyber criminals use new, sophisticated tactics to boost their attack speed and size. Consequently, there is a need for more flexible, adaptable and strong cyber defense systems that can identify a wide range of threats in real-time. In recent years, the adoption of AI approaches has increased and maintained a vital role in the detection and prevention of cyber threats. In this paper, an Ensemble Deep Restricted Boltzmann Machine (EDRBM) is developed for the classification of cybersecurity threats in case of a large-scale network environment. The EDRBM acts as a classification model that enables the classification of malicious flowsets from the largescale network. The simulation is conducted to test the efficacy of the proposed EDRBM under various malware attacks. The simulation results show that the proposed method achieves higher classification rate in classifying the malware in the flowsets i.e., malicious flowsets than other methods.

Keywords: *Cybersecurity, Deep Learning, Restricted Boltzmann Machine, Malware*

1. Introduction

Because of the increasing number of smart devices is increasing and the internetworking becomes more complex than they have ever been [1]. Various research [2]-[10] findings indicate that malware is exploiting newly discovered holes in network equipment in order to carry out its damaging objectives. In terms of technology, we can clearly observe that there is a fight between virus developers and network security professionals. When it comes to network data, the amount of information included within it limits the number of countermeasures that may be implemented [3]. Even while packet-level data can provide a finer level of detail, the resources required to gather it in an enterprise setting make it unfeasible to collect.

This problem was addressed by the introduction of the concept of a network flow, which provided academics with the capacity to develop algorithms that only considered certain data pertaining to network traffic [4] [5]. It is possible to use the methods mentioned in this area to detect a variety of different sorts of security breaches, which are discussed in greater depth under the background section.

Malware is a serious threat and its existence is one of the most difficult to detect. As soon as they have been infected, vulnerable devices can be used to send launch denial-of-service attacks, spam emails, and steal sensitive data [6].

Malware attackers are confronted with a huge challenge in disseminating their programmes to the greatest number of victims conceivable. By employing social engineering techniques, attackers can send email messages that entice recipients to download and install malware software that is linked to their own websites [7]. Despite the fact that this strategy is effective, it is frequently ineffective since it necessitates the participation of the victims. In order to evade detection, it is preferable to lure website visitors into visiting rogue web pages that take advantage of browser weaknesses instead (or their components, such as the PDF reader or the Flash player). Unlike in other scenarios, there is no requirement for the victim to engage with the virus in this one, as it is installed and launched without their knowledge [8].

There are three stages to the infection process in a drive-by download attack. To begin with, the attacker hopes to execute a small bit of code (shellcode) on the

Manuscript received May 5, 2024
Manuscript revised May 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.5.7

target computer to gather information about the victim. In order to accomplish this, the attacker first establishes a website that allows users to download exploit code using a drive-by download mechanism. When a victim navigates to a malicious page, the browser retrieves and executes the drive-by code that was previously downloaded [9]. When the exploit is successful, the browser will execute shellcode on the victim computer. The shellcode then executes the malware binary in the second step of the infection process after it has been downloaded (the installation phase). That when the malware programme actually goes to work and starts doing nasty things, which is the third step (the control phase). Malicious software frequently makes use of a remote command and control server (C & C). Attackers can use this connection to send and receive commands, drop new executables onto the infected host, and steal data from the infected host [10].

Modern anti-malware solutions are primarily concerned with avoiding infection during the first and third phases of the infection process. There has been a substantial amount of study into detecting malicious URLs that contain drive-by download exploits and stopping browsers from visiting malicious pages in the first place, with the goal of preventing the initial exploitation step from occurring. Honeyclients, for example, search the web in order to detect pages containing attack code as quickly as possible and turn their findings into blacklists of domains and URLs to avoid. Because of the fast turnover of malicious domains by attackers, blacklists are perpetually out of date and therefore ineffective. A significant increase has been observed in the number of assaults against honeyclients, who employ fingerprinting and obfuscation techniques in order to avoid detection.

Control phase research is also focused on identifying malicious code that has been executed on the end host, which is another important topic of study. It is possible that antivirus (AV) software is the initial layer of security employed by the great majority of computer users throughout the world. Whenever potentially harmful apps are stored on a computer hard drive or run from the command line, antivirus software relies primarily on signatures to detect and prevent them from being executed. Due to malware developer adaptation of their code to circumvent detection rates for these programmes are steadily falling. Upon successful infection, malware distributors communicate commands to the compromised hosts, which researchers have developed strategies to prohibit using signatures or reputation-based systems to prevent further infection.

A lot of the time, this request is fulfilled by simply launching the built-in functionalities of the browser in question. As far as the network is concerned, such connections appear to be routine requests made by users who have downloaded harmless applications. As illustrated below, after zooming out on a single viral download, the situation changes dramatically. Instead, a malware distribution infrastructure can be identified by examining a significant number of malware downloads from a large number of sites, all of which are tied to the same malware distribution campaign. In some ways, this virus distribution infrastructure can be thought of as a type of content distribution network. Due to the differences between the two types of content, it is easy to identify occasions where hazardous content is being conveyed.

According to this paper, the Ensemble Deep Restricted Boltzmann Machine (EDRBM) is a malware detection approach for network traffic that has been proposed. EDRBM is capable of distinguishing between adware, ransomware, viruses, worms, trojans, and botnets over the course of the detection process. First, EDRBM collects network flows based on the IP addresses of both parties involved in the transaction. A flowset is a term used to describe this form of aggregation. There are 441 statistical features extracted from network flow fields (for example, duration or source port) and they are divided into categories, as shown in the following figure. The RMI metric is used to identify the most essential characteristics of a piece of software. A feature vector for a flowset, also known as a fingerprint, is created by selecting the best of the best from among the best. With the use of fingerprints and the random forest classifier, it is feasible to determine whether flowsets contain traffic generated by the malware.

The most significant contribution of this research to malware detection is an effective method for grouping flows in the network in the form of flowsets and then the statistical fingerprints are collected in order to preserve the critical information in the flowsets, which is the most significant contribution of this research. As illustrated by our research, the study uses fingerprint as a statistical feature and it is dependent exclusively on the number of bytes delivered during transmission. In order to preserve user privacy, we use fingerprinting that is not concerned with IP addresses or ports, making it more resistant to

port spoofing, which is commonly exploited by contemporary malware to steal their identities. Because they do not rely on time-domain attributes, the fingerprints obtained during the evaluation are invariant with respect to network quality and throughput as well as with respect to the evaluation process. We are also capable of combining hundreds of flows into a flowset when the information is retained such that each flow provides information on its maliciousness and other characteristics.

2. Background

In a battle of wits that has lasted decades, security experts and malware developers are pitted against one another. Researchers have developed a variety of classification systems for malware that are based on the objectives and activities of the infection. Malware that makes use of network traffic patterns to conceal its presence is particularly sought after by researchers today. Botnets, which are networked computer systems that can only function while linked to the Internet, are of particular interest to the vast majority of researchers, who are particularly interested in identifying and analysing them. A network infection and coordination dialogue monitoring tool, BotHunter, has been created by Gu et al. [11].

Bilge et al. [12] developed botnet detection that pulls information from NetFlow including the number of flows, client access patterns, and temporal activity. Oujezsky et al. [13] developed time-based behavioural analysis and takes into account the duration of flows, their IP addresses, and their ports. BotMiner is a botnet detection technique developed by Gu et al. [11] that does not rely on the C&C topology or communication protocols of the botnet in question. Kheir et al. [14] also introduced a behavior modelling called BotSuer for detecting botnets, which takes advantage of network features and the analysis is conducted in behavioural manner. Fran et al. [15] propose that NetFlow data be used to construct a model of host dependency that may be used to detect botnets.

According to Amini et al. [16], clustering of NetFlow data can be used to detect botnets in network traffic. Botnet detection has gotten more attention in the research literature than wide malware detection by a single technology, which is understandable given the nature of the field. Bartos et al. [17] propose one of the most important and similar approaches to our own. The researchers have developed an invariant classification method for malware behaviour as part of their effort to identify known and previously unknown security risks. The method first bundle flows into bags and then uses statistical feature representations computed from network traffic to classify the malware. Perdisci et al. [18] finds the malware by analysing the HTTP traffic sent by the infected computer. HTTP traffic is examined by Rafique et al. [19], who developed a clustering malware method that uses clustering. However, if the packet content cannot be retrieved and the malware communicates by using bogus port numbers rather than standard port numbers, both of these methods will fail.

Using network traffic, the MalClassifier, created by AlAhmadi et al. [20] can automatically categorise different types of malwares. Mohaisen et al. [21] propose a technique that is similar to this one, which they term Chatter. When compared to MalClassifier, the extraction of HTTP requests by Chatter needs a more fine-grained examination of packets than the extraction of HTTP requests by MalClassifier. In real-world situations, both of these methods are less trustworthy than other techniques for malware analysis since they are dependent on the order in which packets are received by the network. When the model receives the network traffic, first things it does is organise the traffic into groups of flows. This task has been tackled in a number of different ways in the past.

A good illustration of this is the grouping of flows that occur within a specific time period of time. Grouping rules can be defined as the IP address [12] of both servers composing a flow [17], which is different from the IP address of a flow, which is different in other ways. We take a similar approach, categorising traffic based on source and destination IP addresses rather than the application or server function or the port that is being used. In contrast to dividing the time domain into discrete periods and then collecting all of the flows within each of those intervals, a flowset is defined by the value of its timeout. A variety of techniques for extracting information from aggregated network flows have been proposed, but ours is the first to rely simply on the data features of network flows to detect and distinguish between a wide range of malware types.

3. Proposed Method

In this research, an approach for identifying malware in network traffic is described, which involves clustering of flows into flowset or logically coherent units and analysing them. Every network flow that passes

between two hosts is collected by them during a specified amount of time (identified by their IP addresses). The timeout parameter used in the study finds the length of a flowset timeout interval, is used to create this type of interval of time. This value is used to determine whether or not an attempt is made to include a flow into the flowset has been successful. A flowset is generated as long as the IP addresses of the source and destination are the same.

If the features obtained from flowsets, which is comprised of flowset for differentiating one flowset from another, the flowset cannot be distinguished from other flowsets. EDRBM solves this challenge by extracting 441 features from each flowset, which are then combined. Time, port, and data are three sorts of feature groups that can be logically organised based on the flow fields from which these characteristics are derived. The time group includes the interarrival time (the time elapsed between the timestamps of consecutive flows) and the duration of the flow. Apart from flow fields for the protocol (such as port 6, which represents TCP), the port group contains flow fields for the source and destination ports as well as flow fields for the protocol (such as port 80, which represents HTTP). This information is provided by the data group if there are many flows in a flowset that each have a defined number of packets. When using the backing flow field-based feature, it is possible to extract a number of statistical characteristics (such as the standard deviation) for each of the feature groups (e.g., source port).

It is not possible to calculate feature groups in the same way as in the previous example. The underlying flow fields are represented as numeric values, which are then used to calculate characteristics for different time and data groups. Each collection will eventually be subjected to statistical analysis in order to derive a variety of statistical characteristics. It was not possible to construct a port group using this strategy. In this case, flow fields are treated as encoded values or numeric values would be counter-productive. Due to the lack of ordinary linkages between ports 80 (HTTP) and 443 (HTTPS), the order or distance between them failed to provide the difference of service between them. The group of features from the port is calculated by encoding the frequency of each flow respective field values in one-hot form and then dividing the result by the number of flows.

Based on the flowset field, statistical features are generated for each collection in the collection. All of the statistical variables, as well as the number of different frequency values, are included in the protocol-based collection.

Collections from source or destination ports are a statistical feature that can be added to this collection of statistical features as an additional statistical feature. In this collection of statistical features, we have included the most frequent ports, as well as the aggregated frequency of smaller ports that are not among the most frequently used ports in this dataset. For each of the three feature groups, a flows subset that are belonging to a flowset with IP address where the the flowset was produced is retrieved and stored in a separate file. The only two directions that can exist at the same time are the outgoing and arriving directions.

Algorithm 1: Proposed Model

Model a large-scale network

Cluster the domains

Group the flow into flowsets

Find the co-location of the domain, top-level unique domain names, matching URI path, matching files

Estimate the fitness function

Use EDRBM to classify the flowsets in the network

Find if the classified flowsets are of malicious one

Discard the malicious flowsets from the entire network

End the process

We connect two clusters when we find a single file that has been hosted by at least one element (domain) in each cluster. All clusters that contain two or more elements are considered to be CDNs. For the second step, we distinguish between malicious and benign clusters (CDNs). This distinction is made using a classifier that we trained on a small data set of manually labeled malicious and benign clusters. The classifier leverages six features, as described below:

3.1 Ensemble Deep Restricted Boltzmann Machine

It is possible to optimise the parameters of a generative model such as the RBM by utilising stochastic gradient ascent on training data and log-likelihood to optimise the parameters of the model. The chance of assigning a training instance (visible vector) to each

hidden vector is calculated by adding up all of the potential hidden vectors.

$$P(v) = Z^{-1} \sum_{h} \exp\left(-E(v,h)\right)$$

The log probability derivative of a training vector with regard to weight can be represented as follows:

$$\frac{\partial \log P(v)}{\partial w_{ij}} = \left\langle v_i h_j \right\rangle_{data} - \left\langle v_i h_j \right\rangle_{model}$$

When the $\langle\cdot\rangle_{data}$ and $\langle\cdot\rangle_{model}$ are expected to follow their respective expected distributions, this is denoted by $P(h|v)$. This results in a pretty straightforward learning strategy for stochastic gradients with the steepest ascent in a log probability:

$$P\left(h_j = 1 \mid v\right) = \sigma\left(b_j + \sum_{i=1}^{n} v_i w_{ij}\right)$$

In which the user is required to submit an initial learning rate of ε.

Because the hidden units in an RBM are not directly related to one another, it is possible to collect an unbiased sample of the data $\left\langle v_i h_j \right\rangle_{data}$. Assume a training vector in random v, where the state h_j of a hidden unit j is set to 1.

$$P\left(v_i = 1 \mid h\right) = \sigma\left(a_i + \sum_{j=m}^{n} h_j w_{ij}\right) (j = 1; 2;...;m)$$

where

$r(x)$ - logistic sigmoid function.

The same is true if we have a hidden vector h that allows us to collect an unbiased data sample from the visible state. Since there exist no direct connections between the neural units that are visible. $\left\langle v_i h_j \right\rangle_{model}$, on the other hand, fails to acquire an unbiased sample.

To begin, the method assigns the visible unit states to a vector (training set), which can be found here. The equation is used to compute the binary states in parallel, and it is written as in Eq. (4). It is possible to construct a reconstruction of the hidden units by setting v_i to 1, where the rate of probability is found using Eq. (5). This has resulted in the weight adjustments being provided by the

$$\Delta w_{ij} = \varepsilon\left(\left\langle v_i h_j \right\rangle_{data} - \left\langle v_i h_j \right\rangle_{recon}\right)$$

where $\left\langle v_i h_j \right\rangle_{recon}$ - Gibbs sampling distribution.

With only one step, alternate Gibbs sampling with an initialization of the data produces a distribution of the variables. The biases a_i and b_j should be addressed by using a similar learning process involving individual states rather than pairwise sums, rather than the whole learning process. To approximate $\left\langle \cdot \right\rangle_{model}$, the study develops an alternate sampling method that included alternating Gibbs sampling cycles. It has been demonstrated that it performs well enough in many significant scenarios despite approximating the log probability gradient in most cases.

3.2 Ensemble Classifier

This section provides a number of ways in providing ensemble RBM classifiers, which are motivated by the powerful representation capacity of ensemble RBMs, when it is combined with feature extraction methods. The study concentrate on the various ways bagging may be used in conjunction with RBMs because it simple and easy to install while still delivering good performance.

For starters, we suggest a training collection of independent instances, each of which is represented by an input feature (X) vector and a class label with a label space value as its input feature (X). Because of this, consider having an N-dimensional vector that includes training outputs as an N-dimensional matrix as the input to the algorithm. As an example, training instances can be thought of as a horizontal concatenation of two variables, such as X and Y. As part of an ensemble classifier, we use majority voting to combine the outputs of a large number of basic classifiers into a single result. Base classifiers can be created by combining bagging and RBMs in a variety of ways to achieve high accuracy and diversity.

3.4 RMI Estimation

In order to avoid the curse of dimensionality, we employ a process of selection to identify the most informative characteristics. The relative mutual information (RMI) is used to determine the relevance of

these relationships. Using feature selection, it is possible for the reduction of the computational cost and the memory for storing the flowset vectors by deleting the bulk of characteristics that are not useful for classification. Information exchanged between parties

$MI(X,Y) = ME(Y)\ CE(Y|X)$,

where

CE - conditional entropy and

ME - marginal entropy.

X – 2D flowsets array, and

Y - flowset labels array and this results in

$RMI\ (X,Y) = MI(X,Y)$.

Because of the amount of memory they take up in comparison to all the other alternatives, our RMI scores are used to determine which features are most significant to the user.

4. Results and Discussions

On the basis of malware-generated network traffic from the CTU-13 and MalRec datasets, we put our hypothesis to the test. We conducted two independent experiments. Then, using the malware datasets that we have already acquired, the study trains with the best performing to detect the traffic in a network.

Table 1: Dataset Specifications

Dataset	Parameter	Value
Malrec	Malware Recorded	66,301
	Hashing	MD5
	Network Activity	PCAP form
CTU-13	Total Recordings	13 captures or scenarios

For our research, the scikit-learn Python package, as well as the C# and Python programming languages, were employed. In order to conduct our research, we used a 512 GB of RAM server with 32 CPU cores.

The initial test made use of the MalRec [8] and CTU-13 [20] datasets as well as other publically available datasets, in addition to the MalRec and CTU-13 datasets. MalRec report created a traffic has been included in the dataset, as well as other MalRec reports. Examples of such tools are the AVClass [14] malware labelling tool, which allows us to categorise samples depending on the malware families to which they belong. For the purpose of determining the top 25 families based on the highest number of samples, we add up all of the samples from each family and then rank them. The other category has a total of 24,197 malware samples, which includes the samples from the other malware families. It was decided to create a separate category for it in order to eliminate the need to categorise it farther down. As part of our effort to identify the five most frequent malware kinds, we manually examine each member of the top malware families in order to determine which sort of malware it represents. The botnet traffic from the CTU-13 dataset is utilised to generate this list, which is why the botnet type is included in the list.

The proposed classifier is trained with 50 estimators and balanced weights to account for the differences in sample sizes across different classes. RMI rates the usefulness of each feature in each feature group and assigns a score to each feature in each feature group. We run a series of experiments in which the number of features is varied in order to identify the bare minimum of characteristics required to maximise classification performance. Based on the findings of our study, the study decided to concentrate on the top five data characteristics from this area for the time being (see Table 1).

Port and protocol spoofing, as well as differences in network quality, are all almost unaffected by the feature set that we have chosen for our technique. Following the identification of these characteristics, we analyse the classification performance of each malware species separately. Botnets received an F1-score of up to 94% as a result of this research, which was significantly higher than the random estimate for all other types of malwares, with the exception of ransomware.

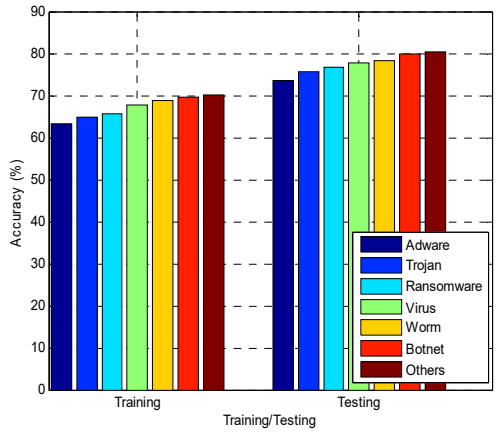

Figure 1: Accuracy

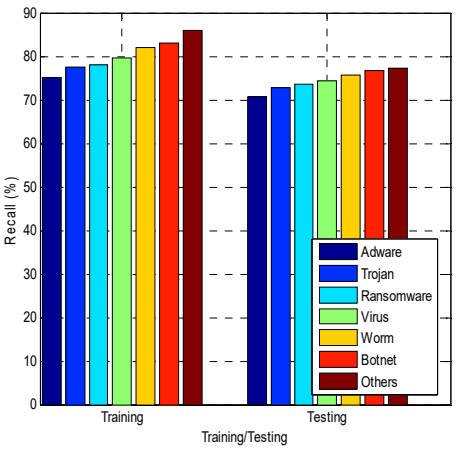

Figure 3: Recall

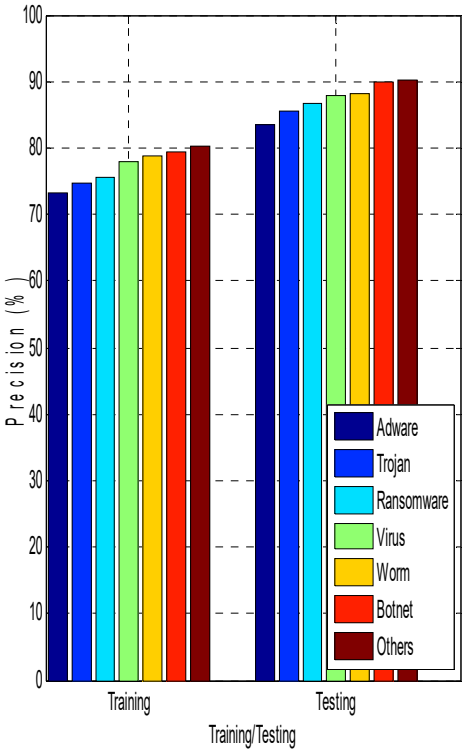

Figure 2: Precision

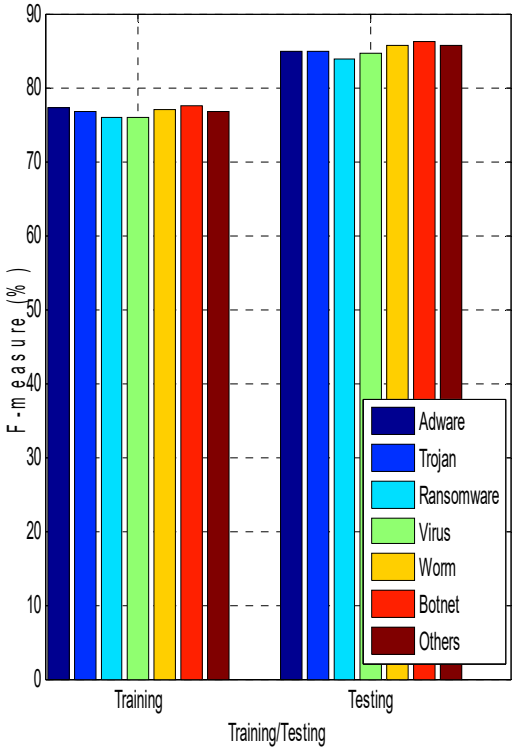

Figure 4: F-Measure

In our investigation, we discovered that certain adware and ransomware samples from other categories had been mislabelled. As a result, some virus kinds might share network properties while executing malevolent behaviours, which is why they are classified as such.

When it comes to a specific type of malware, there is nothing particularly noteworthy about these samples.

For the purpose of removing samples that cannot be discriminated against across different malware categories, we study the concept of a classification confidence level. The accuracy of classification improves, but only at the expense of fewer samples being labelled as distinctive as in Figure 1. As shown in Figure 2, we may improve the proposed classifier performance by adjusting the level of confidence threshold for classifications.

The study suggests that certain malware requires an additional number of samples in order to be identified and identified correctly (Figure 3). Each type of malware has a different confidence threshold, which can be adjusted to achieve the appropriate F1 score. In order to avoid false positives, malware with an F1 score greater than 0.6 must be detected using thresholds such as 0.4 for adware and 0.7 for ransomware. Figure 4 depicts the classification performance in terms of F1-measure that contains a constant confidence interval between 0.5 and 0.90 for three different confidence levels. It is possible to achieve a specific level of classification confidence for a given flowset classification percentage listed in the Samples column.

Following training on malware datasets, a classifier is trained on a real-world dataset and then applied to it as per the insights from the malware dataset analysis. When presented with a malware traffic sample, we have a high degree of confidence in our ability to detect it.

In our research, we discovered that an F1 score of 0.9 effectively reduces the total false positive rates. According to the data, there are approximately 100 adware flowsets/hour and ransomware flowsets of lesser than 20 per hour on average is obtained. It is found that after each month, worms and viruses containing similar confidence threshold (0.7) can be found, but not more than once. It is estimated that fewer than 10 instances of the malicious flowset exist in the wild when the confidence level is set at 0.95.

5. Conclusions

This study presents a novel EDRBM for classifying large-scale network cybersecurity concerns, which is intended to be used in the future. For the purpose of determining the model effectiveness, it is subjected to a wide range of security threats. According to the findings, the proposed strategy outperforms the other methods that were examined. On a large-scale network, malicious traffic sets with rates in the 106 per hour range were discovered. Given the incredibly low incidence of malware outbreaks, this is a reasonable response. In contrast with several false positive alerts acknowledged by the security operations centres, EDRBM finds minimal malicious flowsets in an accurate manner, whereas the latter receives an enormous number of false positive signals.

References

[1] Sarker, I. H. (2021). Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective. *SN Computer Science*, *2*(3), 1-16.

[2] Chen, D., Wawrzynski, P., & Lv, Z. (2021). Cyber security in smart cities: a review of deep learning-based applications and case studies. *Sustainable Cities and Society*, *66*, 102655.

[3] Liu, Z., Wang, R., Japkowicz, N., Tang, D., Zhang, W., & Zhao, J. (2021). Research on unsupervised feature learning for Android malware detection based on restricted Boltzmann machines. *Future Generation Computer Systems*, *120*, 91-108.

[4] Demertzis, K., Iliadis, L., Pimenidis, E., & Kikiras, P. (2022). Variational restricted Boltzmann machines to automated anomaly detection. *Neural Computing and Applications*, 1-14.

[5] Huma, Z. E., Latif, S., Ahmad, J., Idrees, Z., Ibrar, A., Zou, Z., ... & Baothman, F. (2021). A hybrid deep random neural network for cyberattack detection in the industrial internet of things. *IEEE Access*, *9*, 55595-55605.

[6] Thakkar, A., & Lohiya, R. (2021). A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges. *Archives of Computational Methods in Engineering*, *28*(4), 3211-3243.

[7] Bello, I., Chiroma, H., Abdullahi, U. A., Gital, A. Y. U., Jauro, F., Khan, A., ... & Abdulhamid, S. I. M. (2021). Detecting ransomware attacks using intelligent algorithms: Recent development and next direction from deep learning and big data perspectives. *Journal of Ambient Intelligence and Humanized Computing*, *12*(9), 8699-8717.

[8] Gupta, C., Johri, I., Srinivasan, K., Hu, Y. C., Qaisar, S. M., & Huang, K. Y. (2022). A Systematic Review on Machine Learning and Deep Learning Models for Electronic Information Security in Mobile Networks. *Sensors*, *22*(5), 2017.

[9] Basit, A., Zafar, M., Liu, X., Javed, A. R., Jalil, Z., & Kifayat, K. (2021). A comprehensive survey of AI-enabled phishing attacks detection techniques. *Telecommunication Systems*, *76*(1), 139-154.

[10] Tsimenidis, S., Lagkas, T., & Rantos, K. (2022). Deep learning in iot intrusion detection. *Journal of Network and Systems Management*, *30*(1), 1-40.

[11] Gu, G., Porras, P. A., Yegneswaran, V., Fong, M. W., & Lee, W. (2007, August). Bothunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security Symposium* (Vol. 7, pp. 1-16).

[12] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., & Kruegel, C. (2012, December). Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference* (pp. 129-138).

[13] Oujezsky, V., Horvath, T., & Skorpil, V. (2016, June). Modeling botnet C&C traffic lifespans from netflow using survival analysis. In *2016 39th International Conference on Telecommunications and Signal Processing (TSP)* (pp. 50-55). IEEE.

[14] Kheir, N., & Wolley, C. (2013, November). Botsuer: Suing stealthy p2p bots in network traffic through netflow analysis. In *International Conference on Cryptology and Network Security* (pp. 162-178). Springer, Cham.

[15] François, J., Wang, S., & Engel, T. (2011, May). BotTrack: tracking botnets using NetFlow and PageRank. In *International Conference on Research in Networking* (pp. 1-14). Springer, Berlin, Heidelberg.

[16] Amini, P., Azmi, R., & Araghizadeh, M. (2014). Botnet detection using NetFlow and clustering. *Advances in Computer Science: an International Journal*, *3*(2), 139-149.

[17] Bartos, K., Sofka, M., & Franc, V. (2016). Optimized invariant representation of network traffic for detecting unseen malware variants. In *25th USENIX Security Symposium (USENIX Security 16)* (pp. 807-822).

[18] Perdisci, R., Lee, W., & Feamster, N. (2010, April). Behavioral clustering of http-based malware and signature generation using malicious network traces. In *NSDI* (Vol. 10, p. 14).

[19] Rafique, M. Z., & Caballero, J. (2013, October). Firma: Malware clustering and network signature generation with mixed network behaviors. In *International Workshop on Recent Advances in Intrusion Detection* (pp. 144-163). Springer, Berlin, Heidelberg.

[20] AlAhmadi, B. A., & Martinovic, I. (2018, May). MalClassifier: Malware family classification using network flow sequence behaviour. In *2018 APWG Symposium on Electronic Crime Research (eCrime)* (pp. 1-13). IEEE.

[21] Mohaisen, A., West, A. G., Mankin, A., & Alrawi, O. (2014, October). Chatter: Classifying malware families using system event ordering. In *2014 IEEE Conference on Communications and Network Security* (pp. 283-291). IEEE.