Malware Attack Detection in Large Scale Networks using the Ensemble Deep Restricted Boltzmann Machine

Janani Kumar
Department of Computer Science, Karpagam Academy of Higher Education, India
jananikumar6@gmail.com (corresponding author)

Gunasundari Ranganathan
Department of Computer Applications, Karpagam Academy of Higher Education, India
gunasoundar04@gmail.com

Received: 16 July 2023 | Revised: 5 August 2023 | Accepted: 22 August 2023

 DOI: https://doi.org/10.48084/etasr.6204

ABSTRACT

Today, cyber attackers use Artificial Intelligence (AI) to boost the sophistication and scope of their attacks. On the defense side, AI is used to improve defense plans, robustness, flexibility, and efficiency of defense systems by adapting to environmental changes. With the developments in information and communication technologies, various exploits that are changing rapidly constitute a danger sign for cyber security. Cybercriminals use new and 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. This paper presents an Ensemble Deep Restricted Boltzmann Machine (EDRBM) to classify cybersecurity threats in large-scale network environments. EDRBM acts as a classification model that enables the classification of malicious flowsets in a large-scale network. Simulations were carried out to evaluate the efficacy of the proposed EDRBM model under various malware attacks. The results showed that the proposed method achieved a promising malware classification rate in malicious flowsets.

Keywords-malware; restricted Boltzmann machine; cyberthreat; deep learning

I. INTRODUCTION

Nowadays, the Internet has become more complex due to the increasing number of smart devices [1]. Various studies have shown that malware exploits newly discovered holes in network equipment [2]. In terms of technology, there is a fight between virus developers and network security professionals. Regarding network data, the amount of information included limits the number of countermeasures that can be implemented [3]. Although packet-level data can provide a finer level of detail, the resources required to collect them in an enterprise setting make them unfeasible to collect. This problem was addressed by introducing the concept of network flow, which provides the ability to develop algorithms that only consider specific data on network traffic [4-5]. These methods can be used to detect various security breaches. Malware is a severe threat, and its existence is one of the most difficult to detect. Vulnerable devices can be used to launch denial-of-service attacks, send spam emails, or steal sensitive data as soon as they are infected [6]. Using social engineering techniques, attackers can send email messages that entice recipients to download and install malware software [7]. Unlike other scenarios, the victim is not required to engage with the virus, as it is installed and launched without his knowledge [8].

In a download attack, there are three stages in the infection process. At first, the attacker hopes to execute a small code (shellcode) on the target computer to gather information about the victim. To accomplish this, the attacker 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 [9]. When the exploit is successful, the browser executes the shellcode on the victim's computer. The shellcode executes the malware binary after downloading it (installation phase). Finally, the malware program is run (the control phase). Malicious software frequently uses a remote command and control server. Attackers can use this connection to send and receive commands, drop new executables, and steal data from the infected host [10].

A significant increase has been observed in the number of assaults against honey clients using fingerprint and obfuscation techniques to avoid detection [11]. Control phase research also focuses on identifying malicious code that has been executed

on the end host. Antivirus software may be the initial layer of security used by the vast majority of computer users around the world. Antivirus software relies primarily on signatures to detect and prevent potentially harmful apps from being executed on a computer. As malware developers adapt their code to circumvent detection, the effectiveness of antivirus software is steadily decreasing. Upon successful infection, malware distributors send commands to compromised hosts, while researchers have developed strategies to prevent further infection using signatures or reputation-based systems [12].

The Ensemble Deep Restricted Boltzmann Machine (EDRBM) is a malware detection approach for network traffic. EDRBM can distinguish between adware, ransomware, viruses, worms, trojans, and botnets in the detection process [13]. At first, EDRBM collects network flows based on the IP addresses of both parties involved in a transaction. 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 divided into several categories. The Relative Mutual Information (RMI) metric is used to identify the most essential characteristics of software. A feature vector for a flowset, also known as a fingerprint, is created by selecting the best among the best. Using fingerprints and the random forest classifier makes it feasible to determine whether flowsets contain traffic generated by malware. This study aimed to develop an effective method for grouping flows in the network in the form of flowsets. Then statistical fingerprints are collected to preserve critical information in the flowsets.

Researchers are particularly interested in identifying and analyzing botnets. In [14], the BotHinter network infection and coordination dialogue-monitoring tool was presented, which did not rely on the C&C topology or the communication protocols of the botnet in question. In [15], a time-based behavioral analysis was presented that takes into account the duration of flows, their IP addresses, and their ports. In [16], BotSuer behavior modeling was presented to detect botnets, taking advantage of network features and performing analysis in a behavioral manner. In [17], NetFlow data was proposed to construct a host dependency model to detect botnets. According to [18], the clustering of NetFlow data can be used to detect botnets in network traffic. In [19], an invariant classification method was developed for malware behavior to identify known and previously unknown security risks. This method first bundles flows into bags and then uses statistical feature representations computed from network traffic to classify the malware. In [20], malware was detected by analyzing HTTP traffic. HTTP traffic was also examined in [21] using a clustering malware method. However, if the packet content cannot be retrieved and the malware communicates using bogus instead of standard port numbers, both methods will fail. In [22], the MalClassifier was presented to automatically categorize different malware using network traffic. In [23], Chatter was proposed, which is a similar technique to MalClassifier but requires a more fine-grained examination of packets when extracting HTTP traffic [24]. In real-world situations, both methods are less trustworthy than other techniques for malware analysis, as they depend on the order in which the network receives packets.

The proposed EDRBM introduces an effective method for grouping flows into flowsets. Unlike previous approaches that relied on time-domain attributes or port-based information, EDRBM uses statistical features from flowsets, making it more resistant to contemporary malware techniques such as port spoofing. This approach allows for the detection and differentiation of a wide range of malware types, making it a valuable contribution to cybersecurity. Overall, the EDRBM approach addresses the increasing sophistication of cyber-attackers using AI and the need for flexible and strong cyber-defense systems. Using ensemble methods and statistical features from flowsets, it provides a promising solution for real-time detection and prevention of cyber threats in complex network environments. This method uses the fingerprint as a statistical feature that depends exclusively on the number of bytes delivered during transmission [25]. To preserve user privacy, fingerprinting that is not concerned with IP addresses or ports was used, making it more resistant to port spoofing, which is commonly exploited by contemporary malware to steal identities. As fingerprints do not rely on time-domain attributes, those obtained during the evaluation are invariant for network quality and throughput, as well as for the evaluation process. Combining hundreds of flows into a flowset, information is retained such that each flow provides information on its maliciousness and other characteristics.

A good illustration of this is the grouping of flows within a specific period. Grouping rules can be defined as the IP address [26] of both the servers that make up a flow [19], which is different from the IP address of a flow. A similar approach is categorizing traffic based on source and destination IP addresses rather than the application or server function or the port being used. Unlike dividing the time domain into discrete periods and then collecting all flows within each interval, a flowset is defined by its timeout value [27]. Several techniques have been proposed to extract information from aggregated network flows, but the proposed method is the first to rely solely on the data features of network flows to detect and distinguish between a wide range of malware types. Irregularities were identified by analyzing user actions and patterns using AI. Deep learning has become the norm in research directions such as image classification, semantic segmentation, and natural language processing. In [28], the commonly used deep learning models in network attack detection, characterization, and traffic feature extraction were presented.

The proposed EDRBM model showed promising results in classifying malware flowsets, but there is not enough evidence to demonstrate how it significantly outperforms or offers unique advantages over other established methods in the field of cyber security. Further comparative analysis and benchmarks against state-of-the-art approaches are necessary to establish its novelty and superiority.

II. PROPOSED METHOD

EDRBM is used to classify unsupervised data of a probability distribution, reduce dimensionality, and extract more meaningful features. It includes connections only between visible and hidden nodes. The set of connections in hidden layers represents the probability of input data during the

training process. In the visible layer, data patterns are observed by each neuron, while the hidden layer is used to explain the patterns observed by the visible neurons. The learning rate is adjusted during the learning phase so that the model is not under or overfit. To gradually improve accuracy, the process is repeated over multiple iterations.

Every network flow between two hosts is collected during a specified amount of time, identified by their IP addresses. The timeout parameter determines the length of a flowset timeout interval and is used to determine whether an attempt to include a flow into the flowset has been successful. A flowset is generated as long as the source and destination IP addresses do not change. EDRBM extracts 441 features from each flowset, which are combined. Time, port, and data are three kinds of feature groups that can be logically organized based on the flow fields these characteristics are derived. The time group includes the inter-arrival time (the time elapsed between the timestamps of consecutive flows) and the duration of the flow. The port group contains flow fields for the source and destination ports and the protocol. This information is provided by the data group if there are many flows in a flowset, where each has a defined number of packets. When using the backing flow field-based feature, it is possible to extract several statistical characteristics for each feature group.

The underlying flow fields are represented as numeric values and then used to calculate characteristics for different time and data groups. Each collection is eventually subjected to statistical analysis to derive a variety of statistical characteristics. The group of features from the port is calculated by encoding the frequency of each flow's respective field values in one-hot form and then dividing the result by the number of flows. Statistical features are generated for each collection based on the flowset field. All statistical variables, as well as the number of different frequency values, are included in the protocol-based collection.

Collections from source or destination ports can be added as an additional statistical feature. In this collection of statistical features, the most frequent ports and the aggregated frequency of less frequently used ports were included. For each of the three feature groups, a flow subset that belongs to a flowset with the IP address it was produced, is retrieved and stored in a separate file. The only two directions that can exist simultaneously are the outgoing and arriving directions.

```
Step 1: Model a large-scale network
Step 2: Cluster the domains
Step 3: Group the flow into flowsets
Step 4: Find the co-location of the domain,
        top-level unique domain names, matching
        URI path, matching files
Step 5: Estimate the fitness function
Step 6: Use EDRBM to classify the flowsets in the
         network
Step 7: Find if the classified flowsets are of
        malicious one
Step 8: Discard the malicious flowsets from the
        entire network
Step 9: End the process
```

Fig. 1. Algorithm of the proposed model.

Two clusters are connected when a single file is found to be hosted by at least one element (domain) in each cluster. All clusters that contain two or more elements are considered CDNs. In the second step, malicious and benign CDNs are distinguished using a classifier trained on a small dataset of manually labeled malicious and benign clusters.

III. ENSEMBLE DEEP RESTRICTED BOLTZMANN MACHINE

It is possible to optimize the parameters of a generative model such as the RBM by utilizing stochastic gradient ascent on training data and log-likelihood. The chance of assigning a training instance (visible vector) to each hidden vector is calculated by adding up all potential hidden vectors.

$$P(v) = Z^{-1} \sum_h exp\big(-E(v,h)\big) \quad (1)$$

The log probability derivative of a training vector about weight can be represented as follows:

$$\frac{\partial \log P(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{\text{model}} \quad (2)$$

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

$$P\big(h_j = 1|v\big) = \sigma\big(b_j + \sum_{i=1}^{n} v_i w_{ij}\big) \quad (3)$$

where the user is required to submit an initial learning rate of ε. As the hidden units in an RBM are not directly related to one another, it is possible to collect an unbiased sample of the data $\langle v_i h_j \rangle_{data}$. Assume a training vector in random v, where the state h_j of a hidden unit j is set to 1.

$$P(v_i = 1|h) = \sigma\big(a_i + \sum_{j=m}^{n} h_j w_{ij}\big), \ j = 1,2,\dots,m \quad (4)$$

where $r(x)$ is the logistic sigmoid function.

The same is true if there is a hidden vector h that allows to collect an unbiased data sample from the visible state. Since there are no direct connections between the visible neural units, $\langle v_i h_j \rangle_{model}$ fails to acquire an unbiased sample. To begin, the method assigns the visible unit states to a vector (training set). Equation (4) is used to compute the binary states in parallel. It is possible to construct a reconstruction of the hidden units by setting v_i to 1. This results in the weight adjustments being provided by:

$$\Delta w_{ij} = \varepsilon \left(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon} \right) \quad (5)$$

where $\langle v_i h_j \rangle_{recon}$ is the Gibbs sampling distribution.

With only one step, alternate Gibbs sampling with an initialization of the data produces a distribution of the variables. Biases a_i and b_j should be addressed using a similar learning process involving individual states. To approximate $\langle \cdot \rangle_{model}$, the study developed an alternate sampling method that included alternating Gibbs sampling cycles. This has been demonstrated to perform well enough in many significant scenarios despite approximating the log probability gradient in most cases.

A. Ensemble Classifier

This section provides a number of ways to provide ensemble RBM classifiers, motivated by their powerful representation capacity when combined with feature extraction methods. This study focused on the various ways of bagging that can be used in conjunction with RBMs because it is simple and easy to install and delivers good performance. To start, a training collection of independent instances is suggested, each represented by an input feature (X) vector and a class label with a label space value. Because of this, consider having an N-dimensional vector that includes training output as an N-dimensional matrix as input to the algorithm. For example, training instances can be considered a horizontal concatenation of two variables such as X and Y. As part of an ensemble classifier, majority voting is used to combine the output of many basic classifiers into a single result. Base classifiers can be created by combining bagging and RBMs in various ways to achieve high accuracy and diversity.

B. RMI Estimation

To avoid the curse of dimensionality, a selection process was used to identify the most informative characteristics. RMI was used to determine the relevance of these relationships. Using feature selection, it is possible to reduce computational cost and memory for storing the flowset vectors by removing the characteristics that are not useful for classification. Information exchanged between parties $MI(X,Y) = ME(Y)CE(Y|X)$, where CE is conditional entropy, ME is marginal entropy, X is the 2D flowset array, and Y is the flowset labels array. This results in $RMI(X,Y) = MI(X,Y)$. Because of the amount of memory they take up compared to all the other alternatives, these RMI scores are used to determine which features are most significant to the user.

IV. RESULTS AND DISCUSSIONS

The proposed method was trained and evaluated on malware-generated network traffic from the CTU-13 and MalRec datasets. Then, it was compared to Ensemble CNN-RNN (ECR), Ensemble CNN-DBN (ECD), Deep Neural Network (DNN), and Deep Belief Network (DBN). The performance metrics used were accuracy, precision, recall, and F-score. The scikit-learn package, Python, and C# were used on a 512 GB RAM server with 32 CPU cores.

TABLE I. DATASET SPECIFICATIONS

Dataset	Parameter	Value
Malrec	Malware recorded	66,301
	Hashing	MD5
	Network activity	PCAP form
CTU-13	Total recordings	13 captures or scenarios

The initial tests used MalRec [8], CTU-13 [22], and other publicly available datasets. The AV Class [16] malware labeling tool was used to categorize samples depending on the malware families to which they belong. To determine the top 25 families based on the highest number of samples, all samples from each family were added up and then ranked. The "other" category had a total of 24,197 malware samples, which included samples from other malware families to eliminate the need for further categorization. To identify the five most frequent malware types, each member of the top malware families was examined to determine which kind of malware it represents. Botnet traffic from the CTU-13 dataset was also used to generate this list.

The proposed classifier was trained with 50 estimators and balanced weights to account for differences in sample sizes between different classes. RMI rates the usefulness of each feature in each feature group and assigns a score. A series of experiments were conducted, in which the number of features varied to identify the minimum characteristics required to maximize classification performance. Based on the findings, the study focused on the top five data characteristics. Port and protocol spoofing and differences in network quality were almost unaffected by the chosen feature set. After identifying these characteristics, the classification performance was analyzed for each malware species. Botnets received an F1 score of up to 94%, which was significantly higher than the random estimate for all other types of malware except ransomware. It was discovered that particular adware and ransomware samples from other categories were mislabeled. As a result, some kinds of malware might share network properties while performing malicious behavior, which is why they were classified as such. When it comes to a specific type of malware, there is nothing particularly noteworthy about these samples. The concept of a classification confidence level was used to remove samples that cannot be discriminated in different malware categories. The classification accuracy improved, but only at the expense of fewer samples being labeled as distinctive. The performance of the proposed classifier can be improved by adjusting the confidence threshold level for classifications. Certain malware requires an additional number of samples to be identified correctly.

Each type of malware has a different confidence threshold, which can be adjusted to achieve the appropriate F1 score. 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 5 shows the classification performance in terms of F1 score with 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.

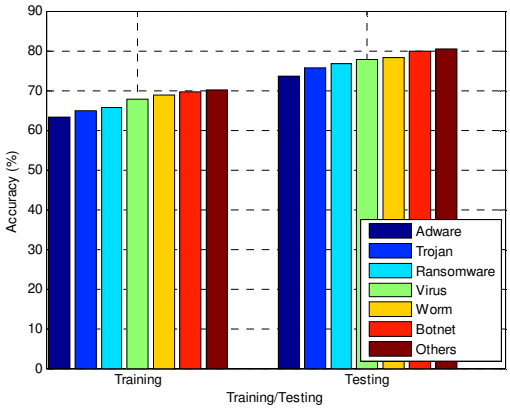

Fig. 2. Accuracy.

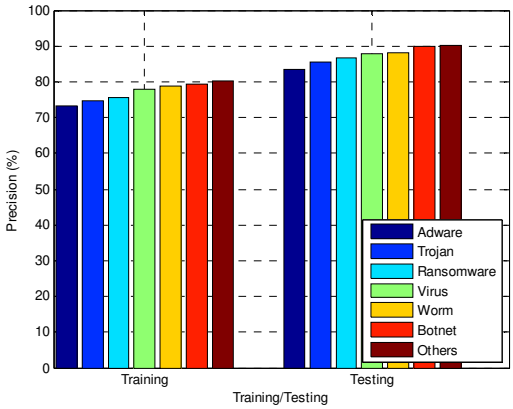

Fig. 3. Precision.

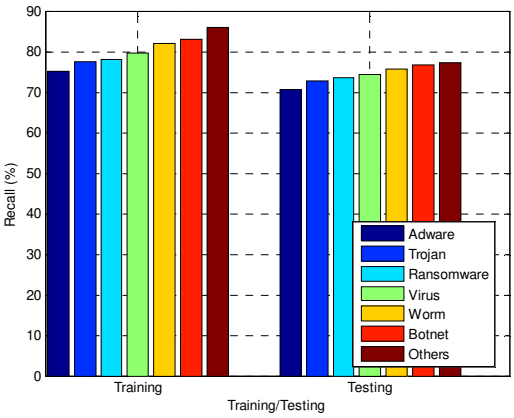

Fig. 4. Recall.

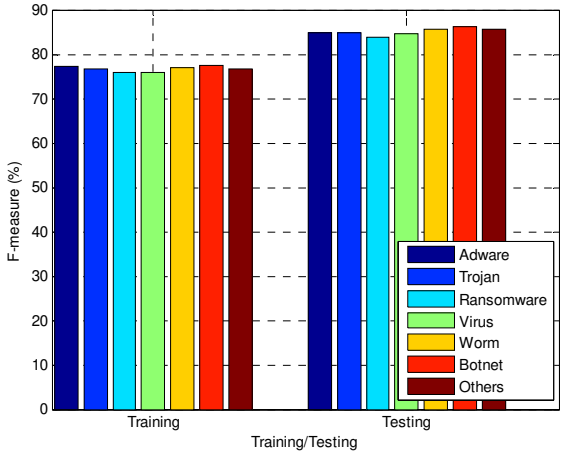

Fig. 5. F1 score.

Following the training on malware datasets, a classifier was further trained in a real-world dataset. When presented with a malware traffic sample, there was a high degree of confidence in its detection. An F1 score of 0.9 effectively reduced the total false positive rates. According to the data, there were approximately 100 adware flowsets per hour and less than 20 ransomware flowsets 20 per hour on average. It was found that worms and viruses containing similar confidence threshold (0.7) can be found after each month, but not more than once. When the confidence level was set to 0.95, fewer than 10 instances of malicious flowsets were estimated to exist. Boltzmann machine algorithms face some limitations, such as:

- There is a problem with the adjustment of weights in the algorithm.
- For probability calculation, more time is needed for collecting statistics.
- Adjustment during simulation is a problem.
- It is significantly slower than the backpropagation method.

Adjusting the confidence threshold level for classifications can lead to better performance for certain types of malware. Different malware categories require different confidence thresholds to achieve the desired F1 score. Using appropriate confidence thresholds, the proposed classifier achieved high classification confidence for specific types of malware flowsets. The results also indicated that the proposed EDRBM model effectively reduced false-positive rates and exhibited high confidence in detecting malware traffic. This study provides insights into the performance of the classifier and the practicality of the approach in real-world scenarios. Overall, this study includes experimental findings and analyses that demonstrate the effectiveness and practicality of the EDRBM model for malware detection in network traffic. The discussion of results provides valuable insights into the performance of the classifier, the impact of confidence thresholds, and its ability to distinguish between different types of malware flowsets.

V. CONCLUSIONS

This study presented the EDRBM method for classifying large-scale network cybersecurity threats. The proposed strategy exhibited remarkable performance in botnet detection and malware classifier techniques. Malicious traffic sets with rates in the 106 per-hour range were discovered on a large-scale network. This is a reasonable response given the incredibly low incidence of malware outbreaks. In contrast with several false positive alerts acknowledged by security operations centers, EDRBM accurately found minimal malicious flowsets. The simulation was carried out on two independent datasets, MalRec and CTU-13, to ensure the consistency of the model. The results showed that EDRBM detects malware with high reliability.

REFERENCES

[1] I. H. Sarker, "Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective," *SN Computer Science*, vol. 2, no. 3, Mar. 2021, Art. no. 154, https://doi.org/10.1007/s42979-021-00535-6.

[2] D. Chen, P. Wawrzynski, and Z. Lv, "Cyber security in smart cities: A review of deep learning-based applications and case studies," *Sustainable Cities and Society*, vol. 66, Mar. 2021, Art. no. 102655, https://doi.org/10.1016/j.scs.2020.102655.

[3] Z. Liu, R. Wang, N. Japkowicz, D. Tang, W. Zhang, and J. Zhao, "Research on unsupervised feature learning for Android malware detection based on Restricted Boltzmann Machines," *Future Generation Computer Systems*, vol. 120, pp. 91–108, Jul. 2021, https://doi.org/10.1016/j.future.2021.02.015.

[4] K. Demertzis, L. Iliadis, E. Pimenidis, and P. Kikiras, "Variational restricted Boltzmann machines to automated anomaly detection," *Neural*

Computing and Applications, vol. 34, no. 18, pp. 15207–15220, Sep. 2022, https://doi.org/10.1007/s00521-022-07060-4.

[5] Z. E. Huma *et al.*, "A Hybrid Deep Random Neural Network for Cyberattack Detection in the Industrial Internet of Things," *IEEE Access*, vol. 9, pp. 55595–55605, 2021, https://doi.org/10.1109/ACCESS.2021.3071766.

[6] A. Thakkar and R. Lohiya, "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*, vol. 28, no. 4, pp. 3211–3243, Jun. 2021, https://doi.org/10.1007/s11831-020-09496-0.

[7] I. Bello *et al.*, "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*, vol. 12, no. 9, pp. 8699–8717, Sep. 2021, https://doi.org/10.1007/s12652-020-02630-7.

[8] C. Gupta, I. Johri, K. Srinivasan, Y. C. Hu, S. M. Qaisar, and K. Y. Huang, "A Systematic Review on Machine Learning and Deep Learning Models for Electronic Information Security in Mobile Networks," *Sensors*, vol. 22, no. 5, Jan. 2022, Art. no. 2017, https://doi.org/10.3390/s22052017.

[9] A. Basit, M. Zafar, X. Liu, A. R. Javed, Z. Jalil, and K. Kifayat, "A comprehensive survey of AI-enabled phishing attacks detection techniques," *Telecommunication Systems*, vol. 76, no. 1, pp. 139–154, Jan. 2021, https://doi.org/10.1007/s11235-020-00733-2.

[10] S. Tsimenidis, T. Lagkas, and K. Rantos, "Deep Learning in IoT Intrusion Detection," *Journal of Network and Systems Management*, vol. 30, no. 1, Oct. 2021, Art. no. 8, https://doi.org/10.1007/s10922-021-09621-9.

[11] M. Veena *et al.*, "A Detection of Malware Embedded into Web Pages Using Client Honeypot," in *Computer Security Threats*, London, UK: IntechOpen, 2020.

[12] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, "Proof of Reputation: A Reputation-based Consensus Protocol for Blockchain Based Systems," in *Proceedings of the 1st International Electronics Communication Conference*, Okinawa, Japan, Apr. 2019, pp. 131–138, https://doi.org/10.1145/3343147.3343169.

[13] C. X. Zhang, J. S. Zhang, N.-N. Ji, and G. Guo, "Learning ensemble classifiers via restricted Boltzmann machines," *Pattern Recognition Letters*, vol. 36, pp. 161–170, Jan. 2014, https://doi.org/10.1016/j.patrec.2013.10.009.

[14] G. Gu, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation," in *Proceedings of the 16th USENIX Security Symposium*, Boston, MA, USA, Aug. 2007, pp. 167–182.

[15] V. Oujezsky, T. Horvath, and V. Skorpil, "Modeling botnet C&C traffic lifespans from NetFlow using survival analysis," in *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, Vienna, Austria, Jun. 2016, pp. 50–55, https://doi.org/10.1109/TSP.2016.7760827.

[16] N. Kheir and C. Wolley, "BotSuer: Suing Stealthy P2P Bots in Network Traffic through Netflow Analysis," in *Cryptology and Network Security*, Paraty, Brazil, 2013, pp. 162–178, https://doi.org/10.1007/978-3-319-02937-5_9.

[17] J. François, S. Wang, R. State, and T. Engel, "BotTrack: Tracking Botnets Using NetFlow and PageRank," in *Networking 2011*, Valencia, Spain, 2011, pp. 1–14, https://doi.org/10.1007/978-3-642-20757-0_1.

[18] P. Amini, R. Azmi, and M. Araghizadeh, "Botnet Detection using NetFlow and Clustering," *Advances in Computer Science: an International Journal,* vol. 3, no. 2, pp. 139–149, Mar. 2014.

[19] K. Bartos, M. Sofka, and V. Franc, "Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants," in *Proceedings of the 25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 807–822.

[20] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, San Jose, CA, USA, Dec. 2010.

[21] M. Z. Rafique and J. Caballero, "FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors," in *Research in Attacks, Intrusions, and Defenses*, Rodney Bay, St. Lucia, Oct. 2013, pp. 144–163, https://doi.org/10.1007/978-3-642-41284-4_8.

[22] B. A. AlAhmadi and I. Martinovic, "MalClassifier: Malware family classification using network flow sequence behaviour," in *2018 APWG Symposium on Electronic Crime Research (eCrime)*, San Diego, CA, USA, Feb. 2018, pp. 1–13, https://doi.org/10.1109/ECRIME.2018.8376209.

[23] A. Mohaisen, A. G. West, A. Mankin, and O. Alrawi, "Chatter: Classifying malware families using system event ordering," in *2014 IEEE Conference on Communications and Network Security*, San Francisco, CA, USA, Jul. 2014, pp. 283–291, https://doi.org/10.1109/CNS.2014.6997496.

[24] W. G. Alheadary, "Controlling Employability Issues of Computing Graduates through Machine Learning-Based Detection and Identification," *Engineering, Technology & Applied Science Research*, vol. 13, no. 3, pp. 10888–10894, Jun. 2023, https://doi.org/10.48084/etasr.5892.

[25] A. Alshutayri, "Fraud Prediction in Movie Theater Credit Card Transactions using Machine Learning," *Engineering, Technology & Applied Science Research*, vol. 13, no. 3, pp. 10941–10945, Jun. 2023, https://doi.org/10.48084/etasr.5950.

[26] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale NetFlow analysis," in *Proceedings of the 28th Annual Computer Security Applications Conference*, Orlando, FL, USA, Sep. 2012, pp. 129–138, https://doi.org/10.1145/2420950.2420969.

[27] W. Ali, G. Wang, K. Ullah, M. Salman, and S. Ali, "Substation Danger Sign Detection and Recognition using Convolutional Neural Networks," *Engineering, Technology & Applied Science Research*, vol. 13, no. 1, pp. 10051–10059, Feb. 2023, https://doi.org/10.48084/etasr.5476.

[28] T. Yi, X. Chen, Y. Zhu, W. Ge, and Z. Han, "Review on the application of deep learning in network attack detection," *Journal of Network and Computer Applications*, vol. 212, Art. no. 103580, Mar. 2023, https://doi.org/10.1016/j.jnca.2022.103580.