Intrusion Detection System based on Network Traffic using Deep Neural Networks

Dimitra Chamou, Petros Toupas, Eleni Ketzaki,
Stavros Papadopoulos, Konstantinos M. Giannoutakis,
Anastasios Drosou, Dimitrios Tzovaras
Information Technologies Institute (ITI)
Center for Research & Technology Hellas (CERTH)
Thessaloniki, Greece
{dimicham,ptoupas,eketzaki,spap,kgiannou,drosou,Dimitrios.Tzovaras}@iti.gr

***Abstract*—Nowadays, the small-medium enterprises security against cyber-attacks is a matter of great importance and a challenging area, as it affects them financially and functionally. Novel and sophisticated attacks are emerging daily, targeting and threatening a large number of businesses around the world. For this reason, the implementation and optimization of the performance of Intrusion Detection Systems have attracted the interest of the scientific community. The malicious behavior detection in terms of DDoS and malware cyber-threats using deep learning methods constitutes an extended and the most important part of this paper. The experimental results for the real-time intrusion detection system showed that the proposed model can achieve high accuracy, and low false positive rate, while distinguishing between malicious and normal network traffic.**

***Index Terms*—Cybersecurity, Intrusion Detection System, Deep Neural Networks, DDoS Detection, Malware Detection**

I. Introduction

The cybersecurity attainment, the data protection and the secure communication are considered essential because of the rapid increment in Internet applications and their use by most of the Internet' s users. At the same time, increased exposure to more sophisticated cyber-threats has been observed over the Internet and computer networks, in the academic and industry digital world, especially in Small-Medium Enterprises (SME), [1], with financial costs. For this reason, the study and the ongoing development of a cybersecurity intrusion detection technology are particularly timely and necessary, operating as the first line of defense, to prevent and tackle intrusion threats, as new security issues are revealed.

The network intrusion detection systems have a primary role in the system reinforcement and defense against ever-increasing malicious behaviors by monitoring and analyzing the network traffic in real-time. The main purpose of IDS is to detect effectively cyber-attacks, i.e. suspicious or abnormal behavior, which violates the system security and exposes the network users to danger, and take the necessary measures to protect the system. Network traffic analysis divides traffic packets into network flows and uses them to monitor the network of the system, [2], [3].

The researchers' attention focuses on implementing IDS to achieve effective solutions against intruders and attacks, [4]. So far, in the research field of IDSs, among other proposed taxonomies, [5], detection techniques are categorized as either signature-based, [6], anomaly-based, [7], or a hybrid combination of both. Compared to the signature-based, the anomaly-based shows a significant difference in identifying novel attacks (zero-day-attack), [4], thereat the proposed model rely on anomaly-based detection.

The machine learning and the deep learning in the cyber-security industry are constantly expanding and are a growing field of research with innovative technologies. Soft computing techniques are based on neural networks to enhance the DDoS detection system performance, especially in case the information about the attack traffic is insufficient and knowledge-based approach based on the history of previous DDoS attacks to develop a defense solution, [8], [9], [10], [11], [12]. The widespread use of machine learning extends to the fields of prediction, classification and estimation, notably in the field of cybersecurity.

The purpose of this paper is to implement an anomaly based intrusion detection system using flow-based statistical data and taking advantage of a deep learning model. The proposed method detects and classifies DDoS and malware types of attacks and normal traffic, separately in binary classification for each type of attack. This paper proposes two deep neural networks (DNN) with multiple fully connected layers (Multi-Layer Perceptron) in order to classify the network traffic of an SME into normal or malicious for DDoS and malware threats, respectively. The proposed solution was implemented under the auspices of FORTIKA H2020 project, which is an ongoing EU-funded project for cyber-security, [13].

The structure of this paper is as follows. Section II describes the related work regarding the intrusion detection systems using machine learning techniques, for malware and DDoS detection. Section III, analyzes the implementation and the training processes of the anomaly detection for both DDoS and Malware cyber-threats. Section IV provides the experimental results and Section V a brief discussion and finally, Section VI concludes the paper and presents future work.

978-1-7281-1016-5/19/$31.00 ©2019 IEEE

II. Related Work

The need to explore new IDS techniques and to improve the existings performance against novel cyber threats, such as DDoS and malware, has drawn the researchers attention. An extended work has been presented in order to address security issues and fulfill vulnerability gaps.

Jiang, et al., [14], presented a hybrid detection model to detect four varieties of ALDDoS attacks. Since they have studied system' s behavior and features of ALDDoS attacks, they combined traffic and user behavior features. They aimed to improve the accuracy of detection and reduce the time and complexity of training user behavior model, using time windows. The experiments conducted in CICDS2017 dataset for DDoS (ALDDoS) detection attacks, by applying three layers Back Propagation Neural Network (BPNN) for the classification. They compared their model with traffic-based and hybrid KNN model and achieved to succeed DDoS attacks detection with accuracy, precision, recall and F1 rate of 99%.

Aksu, et al., [15], conducted a comparative study using 3 machine learning algorithms. They used CICIDS2017 dataset and the supervised machine learning algorithms Support Vector Machine (SVM), K Nearest Neighbor (KNN) and Decision Tree (DT) for binary classification between DDoS and benign examples. They used Fisher Score algorithm in order to reduce the dataset dimension and select the most appropriate features, from 80 features to 30, and non-related features were eliminated. The exported performance measurements, like accuracy, recall, precision and F-score metrics, shown that for the Decision Tree method, evaluation scores did not change, in contrast with KNN' s accuracy was increased and SVM' s was decreased.

Zhou and Pezaros, [5], conducted an analysis using common supervised machine learning algorithms for anomaly flow-based detection. They implemented different algorithms, such as Random Forest classifier, Gaussian Naive Bayes classifier, Decision Tree classifier, Multi-layer Perceptron (MLP) classifier, K-nearest neighbours classifier, Quadratic discriminant analysis classifier to the CIC-AWS-2018 dataset, comparing their performance using the performance criteria of precision, recall, F1 score, and time cost.

Roopak, Tian and Chambers, [16], carried out a comparative analysis between different deep learning algorithms and machine learning algorithms. They have implemented four different classifications deep learning models as Multilayer Perceptron (MLP), CNN, LSTM, CNN+LSTM and they compared them with machine learning algorithms, such as Bayes and Random Forest. The evaluation was done on balanced CICIDS2017 dataset for detection of DDoS attack with binary classification. Based on the performance metric, accuracy, precision and recall, the model with the highest accuracy of 97.16% was CNN+LSTM for both deep and machine learning algorithms on the dataset used.

Faruki et al., [17], proposed a method that is effective against code obfuscation and repackaging. These techniques are widely used to evade anti-virus signature and to propagate unseen variants of known malware. The results demonstrated robust detection of variants of known malware families. Their method created variable length signature and compared it with a signature database using fuzzy logic techniques. Their main goal was to detect unseen and zero-day samples of known malware.

Huang et al., [18], attempted to explore the possibility of detecting malicious applications in the Android operating system. They analyzed the required and requested permission for an Android application using machine learning algorithms on three data sets. Four commonly machine learning algorithms including AdaBoost, Naive Bayes, Decision Tree, and Support Vector Machine, were used to evaluate the above performance. Their experimental results detected more than 81% of malicious samples.

Shabtai et al., [19], proposed Andromly, a behavior-based Android malware detection system. They classified the application as normal or malware based on continuously monitoring specific features and patterns that indicate the device state such as battery level, CPU consumption, etc. while it is running and then applied different machine learning algorithms to discriminate between malicious and benign applications.

Sanz et al., [20], presented PUMA, a method for detecting malicious Android applications through machine learning techniques by analyzing the extracted permissions from the application itself. Their methods can be used as a first step before other more extensive analysis, such as a dynamic analysis.

The study contribution is the implementation of a novel anomaly detection solution, achieving promising results in binary classification between normal and malicious network traffic, by using a deep learning model with multiple fully connected layers (Multi-Layer Perceptron). A more detailed explanation of the proposed implementation is given in Section III.

III. Methodology

A. DDoS Detection

The DDoS input data depend on the choice of the detection method for the method specification. Plus, it is essential to provide features that describe information regarding the network traffic and features that will give information for the detection process. The features that provide information for the network traffic, are the source IP address, the destination IP address and the traffic volume that can be measured directly from the network and the resource entropy that can be calculated and provide the information rate of the traffic. For the DDoS process detection these features are demanded as input data.

Figure 1 illustrates the architecture of the proposed approach for Distributed Denial of Service (DDoS) detection. The proposed model is comprised of three modules, namely 1) Raw Data extraction, 2) Feature Vector Representation and 3) Classification using Deep Neural Networks.

The Raw Data extraction module monitors the system activity and stores to a database the network traffic exchanged between the system and other systems, or within modules of the same system. Specifically, the Raw Data extraction module monitors packets, their size, and the time in which they were sent/received. The data stored in the database are used as input by the Feature Vector Representation module. Particularly, the data are aggregated in specific time windows, and the total number of packets and bytes exchanged within each window is measured. The size of the window is a parameter of the model. Given the aggregated information and the window size, seven features are extracted for each time window, namely, i) Number of Packets, ii) Average packet size, iii) Time Interval Variance, iv) Packet Size Variance, v) Number of Bytes, vi) Packet Rate, and vii) Bit Rate. Finally, the produced feature vectors are given as input to a Deep Neural Network for classification. The proposed network is composed of a total of five layers, including fully connected layers, and non-linear activation functions (ReLU). The output of this network is a binary classification, representing either the DDoS attack, or normal traffic existence. The Deep Neural Network is trained on instances of both DDoS attacks and normal traffic in order to learn to differentiate between the two.

B. Malware Detection

The malware detection for operational systems was an inspiriting motivation for the proposed methodology that has been developed.

The requirements process needed for malware defense of an organization require multiple layers of defenses, [21]. The defense should consider both the enclave boundary area, which concerns the organization network interaction with the Internet focusing on the firewalls and IDS, and the computing environment area, that concerns the inner network of the organization.

Figure 2 shows the architecture of the proposed approach for malware detection. It is comprised of three components, 1) Raw Data extraction, 2) Feature Vector Representation and 3) Classification using Deep Neural Networks.

The Raw Data extraction component monitors the system calls for each application on the monitored system. It has a specific list of system calls to monitor, related to file system calls, process calls, network calls, and memory calls. The output of this component is a system calls sequence for each application running in the system. These sequences of calls are given as input to the second component, namely, the Feature Vector Representation. For each application call sequence, N-grams of consecutive system calls are considered, for N=4. Each unique N-gram is encoded in a unique position in the feature vector. The size of the final feature vector is equal to the number of unique N-gram for all the applications. The value of each position of the vector is equal to the number of appearances of the corresponding N-gram. Finally, these feature vectors are given as input to a Deep Neural Network for classification. The proposed network is comprised of a total of 5 layers, including fully connected layers, and non-linear activations (ReLU). The output of this network is a binary classification, representing either malware or benign application. The Deep Neural Network is trained on instances of both malware and benign applications in order to learn to differentiate between the two.

C. Implementation

The DNN training was achieved through the use of a large amount of data collected from various public sources, [22], [23], [24], on the web, where many different cyber-attacks of malicious network traffic, such as DDoS and malware attacks, along with normal network traffic were recorded.

The data collection was captured with TShark tool, which is a terminal-based version of Wireshark. This tool analyzes network packets by reading them from a previously saved pcap file. The developed tool using TShark, converts the network traffic captured in pcap files (binaries), into human readable json files (netflow v9 relevant format) containing important information about the flows in the network. Thus, it is possible to filter the information, as part of the pre-processing, included in pcap files, and so to reduce and eliminate the noise and the redundant information of the data and store only the information considered important, in order to extract the DNN input feature vector. The next step of the feature extraction process is to parse the produced json files, and create the feature vectors for the DNN input feature vector.

The output of feature extraction was unbalanced with the range of values for each one of the seven extracted features is different and the deviations vary from very small to very big intervals. It was necessary to scale the range of all features into a smaller and common factor for all features. The dataset balance was achieved through the normalization technique in order to improve the training of the deep learning method. Finally, the features' rate varied between 0 and 1 by using the normalization of Python scikit learn library, [25].

In the next part of pre-processing the dataset was split into three individual datasets: training, validation, and testing set. The DNN was trained using only the training set. The validation set was used in order to fine-tune the hyper-parameters of the neural network, like learning rate, dropout rate, choose between multiple optimizers, find the best value for the number of layers and the number of nodes in each layer, etc. Moreover, the test set does not interfere at any point with the training process, so the DNN evaluation can be performed using this set and considering the evaluation result to be quite safe. The model evaluation was performed with a 10 fold cross-validation method.

The proposed DNN architecture, consists of the input layer with 8 nodes, followed by a first hidden layer with 50 nodes, a second hidden layer with 20 nodes, a third hidden layer with 10 nodes, a fourth hidden layer with 20 nodes, and the output layer. "ReLU" activation function, [26] has been applied in the output of the four hidden layers, while "Sigmoid" activation function has been applied in the output layer. Also, the "lecun uniform" initializer, [27], was used for the weights initialization in each one of the layers.

The training of the DNN "Adam" optimizer, [28], was achieved with the back-propagation process using the default values of the optimizer. The "Binary crossentropy" was used

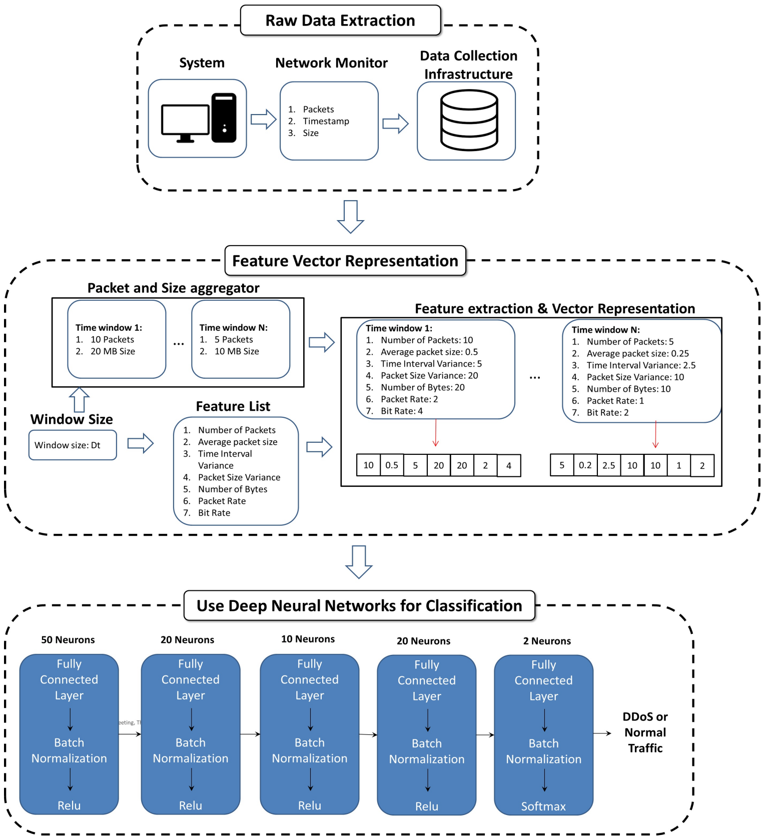


Fig. 1: DDoS architecture

as the loss function for classification and the DNN has been training for 30 epochs. All of the above-mentioned functions and components, which were used, are part of the Python Tensorflow library, [29]. As mentioned above, the validation set has been used to keep tracking of the overall training process and its performance.

IV. Experimental Results

During the training process of the DNN, a graph indicating training and validation accuracy, and one about training and validation loss over all epochs, have been created to gain a better intuition about the DNN performance. These graphs are presented in Figure 3a, 3b, 4a and 4b respectively.

Since this model consists of multiple fully connected layers, most of the calculations are based on the dot product, which is a combination of multiplications and additions (MACCs). Based on the matrix multiplication, the non linear transformation and the weight sharing of the neural network, the total number of floating point operations for a single pass through the entire network (feed-forward) are 24.4K for the multiplication operations and 9.54K for the add operations.

The evaluation of the DNN was based on the testing set and it is shown in Table I.

The DNN prediction (feed-forward) was also implemented and accelerated in FPGA achieving a speedup of 40 compared to the dual-core ARMv7 Processor 2.64GHz CPU implementation in Xilinx Zynq Platform. The accelerator design is

TABLE I: DNN evaluation for DDos and malware

	DDoS	**Malware**
Test Set Accuracy	99.79%	99.44%
True Positives	151737	4030
False Positives	208	102
True Negatives	30091	30324
False Negatives	178	92

implemented with Vivado HLS(v2016.4). This implementation is built on the VZynq 7 Series Xilinx FPGA SoC. The feed-forward process is being executed in a total of 100 clock cycles and a AXI4S width of 64b.

The experimental implementation in FPGA was crucial since the threat prediction in an SME is taking place in real-time, so the faster the decision can be exported from the DNN the better for the SME, because it can come up faster with a mitigation action regarding the threat. The average execution time required for a successful detection of a threat example is about 3 milliseconds in FPGA, as well as in an ARM processor for just one example. The great differentiation has been when the number of inputs increases and the time required in ARM processor for execution is increasing by far.

V. Discussion

The proposed approach allows detecting efficiently cyber-threats, such as DDoS and malware attacks in SME's workstations. Comparing these results to previous work is difficult,

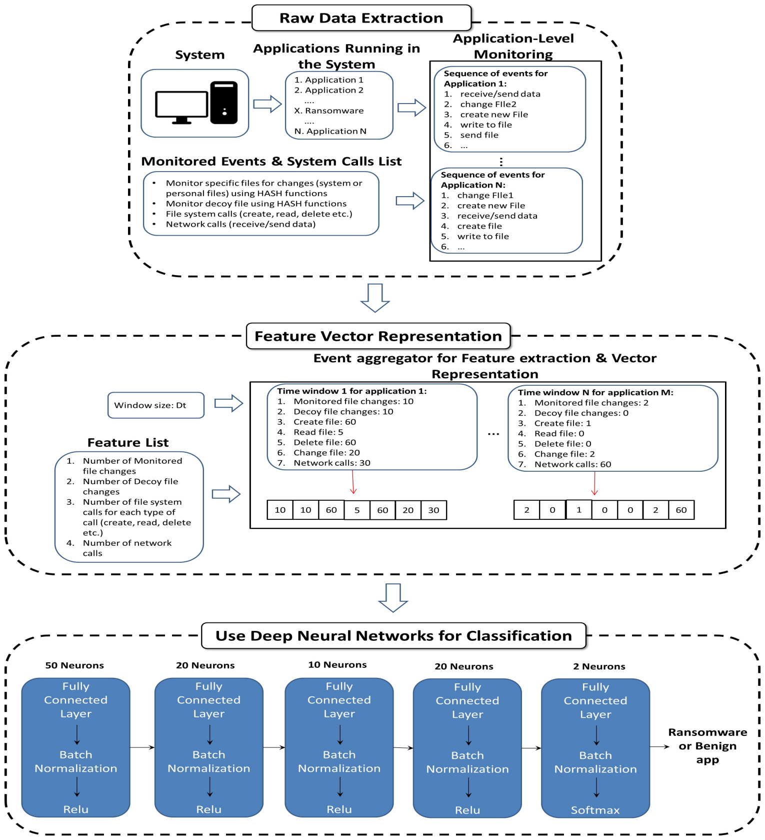
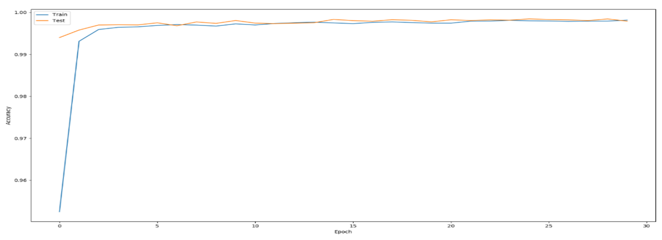
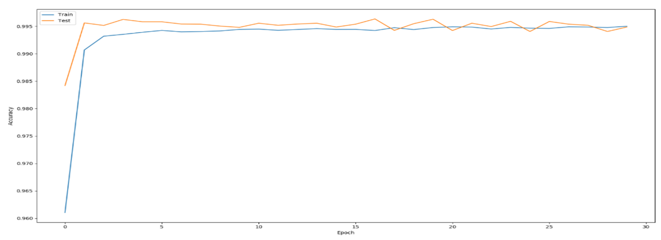

Raw Data Extraction
System
Applications Running in the System
1. Application 1
2. Application 2
....
X. Ransomware
....
N. Application N
Application-Level Monitoring
Sequence of events for Application 1:
1. receive/send data
2. change FIle2
3. create new File
4. write to file
5. send file
6. ...
⋮
Sequence of events for Application N:
1. change FIle1
2. create new File
3. receive/send data
4. create file
5. write to file
6. ...
Monitored Events & System Calls List
• Monitor specific files for changes (system or personal files) using HASH functions
• Monitor decoy file using HASH functions
• File system calls (create, read, delete etc.)
• Network calls (receive/send data)
Feature Vector Representation
Event aggregator for Feature extraction & Vector Representation
Window size: Dt
Feature List
1. Number of Monitored file changes
2. Number of Decoy file changes
3. Number of file system calls for each type of call (create, read, delete etc.)
4. Number of network calls
Time window 1 for application 1:
1. Monitored file changes: 10
2. Decoy file changes: 10
3. Create file: 60
4. Read file: 5
5. Delete file: 60
6. Change file: 20
7. Network calls: 30
...
Time window N for application M:
1. Monitored file changes: 2
2. Decoy file changes: 0
3. Create file: 1
4. Read file: 0
5. Delete file: 0
6. Change file: 2
7. Network calls: 60
10 10 60 5 60 20 30
2 0 1 0 0 2 60
Use Deep Neural Networks for Classification
50 Neurons
20 Neurons
10 Neurons
20 Neurons
2 Neurons
Fully Connected Layer
Batch Normalization
Relu
Softmax
Ransomware or Benign app

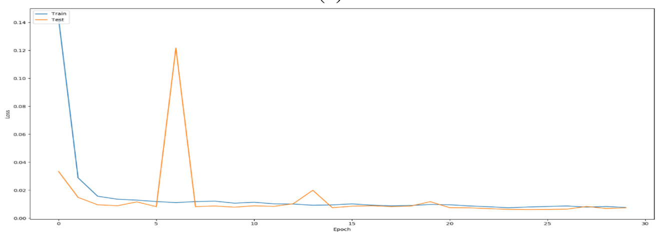
Fig. 2: Malware Architecture



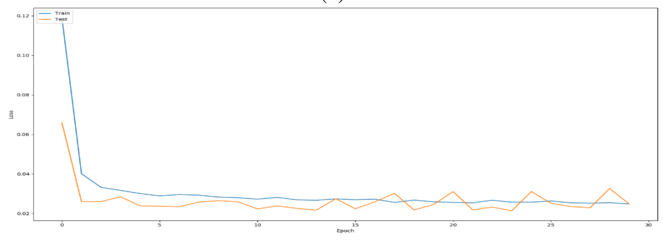
(a)



(a)



(b)



(b)

Fig. 3: (a) DDoS accuracy change during training for 30 epochs (b) DDoS loss change during training for 30 epochs

Fig. 4: (a) Malware accuracy change during training for 30 epochs (b) Malware loss change during training for 30 epochs

since other studies use different datasets and the feature extraction and the deep learning procedures are not fully specified. In comparison with the use of Multi-Layer Perceptron model, this specific approach has succeeded better accuracy and performance metrics in the field of DDoS detection than other detection methods, [16] with 97,16% accuracy. Hence, this method represents an initial and economic cyber-security solution for SMEs provide the best accuracy, with a high detection rate and a low false positive rate. Plus, results showed that the proposed solution can be further extended in future, besides the detection part, to the cyber-threat prevention and be further compared.

The advantage of the specific approach is the FPGA use, so the execution time and the resource consumption of the system can be limited. The major drawback of this solution is that the training of the DNN is based on binary classification, so it is not feasible to determine the type of the attack.

Concluding the discussion, let us denote that the proposed architecture and the implementation is not limited or specific to the binary classification, but also can be extended in the field of multi-classification. In this respect, the anomaly detection model will have the ability to detect the type of multiple cyber-threats.

VI. Conclusions and Future Work

In this work, an anomaly detection system combined with deep learning methods was presented. A specialized solution targeting the needs of an SME to defend against cyber-attacks, such as DDoS and malware was introduced. The proposed solution used network flows in order to classify the cyber-threats into binary categories (normal or malicious) using multiple fully connected layers (Multi-Layer Perceptron). The classification achieved using flow-based statistical data. Thus, the anomaly detection system implementation was achieved with high accuracy rate of 99.79% for DDoS detection and 99.44% for malware detection.

In future, there is an intention to expand existing work and try novel deep learning algorithms in a complete, rich, up-to-date and well-formed dataset for training, validation and evaluation, in order to extract additional results. Furthermore, it is intended to use system logs data metrics, collected by system monitoring agents, for expanding the features of malware detection. Additionally, the problem will be approached by the aspect of multi-class classification in order to detect and classify more categories of cyber-threats.

Acknowledgment

This work has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation, with Title H2020-FORTIKA "cyber-security Accelerator for trusted SMEs IT Ecosystem" under grant agreement no 740690.

References

[1] C. Paulsen, "Cybersecuring small businesses," *Computer*, vol. 49, no. 8, pp. 92–97, Aug 2016.

[2] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network intrusion detection: Based on deep hierarchical network and original flow data," *IEEE Access*, vol. 7, pp. 37 004–37 016, 2019.

[3] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," *IEEE transactions on information forensics and security*, vol. 8, no. 1, pp. 5–15, 2012.

[4] A. Boukhamla and J. Coronel, "Cicids2017 dataset: Performance improvements and validation as a robust intrusion detection system testbed." *International Journal of Information and Computer Security*, 09 2018.

[5] Q. Zhou and D. Pezaros, "Evaluation of machine learning classifiers for zero-day intrusion detection - an analysis on CIC-AWS-2018 dataset," *CoRR*, vol. abs/1905.03685, 2019. [Online]. Available: http://arxiv.org/abs/1905.03685

[6] H. Holm, "Signature based intrusion detection for zero-day attacks: (not) a closed chapter?" in *2014 47th Hawaii International Conference on System Sciences*, Jan 2014, pp. 4895–4904.

[7] V. Jyothsna, V. V. Rama Prasad, and K. Munivara Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, pp. 26–35, 08 2011.

[8] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," 10 2010, pp. 408–415.

[9] C. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "Nice: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 198–211, July 2013.

[10] N. Z. Bawany, J. A. Shamsi, and K. Salah, "Ddos attack detection and mitigation using sdn: Methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, Feb 2017. [Online]. Available: https://doi.org/10.1007/s13369-017-2414-5

[11] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *16th International Conference on Advanced Communication Technology*, Feb 2014, pp. 167–171.

[12] J. Wang, R. C. . Phan, J. N. Whitley, and D. J. Parish, "Augmented attack tree modeling of distributed denial of services and tree based attack detection method," in *2010 10th IEEE International Conference on Computer and Information Technology*, June 2010, pp. 1009–1014.

[13] E. Markakis, Y. Nikoloudakis, G. Mastorakis, C. X. Mavromoustakis, E. Pallis, A. Sideris, N. Zotos, J. Antic, A. Cernivec, D. Fejzic, J. Kulovic, A. Jara, A. Drosou, K. Giannoutakis, and D. Tzovaras, "Acceleration at the edge for supporting smes security: The fortika paradigm," *IEEE Communications Magazine*, vol. 57, no. 2, pp. 41–47, February 2019.

[14] J. Jiang, Q. Yu, M. Yu, G. Li, J. Chen, K. Liu, C. Liu, and W. Huang, "Aldd: A hybrid traffic-user behavior detection method for application layer ddos," 08 2018, pp. 1565–1569.

[15] D. Aksu, S. Ustebay, M. Aydin, and T. Atmaca, *Intrusion Detection with Comparative Analysis of Supervised Learning Techniques and Fisher Score Feature Selection Algorithm*, 09 2018, pp. 141–149.

[16] M. Roopak, G. Yun Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," 01 2019, pp. 0452–0457.

[17] P. Faruki, V. Ganmoor, V. Laxmi, M. Gaur, and A. Bharmal, "Androsimilar: robust statistical feature signature for android malware detection," 11 2013, pp. 152–159.

[18] C.-Y. Huang, Y.-T. Tsai, and C.-H. Hsu, *Performance Evaluation on Permission-Based Detection for Android Malware*, 01 2013, vol. 21, pp. 111–120.

[19] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, ""andromaly": A behavioral malware detection framework for android devices," *J. Intell. Inf. Syst.*, vol. 38, pp. 161–190, 02 2012.

[20] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. Bringas, and G. Alvarez, *PUMA: Permission Usage to Detect Malware in Android*, 01 2013, vol. 189, pp. 289–298.

[21] R. Rehman, G. Hazarika, and G. Chetia, "Malware threats and mitigation strategies," 01 2005.

[22] "Datasets overview," https://www.stratosphereips.org/datasets-overview, accessed: 2019-07-22.

[23] "A source for pcap files and malware samples..." https://www.malware-traffic-analysis.net/index.html, accessed: 2019-07-22.

[24] "Mcfp dataset," https://mcfp.weebly.com/mcfp-dataset.html, accessed: 2019-07-22.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[26] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.