Ilkogretim Online - Elementary Education Online, 2021; Vol 20 (Issue 5): pp. 7401-7408
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.05.839

Enhancing the Cyber Security Intrusion Detection based on Generative Adversarial Network

Prabakaran P, Assistant Professor (Sr.G), KPR Institute of Engineering and Technology, Coimbatore
Dr.R.S. Mohana, Associate Professor, Department of Computer Science and Engineering, Kongu Engineering College, Erode
Dr. S. Kalaiselvi, Assistant Professor (SG), Department of Computer Technology, Kongu Engineering College, Erode

Abstract- We will introduce a survey of profound learning methods, data settings and a comparative analysis for the identification of cyber security intrusions. In particular, we study intrusion detection systems using profound learning approaches. Deep learning (DL) approaches: if the research is based on DL for intrusion systems. DL approaches, if machine learning for intrusion detection systems (IDS) is included in the report. Evaluation of approaches to deep learning: it shows whether DL approaches for IDS are evaluated. We research success in two classification categories under two new real-world data sets for each model, namely, CSE-CIC-IDS2018 and the Bot-IoT dataset in order to deeply learn each model. Furthermore, we use the main metrics of success to evaluate the effectiveness of several methods: false alarm rate, detection rate and Accuracy. This shows that the Generative Adversarial Network (GAN) produces much better outcomes than other state-of-the-art approaches.

Keywords:- **Intrusion detection systems, Machine learning, deep learning, Comparative study and Generative Adversarial Network**

I. INTRODUCTION

The request for cyber security and defense against different kinds of cyber attacks has steadily increased in recent days. The key explanation for this is the popularity of the IT, the massive development of networks and the enormous amount of applications used for personal and commercial uses by individuals and groups. Non-reparable disruption and financial damages in large-scale Networks resulted in cyber attacks, including Denial-of-Service (DoS) attack [1], malware [2] or unauthorized admission [1]. In May 2017, for example, a ransomware virus led to a loss of 8 billion dollars to many organisations and sectors, including banking, medical, oil, and universities. On average, other figures have reported that data infringement invokes USD 3.9 million and USD 8.19 million in the United States to the affected organization [4]. Therefore, the request for cyber security and defense against different categories of cyber-attacks is rising day after day in the cyber world, according to the needs of today.

Usually, a cyber security framework includes a network security scheme and a security scheme. While many schemes such as firewall have been developed in order to tackle Internet-based cyber-attacks, the IDS system is better able to resist external attacks on the computer network [5]. An IDS therefore mainly seeks to detect malicious communications and computer systems for protection of different forms of malicious network. The traditional solutions, for example firewalls, cannot do the tasks properly[6-8]. An IDS detects cyber-assault activity on a network and tracks and analyses day-to-day operations on a network to detect security hazards or attacks, such as DoS. An IDS also enables unauthorized system conduction, such as illegal access or alteration and disruption to be identified, determined and identified [9–11]. To promote system security, it is therefore important to classify numerous kinds of cyber-attacks and anomalies within a network and create effective IDS that play a vital part in the current network safety.

II. RELATED WORKS

CNNs and the RNNs used to recognise malware by Kolosnjaji et al.[12]. The list of sequences in the kernel of the API is translated by one-hot encoding into binary vectors. One hot encoding is a method that makes it easier for machine learning to store categorical data. These data are used to train a CNN and RNN DL algorithm. This model achieves 89,4% accuracy, 85,6% accuracy, and a reminder of 89,4%.

Tobiyama and others[13] have developed a malware detector that fed the API to RNN for functional extraction by calling time series data. These characteristics are then turned into an image, which is classified as malicious or natural by a CNN. The RNN uses an LSTM and the CNN has two layers of overlapping and two overlapping. Two completely linked layers are followed. Although the data set was relatively limited, an AUC of 0.96 could be achieved.

By preprocessing portable executable (PE) files for the n-gram extraction, Ding, Chen, and Xu [14] generated a DBN using the machine language operating codes. Three layers were concealed in the DBN. It contains 3000 benign, 3000 malicious and 10,000 unlabeled files. When pre-trained with unlabeled results, the DBN model outpaces sVMs, decision trees. DBN's highest performance accuracy was 96.7%.

The new technique for making malware signs using DBN was developed by David and Netanyahu[15], which was formed on undefined information and classified malware using DBNs and denoising autoencoders. Software signatures have been created using the logs of a sandbox, using n-grams to process them by taking the 20,000 most popular images that appear on a malware only and generating the 20,000-specific vector to decide if an image is seen. This data is then used to pre-train an 8-layer DBN with autoencoders that denoise them. A vector of 30 numbers was the final malware signature. The network has been trained on C4 Security's 1800 malware example with six different kinds of malware - three hundred for each kind of malware. The malware classifier was developed using an SVM, using 1200 malware instances used for training after the features were computed. The findings were positive and showed 98.6% precision.

Woodbridge et al. [16] have created a technique of detecting malicious DGAs domain names associated with C2 server malware contact. Earlier, most work was done with handcrafted features in this field. But Woodbridge et al. only used the domain name and gritnodes in an RNN to categorise domain names by treatment and embedding into the sheet, followed finally a classification layer, for each character within the domain name to be treated. A TPR of 98% and an FPR of 0.1% were demonstrated to be excellent performance.

III. PROPOSED SYSTEM

Two new traffic datasets (CSE-CICIDS2018) [18] and Bot-IoT[17] for the planned experiments have been used for the project. The framework is now being implemented. The statistics for training and test attacks in both datasets are summarized in Tables 1 and 2. The test is carried out with the tensor flux and graphics processing unit in Google Colaboratory1 in python 3 (GPU). The specifics of the experimental IDS practise are shown in Fig.1. The system contains specifically of four stages: (1) phase data sets, (2) stage preprocessing, (3) phase training and (4) phase of testing.

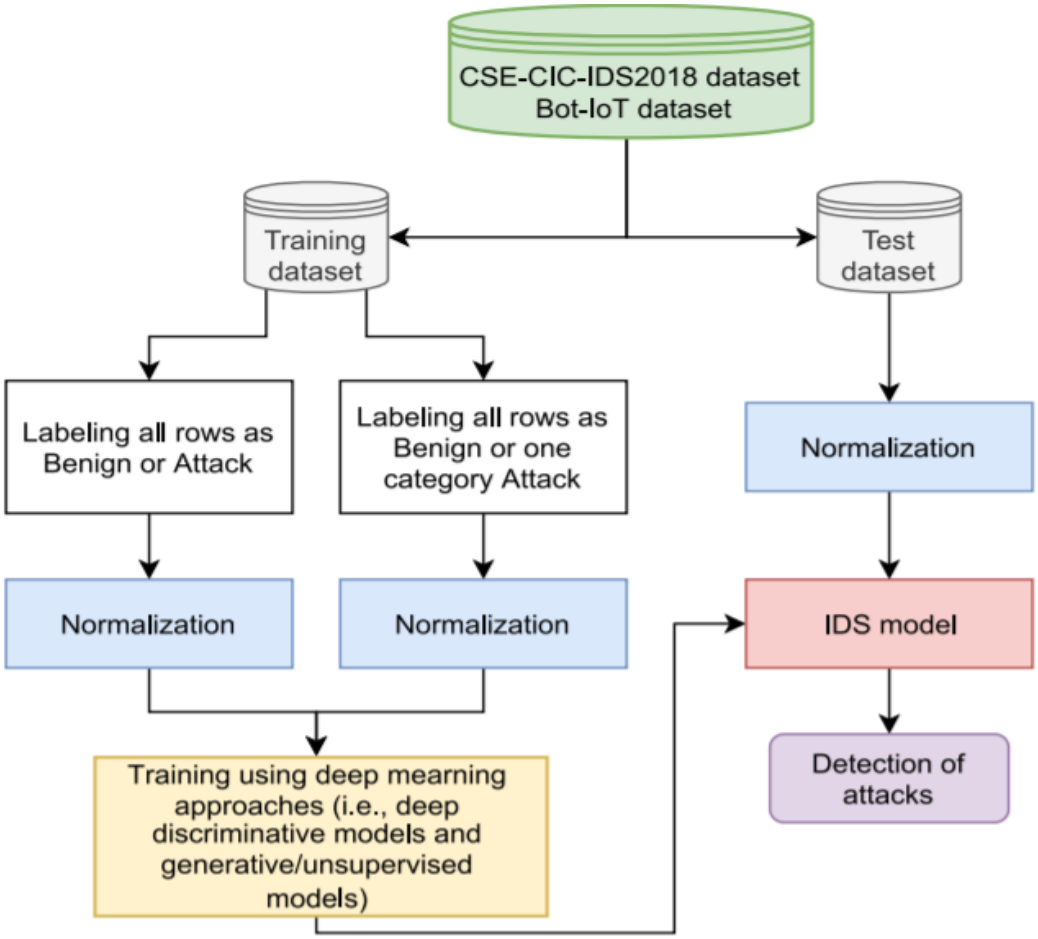

Fig.1. Flowchart of the IDS methodology.

3.1. Dataset:-

(i) Bot-IoT dataset

This datasets comprises over 72 thousand thousand documents, including attacks for DDoS, Keylogging and Data Exfiltration. Koroniotis etal. [19] proposes that the Bot-IoT dataset is a new dataset compared with previous ones for the IoT setting. To simulate IoT devices network activity, the authors used the Node-Red tool. The dataset is based on the MQTT protocol, the lightweight protocol for communication between machine and machine (M2M). However, five IoT scenario are used in the test bed: weather station, smart refrigerator, moving lights, remote garage door, and intelligent thermostat.

Table 1 Attack types in CSE-CIC-IDS2018 dataset.

Category	Attack type	Flow count	Training	Test
Brute-force	SSH-Bruteforce	230	184	46
	FTP-BruteForce	611	489	122
Web attack	Brute Force -XSS	187,589	7504	1876
	Brute Force -Web	193360	15,469	3867
	SQL Injection	87	70	17
DoS attack	DoS attacks-Hulk	466664	18,667	4667
	DoS attacks-SlowHTTPTest	139890	55,956	13,989
	DoS attacks-Slowloris	10990	4396	1099
	DoS attacks-GoldenEye	41508	16,603	4151
DDoS attack	DDOS attack-HOIC	686012	27,441	6860
	DDOS attack-LOIC-UDP	1730	1384	346
	DDOS attack-LOIC-HTTP	576191	23,048	5762
Botnet	Bot	286191	11,448	2862
Infilteration	Infilteration	161934	6478	1620
Benign	/	12,697,719	50,791	12,698
Total	/	15,450,706	231,127	57,782

(ii) CSE-CIC-IDS2018 dataset

The CSE and the Canadian Cybersecurity Institute are proposing this dataset. this knowledge (CIC). The data collection CSE-CIC-IDS2018 consists of seven separate attack scenarios, such as Heartbleed, Brut-force, DoS, DDoS, Botnet and Web attacks and infiltration. The CICFlowMeter device is used to extract 80 network flow functionality from network traffic generated, similarly to the CICDS2017 data set.

Table 2 Attack categories in Bot-IoT dataset

Category	Attack type	Flow count	Training	Test
BENIGN	BENIGN	9543	7634	1909
Information gathering	Service scanning	1,463,364	117,069	29,267
	OS Fingerprinting	358275	28,662	7166
DDoS attack	DDoS TCP	19,547,603	1,563,808	390,952
	DDoS UDP	18,965,106	1,517,208	379,302
	DDoS HTTP	19771	1582	395
DoS attack	DoS TCP	12,315,997	985,280	246,320
	DoS UDP	20,659,491	1,652,759	413,190
	DoS HTTP	29706	2376	594
Information theft	Keylogging	1469	1175	294
	Data theft	118	94	24
Total	/	73,370,443	5,877,647	1469413

3.2. Data-set pre-processing

The dataset CSE-CIC-IDS2018 comprises 15 450 706 rows, each of which consists of 10 files, with 80 functions per row. The file contents are listed as follows:

- Folder 1 ”Thur-20-02-2018”: DDOS attack-LOICHTTP (576,191 rows) and benign traffic (SQL Injection (34 rows).
- Folder 2 ”Wed-21-02-2018”: DDOS attack-HOIC (686,012 rows), benign traffic (360833 rows).
- Folder 3 ”Thur-22-02-2018”: Brute Force -XSS (79 rows), Brute Force-Web (249 rows), benign traffic (1048213 rows).
- Folder 4 ”Fri-23-02-2018”: It contains Brute Force -XSS (151 rows), Brute Force-Web (249 rows), SQL Injection (286391 rows) and benign traffic (762,384 rows).
- Folder 5 ”Thu-15-02-2018”: DoS attacksGoldenEye (41,508 rows), DoS attacks-Slowloris (10,990 rows), and benign traffic (996077 rows).
- Folder 6 ”Friday-16-02-2018”: DoS attacksSlowHTTPTest (139,890 rows), DoS attacks-Hulk (466,664 rows), and benign traffic (442120 rows).
- Folder 7 ”Wed-28-02-2018”: Infiltration attack
- Folder 8 ”Wed-14-02-2018”: FTP-BruteForce (183,360 rows), SSH-Bruteforce (187,589 rows), and benign traffic (667,626 rows).
- Folder 9 ”Thu-01-03-2018”: It contains Infiltration attack (68871 rows) and benign traffic (544200 rows).

❖ Folder 10 ”Fri-02-03-2018”: Botnet attack DoS attacksGoldenEye (41,508 rows.

The Bot-IoT dataset comprises over 72,000,000 records for 74 files with 46 characteristics for each row. We use the version of the training and testing version planned by Koroniotis et al.[19], which comprises 5% of the total collection of data. We put the files to one JSON document with PyMongo 3.7.2 to build a subset of training and testing.

3.3. Classification

It is one of the common generative models in this framework proposed by GAN Network. Goodfellow et al. launched GANs in 2014. The GAN functioning concept is based on theory of games. A GAN operates on a high level by battling between networks of generators and discriminators. The discrimination against individuals is responsible for distinguishing among the original data set and the generator data. The generator's job is to trick the discriminator into thinking it is true. Approximating high dimensional data has been shown to GANs successfully.

The GAN framework can be trained and created. The generative model $G(z|y)$ maps in the original formulation random input from a $p_z(z)$ noise distribution and condition to the target data distribution p data (x). A discriminatory $D(x|y)$ model then calculates, given the condition, the likelihood of a data point being part of the destination. The generator and the discriminator are simultaneously trained in order to maximize the chance that the correct label is assigned to "true" data (from p data(x) and "falsified" data (from $p_{data}(x)$) while the generator attempts to maximize the chance of samples for "real" data being generated by discrimination. In other words, the following minimax game with the value function $V(D,G)$ is played by both models:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \epsilon p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \epsilon p_z(z)}[\log(1-D(G(z|y)|y)] \quad (1)$$

Practical preparation, this means that if you have a correct sample of real data $s_c = (x_c, y_c)$, have a correct status and fake sample $s_w = (x_w, y_w)$, fake data (s) and a negative log-like loss is calculated as a discriminatory sample (s) and you have a correct sample s) of false data (arrested and arbitrary).:

$$\mathcal{L}_D = -\log D(x_c|y_c) - \log(1 - D(x_w|y_w)) \quad (2)$$

While the generator loss, for an analogous s_w sample, is:

$$\mathcal{L}_G = -\log D(x_w|y_w) \quad (3)$$

In our case, each matrix's condition vector is the average of the EMD characteristics for each class matrix and all subjects. We assume that the generator and the discriminator can catch this repairability and comply if the EMD function is properly trained to generate distinguishable features for various classes. In Fig.2 both D and G are convolutionary networks.

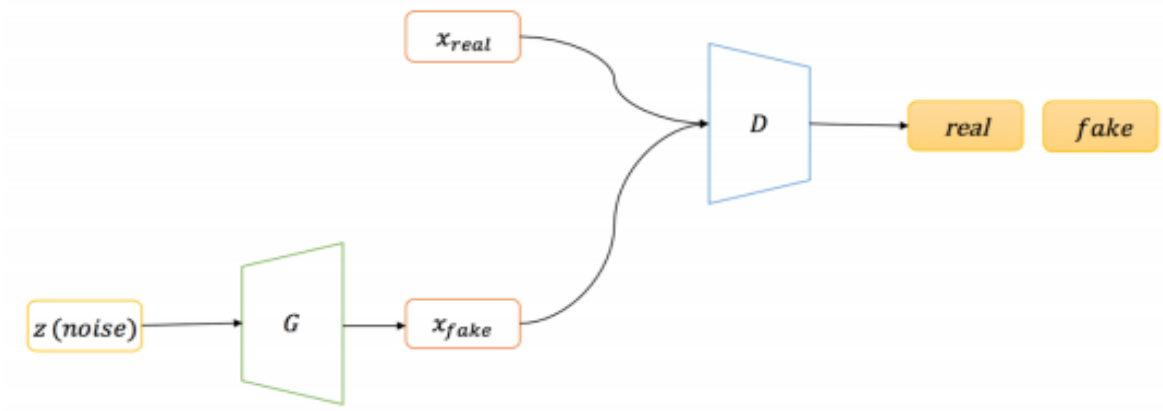

Fig.2. The architecture of GANs.

Condition y is attached in the generator to a random noise Vector z, and the concatenated input to an output mark is proven by a cascade of transposed turns. The condition y of the input matrix is spatially repeated and appended to the feature maps from second- to last convolution layer, which are to be used in the final probability calculation after passing through a few convolutionary layers that decrease the size of feature maps.

We change the discrimination loss function provided in eq. 2 during learning. During learning. Instead of the discriminator being trained on the basis of the proper conditions and arbitrary false matrix (forcing the discriminator to learn how to discriminate between the real matrix under the right circumstances and the real matrix under the wrong conditions without any clear monitoring), Hence, given a correct sample $s_c = (x_c, y_c)$ and wrong samples $s_w = x_c, y_w$ and $s_{w2} = (x_w, y_w)$, the discriminator loss becomes:

$$\mathcal{L}_D = -\log D\ (x_c|y_c) - \quad \log\big(1 - D(x_c|y_w)\big) - \log(1 - \quad D(x_w|y_w)) \qquad (4)$$

IV. RESULTS AND DISCUSSION

4.1. Performance Metrics

Performance Metrics can be categorized using four indexes based on the exact, reminder, error rate and accuracy measurement. The effectiveness of the offered solution is dependent on the coherence of a particular course and the particulars of the rest of the documentation, some documents and most categories noted. The identification of precision or reminder value that specifies the sensitivity parameter is thus successful in reflecting the classifier's precision. These are defined as equations for calculating these steps:

$$Precision = \frac{True\_Positive}{True\_Positive + False\_Positive} \qquad (5)$$

$$Recall = \frac{True\_Positive}{True\_Positive + False\_Negative} \qquad (6)$$

$$Error\ rate = 2 \times \frac{precision \times recall}{precision \times recall} \qquad (7)$$

$$Accuracy = \frac{True\_Positive + True\_Negative}{True\_Positive + True\_Negative + False\_Positive + True\_Negative} \qquad (8)$$

4.2. Qualitative Analysis

From the table 3, it is clearly shows that the highest accuracy is GAN classifier, i.e. 93.21% of accuracy.

Table 3: Comparative Analysis of Proposed Classifier

Parameters Metrics	**Classifier**	**Parameters (%)**
Accuracy	GAN	93.c4
Precision		93.89
Recall		92.60
Accuracy	CNN	90.41
Precision		89.65
Recall		90.82

4.3. Quantitative analysis

In this division, the proposed system performance is compared with other existing methods in terms of accuracy and MSE for overall dataset, which is given in Table 4.

Table 4: Comparative Analysis of Proposed Classifier

Methodology	**Parameter Metrics**	
	Accuracy (%)	**Error rate**
DT	89.5	18.14
ANN	78	25.63

MLP	81.72	22.44
KNN	85.14	19.42
Auto encoder	88.32	16.70
CNN	90.82	12.6
GAN	**93.34**	**5.12**

Accuracy is the ratio of true negative and true positive and true negative, and false negatives and false positives. This determines how well an occurrence measurement is evaluated. From the Table 4, it is stated that DT, CNN, KNN and MLP achieved nearly 89% to 90% of overall accuracy for CBIR query results. However, GAN achieved very low classification accuracy and high RMSE value than any other existing techniques. In this research study, proposed techniques are included with the GAN classifier to improve the query classification results and the results proved that proposed GAN achieved 97.34% of overall accuracy and less Error rate i.e. 5.12. The next section will discuss the performance of proposed technique with various classifiers.

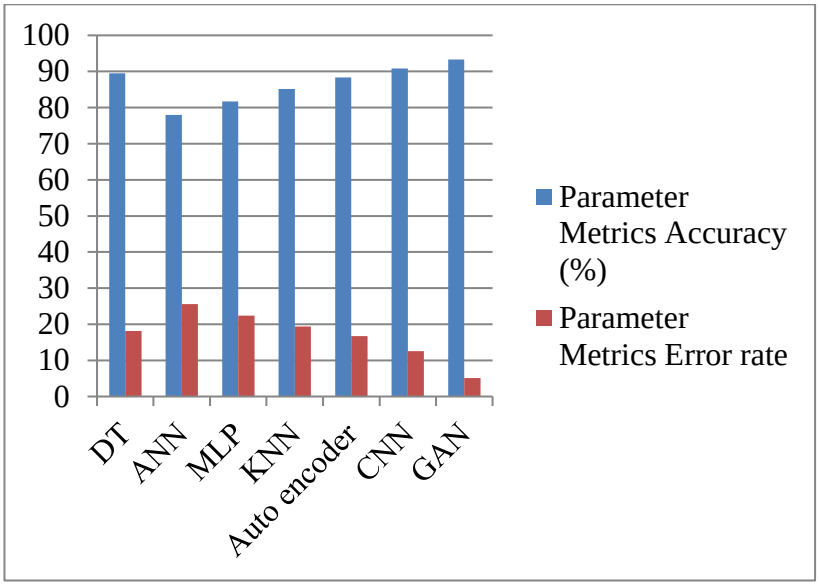

Fig. 3. Comparative Analysis.

V. CONCLUSION

In this article we carried out a comparative analysis on intrusion detection machine learning methods such as DT, ANN, MLP, KNN CNN and generative/unsupervised models. The two new datasets, the CSE-CICIDS2018 dataset and the Bot-IoT dataset, combine these approaches to generate three significant data performance displays as, accuracy and detector rates. The result shows that Generative Adversarial Network provides much better results compared to other existing systems.

REFERENCES

[1]. Sun, N.; Zhang, J.; Rimba, P.; Gao, S.; Zhang, L.Y.; Xiang, Y. Data-driven cybersecurity incident prediction: A survey. IEEE Commun. Surv. Tutor. 2018, 21, 1744–1772.
[2]. Dainotti, A.; Pescapé, A.; Ventre, G. Worm traffic analysis and characterization. In Proceedings of the 2007 IEEE International Conference on Communications, Glasgow, UK, 24–28 June 2007; pp. 1435–1442.
[3]. Qu, X.; Yang, L.; Guo, K.; Ma, L.; Sun, M.; Ke, M.; Li, M. A Survey on the Development of Self-Organizing Maps for Unsupervised Intrusion Detection. Mob. Netw. Appl. 2019.
[4]. IBM Security Report. Available online: https://www.ibm.com/security/data-breach (accessed on 20 October 2019).

[5]. Tsai, C.F.; Hsu, Y.F.; Lin, C.Y.; Lin, W.Y. Intrusion detection by machine learning: A review. Expert Syst. Appl. 2009, 36, 11994–12000.
[6]. Mohammadi, S.; Mirvaziri, H.; Ghazizadeh-Ahsaee, M.; Karimipour, H. Cyber intrusion detection by combined feature selection algorithm. J. Inf. Secur. Appl. 2019, 44, 80–88.
[7]. Tapiador, J.E.; Orfila, A.; Ribagorda, A.; Ramos, B. Key-recovery attacks on KIDS, a keyed anomaly detection system. IEEE Trans. Dependable Secur. Comput. 2013, 12, 312–325.
[8]. Tavallaee, M.; Stakhanova, N.; Ghorbani, A.A. Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 2010, 40, 516–524.
[9]. Milenkoski, A.; Vieira, M.; Kounev, S.; Avritzer, A.; Payne, B.D. Evaluating computer intrusion detection systems: A survey of common practices. ACM Comput. Surv. (CSUR) 2015, 48, 1–41.
[10]. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutor. 2015, 18, 1153–1176.
[11]. Xin, Y.; Kong, L.; Liu, Z.; Chen, Y.; Li, Y.; Zhu, H.; Gao, M.; Hou, H.; Wang, C. Machine learning and deep learning methods for cybersecurity. IEEE Access 2018, 6, 35365–35381.
[12]. Kolosnjaji, B.; Zarras, A.; Webster, G.; Eckert, C. Deep learning for classification of malware system call sequences. In Proceedings of the Australasian Joint Conf. on Artificial Intelligence, Hobart, Australia, 5–8 December 2016; pp. 137–149.
[13]. Tobiyama, S.; Yamaguchi, Y.; Shimada, H.; Ikuse, T.; Yagi, T. Malware detection with deep neural network using process behavior. In Proceedings of the IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, USA, 10–14 June 2016; Volume 2, pp. 577–582.
[14]. Ding, Y.; Chen, S.; Xu, J. Application of Deep Belief Networks for opcode based malware detection. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 3901–3908.
[15]. David, O.E.; Netanyahu, N.S. Deepsign: Deep learning for automatic malware signature generation and classification. In Proceedings of the 2015 International Joint Conference Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8.
[16]. Woodbridge, J.; Anderson, H.S.; Ahuja, A.; Grant, D. Predicting domain generation algorithms with long short-term memory networks. *arXiv*, 2016; arXiv:1611.00791.
[17]. Bot-IoT Dataset. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/ cybersecurity/ADFA-NB15-Datasets/bot_iot.php. last accessed 30 May 2019.
[18]. [CSE-CIC-IDS2018 Dataset. https://www.unb.ca/cic/datasets/ids-2018.html. last accessed 30 May 2019.
[19]. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Fut. Gener. Comput. Syst. 2019;100:779–96.