PcapGAN: Packet Capture File Generator By Style-Based Generative Adversarial Networks

1st Dowoo Baik
Agency for Defense Development
Daejeon, Republic of Korea
arizona95@add.re.kr

2nd Yujin Jung
Agency for Defense Development
Daejeon, Republic of Korea
ooparts67@add.re.kr

3rd Changhee Choi
Agency for Defense Development
Daejeon, Republic of Korea
changhee84@add.re.kr

***Abstract*—After the advent of GAN technology, many varied models have been studied and applied to various fields such as image and audio. However, in the field of cyber data, which has the same issue of data shortage, the research on data augmentation is insufficient. To solve this problem, we propose PcapGAN that can augment pcap data, a kind of network data. The proposed model includes an encoder, a data generator, and a decoder. The encoder subdivides network data into four parts. The generator generates new data for each part of the data. The decoder combines the generated data into realistic network data. We demonstrate the similarity between the generated data and original data, and validation of the generated data by increased performance of intrusion detection algorithms.**

***Index Terms*—Cyber Data, Network Data, Generation, GAN, Encoder, Insturion Detection Algorithms**

I. Introduction

Recently, there has been a rapid technical improvement in producing high quality images using generative methods, including Generative Adversarial Networks (GAN) [1], [2]. Various attempts have been made to find a way to improve the performance of image classification using the generated images [3], [4]. However, since cyber data are atypical and combines a variety of information, it is difficult to apply the latest deep learning technology immediately. Thus, in contrast to the rapid development of technology with respect to the image data, technological progress on the generation of cyber data is very slow. There was an attempt to augment a packet data using GAN in [5], but the generated packet data could not be read by the network packet analysis tool such as Wireshark, and attack packets could not be generated even though anomaly dataset was replicated. Attempts have been made to analyze the network flows and create new network flows using IP2Vec, which maps IPs of hosts with similar network behavior into similar vectors [6]. However, IP2Vec could not make a new type of network flow graph because the IP vector was static, and it simply interprets the packet interval time data as only a continuous information, so that it could not analyze patterns of automated cyber attacks. Furthermore, for our best knowledge, there is no method that can quantitatively evaluate the generated packets. In this paper, we propose a generative method of PcapGAN, which is inspired by various botnet analysis methods for anomaly packets [7]–[10] and is not against this methods. PcapGAN can generate packet data that can be analyzed by a network packet analysis tool, and the data generating process is visible so that the quality of the generated data can be judged by the human eye like an image data. Through PcapGAN, we generated an augmented dataset based on MACCDC 2012 [11] and GTISK dataset [12]. After that, we converted the generated data and original data into KDDcup99 [13] format. We conducted the experiments to determine whether they could be distinguished using intrusion detection algorithms and visually validate the similarity between the generated data and the original data using Principal Component Analysis(PCA). We also tested whether the performance of intrusion detection algorithms can be increased by the generated data, in purpose of validation. As a result, intrusion detection algorithms' performance improved overall, indicating the validity of the generated data.

II. PcapGan

In network data, abnormal data often have a special pattern such as periodic communication between botnet and command and control server(C&C server) [14], or a specific client's very high dns traffic, so information from the data should be extracted and analyzed appropriately. The architecture of PcapGAN model presented in this paper consists of an encoder, a generator, and a decoder. First, the encoder extracts information from pcap data [15] into appropriate format such as a graph (source IP $\longrightarrow$ destination IP), an image (time interval), and a sequence (layer structure sequence) from network data (pcap data). In order to analyze and generate data reflecting the relationship between hosts in cyber data, a feature that expresses the relation between hosts is needed. In this paper, we propose 'style', a vector value that can effectively represent the relations between hosts such as server-client and C&C server-botnet relationship. In the IP graph, the relationship between hosts is expressed as an edge, and each edge has a style value. Each data generated by encoder is labeled by the edge style, and the generator is designed based on the style. The next step, the Generation step, operates in a hybrid structure in which modern GAN models for each data format extracted by the encoder (graph, image, sequence) generates new data. Finally, the decoder combines the generated information and reconstruct a valid pcap file.

978-1-7281-4550-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMLA.2019.00191

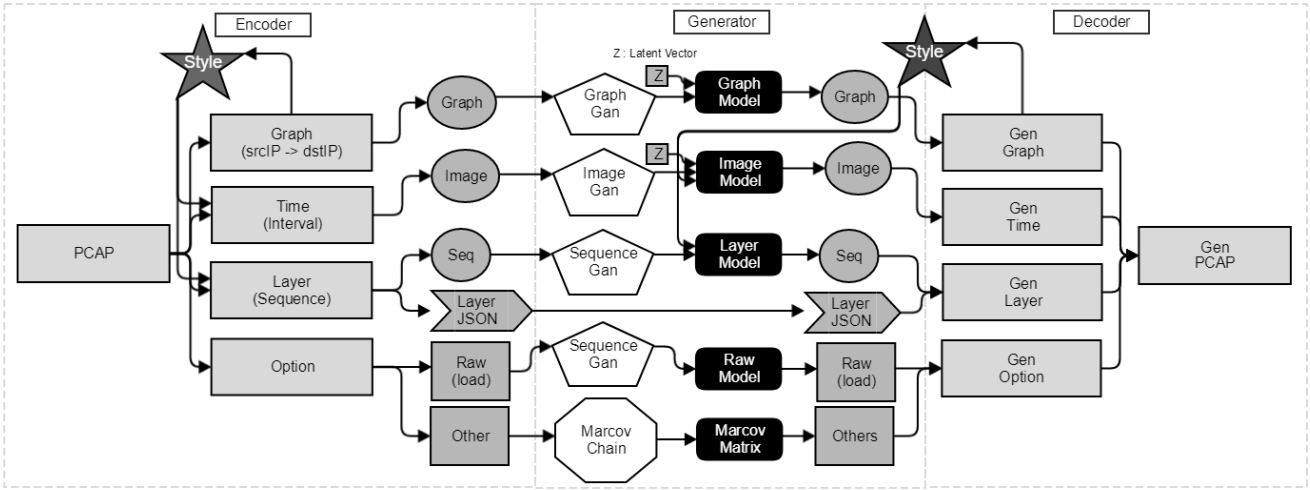


Fig. 1: Overall process of the PcapGan generative method

A. Style-based encoder

IP Graph : Packet data are sent and received between multiple hosts which has it's own IPs each, and the network flow graph shows the flow of packet data, so it is possible to analyze the botnet and C&C relations in the network by the network flow graph [7], [8]. Therefore, in this paper, we first create a network flow graph. By regarding each IP addresses included in the pcap file as nodes, and each packet as directional edges from source IP to destination IP, entire pcap file is transformed into a digraph G. Let $G = (N, E)$, $N = \{v_1, ...v_V\}$, $E = \{e_{ij}\}_{i,j=1}^{V}$, for $v_i, v_j \in N$, let $P(v_i, v_j)$: set of packets received from v_i to v_j. Each edge of digraph G has style vector in list form. Let s_{ij} = style of $e_{ij} \in E$, s_{ij} = [log($|P(v_i, v_j)|$), Degree of v_i+Degree of v_j]. The vector size of style S is 2. Between normal client and server, the amount of packets is small, so that the degree of the node is low. However, between the bot and the target, the amount of packets is large and the degree of the node is high.

Time Image : In [9], [10], which analyzed the network traffic using timestamp information in the packet data, the time interval of packets is represented as an image. This is because the packet interval time contains information such as whether the communication is automatic or human controlled. By expressing the packet interval time as an image, we can see a certain temporal pattern of bots controlled by the command and control server. In this paper, we express time information in the form of image and call this as time image. Let P be the entire packet set and for $p \in P$, $F_t(p)$ be the packet arrival time. For packets $p_n, p_{n+1} \in P(v_i, v_j)$ that arrive in succession, an image spot of the time image is defined as $[(F_t(p_n) - F_t(p_{n+1})) \times \alpha)] + s_{ij}$. The number of channels of the time image is $S + 1$.

Layer Sequence : A packet consists of several layers, and there is a protocol used for each layer. To analyze the combination of these protocols and use it for data generation, we replace the combination of protocols with numbers(ex. Ether/IP/UDP/DHCP : 1) and create a dictionary about these corresponding relationships. The dictionary, let us say Layer Dictionary, is in JSON [16] format. By replacing each packet protocol data into numerous through Layer Dictionary, we make sequential data and call this as layer sequence. In this way, we represent packet's protocol data in the form of sequential data so that the model can learn and generate data reflecting sequencing information among packets.

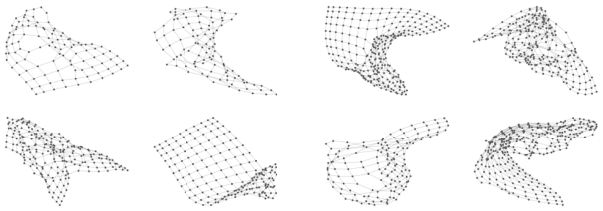
Option Data : Encoder extracts option variable for each protocol of the packet except IP, IPv6, mac, checksum, and id. The option data which have a very simple pattern(ex. Version, ihl, ext) and the load option of the raw layer, which are not simple data, are extracted separately.

B. Style-based generator

IP Graph : IP graph is a sparse matrix in the form of $V \times V \times S$, generated by referring to GraphGan [17]. GraphGAN model takes $V \times V \times batchsize$ array as input, so in this paper, we set style vector's size S as batch size for training. In digraph G, for a given node v_c, the probability of connecting to another node is $p_{true}(v|v_c)$. For Generator $G(v, v_c; \theta_G)$ and Discriminator $D(v, v_c; \theta_D)$, value function $V(G, D)$of the GAN structure is like equation 1.

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{V} (E_{v \sim p_{true}(.|v_c)}[logD(v, v_c; \theta_D)] + E_{v \sim G(.|v_c; \theta_G)}[log(1 - D(v, v_c; \theta_D))]) \quad (1)$$

For the generated graph, like Figure 2, it is given random IP, mac, and IPv6 information for each vertex. The generated graph becomes the network flow graph of packet data to be generated. The style vector generated for each edge becomes input, and the rest part of the generator generates data suitable for each edge.



(a) Original Graph (b) Generated Graph

Fig. 2: IP graph comparison

Time Image : We used a customized StyleGAN model from NVIDIA [1] as illustrated in Figure 3. In StyleGAN, the mapping network maps latent space Z with intermediate latent space ω, and input the intermediate vector $w \in \omega$ into adaptive instance normalization(AdaIN) [18]. In this paper, we use a new intermediate vector, a concatenation of style vector and w. Let's say the style of normal client→server edge be style A, and the style of bot→target edge be style B. As Figure 4 shows, the time images corresponding to style A and style B are generated differently.

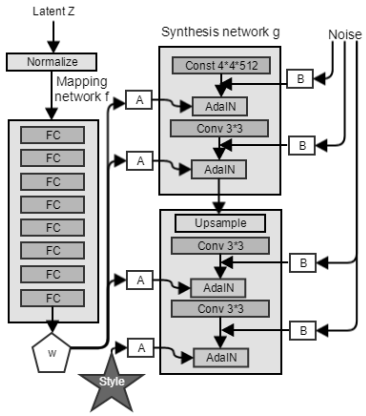
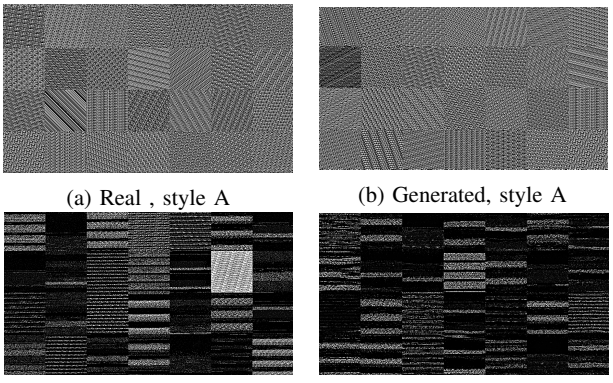


Fig. 3: Time generator model based on style



(a) Real , style A (b) Generated, style A

(c) Real , style B (d) Generated , style B

Fig. 4: Time image comparison (contrast enhancement is applied for visuality))

Layer Sequence : It is inappropriate to generate layer sequence without preprocessing because a pattern in which the same number is repeated continuously and innumerably may occur due to the nature of the network data. Namely, for the sequence of pattern like that, the machine learning model will not be able to remember the previous pattern as it learns. Besides, the sequence information contains the rules of network flow, and these are very detailed rules such that two packets are paired(ARP, LLMNRQuery) or large of number of packets in a row when the TCP flow starts. In order to solve the above issues, this paper proposes a method of preprocessing layer sequence as Algorithm 1.

Algorithm 1 Layer encoder

for $L1, L2 \in$ Layer Sequence **do**
  $G1$ = $L1$ except Raw,Padding,Dot1Q Layer
  $G2$ = $L2$ except Raw,Padding,Dot1Q Layer
  if $G1$ == $G2$ **then**
    grouping $L1, L2$
    append $L1, L2$ to Grouped Sequence[$G1$]
  end if
end for
for $L \in$ Layer Grouped Sequence **do**
  if L == before L **then**
    append L to Encoder Group
  else
    concat L+'@'+$log(NumofEncoderGroup)$ to Layer Grouped Sequence Compress
    Encoder Group to Empty
  end if
  if Cnt $>$ 25 or L has Error Layer **then**
    append LineSwitch to Layer Grouped Sequence Compress
    append Style, Cnt=0
  end if
end for

The data encoded by Algorithm 1 are similar to natural language in that it modifies and describes the data that follows it. Therefore, we express the layer sequence information in the form of sequential data. We used SeqGan [19] model, which can generate the sequential data, and customized the model to create the sequential data labeled with the style vector which is similar to the input style vector. In Algorithm 1, the grouped sequence was created as a Markov chain because it is a sequence of simple patterns consisting of three words which can determine whether there is a raw or padding layer among consecutive layers. When decoding, Algorithm 1 is performed in the reverse order, using the combination of generated data by SeqGAN and grouped sequence.

Option Data : An option is a payload of complex patterns or simply a sequence of identical numbers. The former is augmented with SeqGan [19], and the latter is augmented with a simple model, Markov chain. Since options are dependent on IP nodes, they are not labeled with style.

C. Decoder

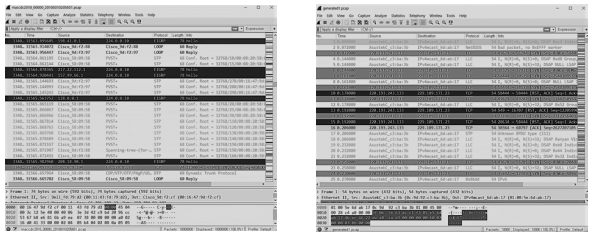
A Pcap file is created by the Decoder using the IP graph generated, the time image generated by edge style of IP graph, the layer sequence, and the option data. Let p_f be the number of all packets to be generated, and $S1_f$ be the first style vectors set of edges of the generated graph. For each edge of the generated IP graph, when the edge style is $[s_1, s_2]$, the number of the generated packets is as shown in Equation 2.

$$p_e = \frac{p_f \times \varepsilon^{s_1}}{\sum_{s1 \in S1_f} \varepsilon^{s1}} \tag{2}$$

The procedure of creating a packet is as follows. First, convert layer sequences into combinations of protocols by referring to the layer dictionary (ex. 1 : [Ether, IP, UDP, DHCP]). Second, for the converted combinations, make packet data for each protocol by using the option data. At this time, if the option contains features such as src, dst, psrc, and pdst, set its value as assigned to the IP node. Third, set the start time of the first packet of each edge randomly, and set the reception time of other packets according to the time interval information of the time image. Sort the packets generated at each edge of the IP graph in chronological order and transform it into a pcap file.

III. Quality of the Generated Pcap File

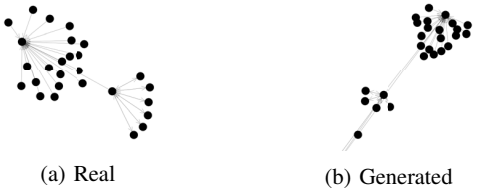
By visualizing the packet data generated with the Pcap-GAN, the quality of the generated pcap file can be visually confirmed. We compared real GTISK packet sample with the generated GTISK packet sample.



(a) Real (b) Generated

Fig. 5: Packet analysis by Wireshark

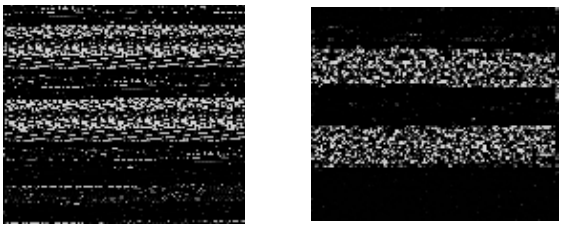
As shown in Figure 5, the generated pcap file can be analyzed with Wireshark, a network packet analyzing tool. We found that the generated pcap file can be analyzed with other analysis tools(ex. Tcpdump, OmniPeak) either. This seems to be possible because the data was generated by splitting and reassembling packets in the pcap file.



(a) Real (b) Generated

Fig. 6: Generated IP graph

As shown In Figure 6, the network flow graphs of real data and generated data are similar.



(a) Real (b) Generated

Fig. 7: Time image generated with style of bot-target edge

Figure 7a is time image of a bot client conducting Battery Drain Attack. It shows a pattern of bots attacking the target server by continuously sending tens of thousands of TCP packets to attack the target server (dense area). Figure 7b shows the generated time interval data, which contains a pattern similar to the original data.

TABLE I: IPv6 Option Data

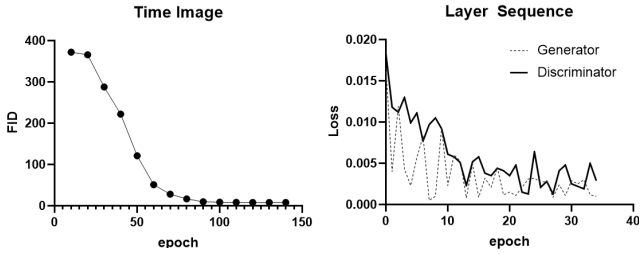
fl	hlim	nh	plen	ver
0	255	58	80	6
0	1	0	36	6
0	255	58	80	6
0	1	17	33	6
0	255	58	32	6
0	1	0	36	6
0	1	17	33	6
0	1	0	36	6
0	1	0	36	6
0	255	58	32	6
0	255	58	32	6
0	255	58	80	6
0	255	58	80	6
0	255	58	80	6

(a) Real

fl	hlim	nh	plen	ver
0	1	0	36	6
0	1	17	33	6
0	1	0	36	6
0	1	0	36	6
0	255	58	32	6
0	255	58	32	6
0	255	58	32	6
0	255	58	80	6
0	255	58	80	6
0	1	0	36	6
0	1	0	36	6
0	255	58	32	6
0	1	0	36	6
0	1	0	36	6

(b) Generated

Table I shows that the PcapGAN generates option data well.



(a) Time image (FID) (b) Layer sequence (Loss)

Fig. 8: The evaluation on the generator of PcapGAN

The figure above is a quantitative evaluation on Generator of PcapGAN. As shown in Figure 8a, as the training progresses,the FID score [20](lower is better) of the time image decreases. Figure 8b that the loss of Generator and Discriminator decreases as the leaning epoch increases in SeqGAN used for generating the layer sequence data.

IV. Evaluation

We evaluated similarity and validity of data generated by PcapGAN experimentally. Namely, we evaluated similarity between the generated data and the original data and how realistic the generated data was. Since the data are cyber data, it is necessary to convert the file format for evaluating the data using intrusion detection algorithms.

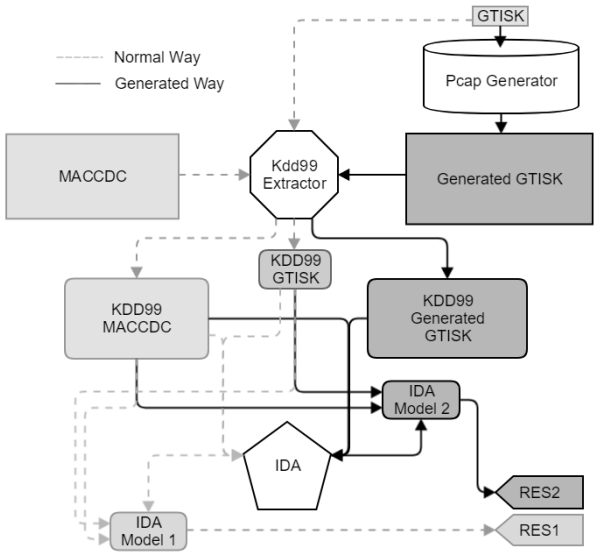


Fig. 9: Evaluation scenario

A. KDD99 extractor and intrusion detection algorithms

As the scenario illustrated in Figure 9, through PcapGAN, we augmented the GTISK dataset (which is cyber attack data) using a model pre-trained with the MACCDC 2012 dataset (which is normal data). We converted the two original data (GTISK, MACCDC2012) and the generated data into KDD form through KDD99 extractor, in order to apply it to the intrusion detection algorithms developed based on KDD99 data format. When converting a file, 28 features as shown in Table II were used, and the meaning of each feature is the same as described in [13]. We call the converted MACCDC data as data A, the converted GTISK data as data B, and the converted generated data (based on the GTISK data) as data C. This experiment is about whether the measurement result of the algorithm is increased by the experimental data, so that the performance and level of the algorithm itself are not discussed in this paper. We tested with algorithms presented in [21]. We transformed string data from the converted dataset into an integer using LabelEncoder class of sklearn, and then normalized it. After that, we calculated accuracy, precision, recall, and f1score value using algorithms of sklearn.

B. Similarity of the original data and the generated data

For the first step of evaluation, in order to prove the similarity, we tested whether the intrusion detection algorithms can distinguish data B and data C. In addition, we used

TABLE II: KDD99 extractor features

1	duration	2	protocol type
3	service	4	flag
5	src bytes	6	dst bytes
7	land	8	wrong fragment
9	urgent	10	count
11	srv count	12	serror rate
13	srv serror rate	14	rerror rate
15	srv rerror rate	16	same srv rate
17	diff srv rate	18	srv diff host rate
19	dst host count	20	dst host srv count
21	dst host same srv rate	22	dst host diff srv rate
23	dst host same src port rate	24	dst host srv diff host rate
25	dst host serror rate	26	dst host srv serror rate
27	dst host rerror rate	28	dst host srv rerror rate

PCA to verify similarity visually. As the result of the first experiment(Table III) shows, the accuracy consistently records similar values around 0.5. This proves that the original data and the generated data are difficult to be distinguished by the intrusion detection algorithm used in this experiment, which shows that the generated data is similar to the original data. Figure 10 also shows that the principal component distributions of the two data are in perfect agreement.

C. Validity of the generated data

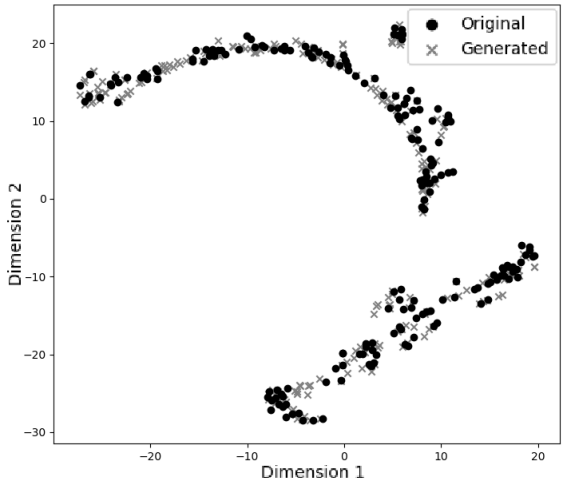


Fig. 10: PCA(n=2) plot of data B, data C

For the second step of evaluation, using intrusion detection algorithms, we generated a classification model by learning to distinguish data A and data B (IDA model 1), and generated another model based on data A and data C (IDA model 2). After that, we conducted experiments to distinguish data A and data B by using the two generated models to check the accuracy. If the result of IDA model 2 is better than IDA model 1, it means that the generated GTISK dataset is valid. Table IV and Table V show the result of IDA model 1 and IDA model 2 each. As can be seen from Table V, except KNN, the performance of each intrusion detection algorithms are 2-4% improved. Figure 11 shows the results of this experiment visually.

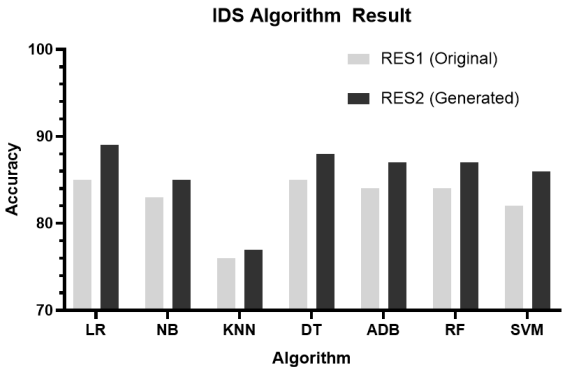


Fig. 11: Result of intrusion detection algorithm on the classification models. RES1 is classification result by IDA model 1, RES2 is classification result by IDA model 2.

TABLE III: MACCDC - generated MACCDC classification

algorithm	LR	NB	KNN	DT	Adb	RF	SVM
accuracy	0.500	0.499	0.490	0.476	0.499	0.476	0.501
precision	0.497	0.499	0.489	0.474	0.499	0.475	0.494
racall	0.054	0.993	0.488	0.450	0.635	0.476	0.016
f1score	0.098	0.664	0.489	0.461	0.559	0.476	0.031

TABLE IV: Data A - data B classification by IDA model 1

algorithm	LR	NB	KNN	DT	Adb	RF	SVM
accuracy	0.853	0.837	0.762	0.855	0.847	0.846	0.820
precision	0.989	0.988	0.989	0.999	0.995	0.999	0.994
racall	0.821	0.923	0.921	0.912	0.912	0.767	0.770
f1score	0.897	0.955	0.954	0.954	0.952	0.868	0.868

TABLE V: Data A - data B classification by IDA model 2

algorithm	LR	NB	KNN	DT	Adb	RF	SVM
accuracy	0.898	0.853	0.776	0.882	0.874	0.873	0.861
precision	0.957	0.940	0.909	0.998	0.988	0.908	0.957
racall	0.987	0.997	1.000	0.996	0.999	0.998	0.987
f1score	0.972	0.968	0.954	0.997	0.994	0.951	0.972

V. Conclusion

In this paper, we present a new method for augmenting pcap data, using a model designed based on the style. Unlike the image data, the cyber data usually contain sophisticated data such as checksum, and it need an adequate number in the designated location or else analysis tools such as Wireshark cannot even recognize the data. Therefore, it is very difficult to generate the cyber data by existing methods of augmenting images, but PcapGAN presented in this paper not only solves these problems, but also produces high quality pcap files that sophisticated analysis such as network flow graph analysis and timestamp analysis is available. We expect that the method presented in this paper will inspire and generate ideas for future cyber data generators. As a room for improvement that we have not yet tried in this paper, more advanced pcap file generators may emerge through different ways of designing the style vectors and new ways of associating the style vectors with the option data. Although the detection rate by intrusion detection algorithms on the generated data increased, we only indirectly assessed the accuracy of each GAN discriminator model, and did not cover the overall and quantitative evaluation of the generated packet data. We hope that the quantitative evaluation system for GANs will be developed so that the quality of data generated by the method presented in this paper can be evaluated.

References

[1] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[2] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[3] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*. IEEE, 2018, pp. 117–122.

[4] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification," *Neurocomputing*, vol. 321, pp. 321–331, 2018.

[5] Å. Kamphaug, "A generative adversarial approach for packet manipulation detection," Master's thesis, Universitetet i Agder; University of Agder, 2018.

[6] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, 2019.

[7] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, "Botnet detection using graph-based feature clustering," *Journal of Big Data*, vol. 4, no. 1, p. 14, 2017.

[8] S. Ranjan, "Machine learning based botnet detection using real-time extracted traffic features," Mar. 25 2014, uS Patent 8,682,812.

[9] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[10] C. Yin, "Towards accurate node-based detection of p2p botnets," *The Scientific World Journal*, vol. 2014, 2014.

[11] "Capture files from the national cyberwatch mid-atlantic ccdc," http://www.netresec.com/?page=MACCDC, 2016, accessed: 2016-12-16.

[12] Dolan-Gavitt, "Gtisk panda malrec—pcap files from malware samples run in panda," 2018.

[13] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets," in *Proceedings of the third annual conference on privacy, security and trust*, vol. 94, 2005, pp. 1723–1722.

[14] H. R. Zeidanloo and A. A. Manaf, "Botnet command and control mechanisms," in *2009 Second International Conference on Computer and Electrical Engineering*, vol. 1. IEEE, 2009, pp. 564–568.

[15] W. Wiki, "Libpcap file format," *URL https://wiki. wireshark. org/Development/LibpcapFileFormat*.

[16] D. Crockford, "The application/json media type for javascript object notation (json)," 2006.

[17] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[18] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[19] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[20] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[21] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.