DOI: 10.20998/2079-0023.2023.02.11
UDC 004.056:004.089

A. K. DREMOV, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Postgraduate Student of the Computer Engineering Department of the faculty of Informatics and Computer Science, Kyiv, Ukraine, e-mail: a.k.dremov@gmail.com, ORCID https://orcid.org/0009-0005-7214-9458

METHODS AND MEANS TO IMPROVE THE EFFICIENCY OF NETWORK TRAFFIC SECURITY MONITORING BASED ON ARTIFICIAL INTELLIGENCE

This paper aims to provide a solution for malicious network traffic detection and categorization. Remote attacks on computer systems are becoming more common and more dangerous nowadays. This is due to several factors, some of which are as follows: first of all, the usage of computer networks and network infrastructure overall is on the rise, with tools such as messengers, email, and so on. Second, alongside increased usage, the amount of sensitive information being transmitted over networks has also grown. Third, the usage of computer networks for complex systems, such as grid and cloud computing, as well as IoT and "smart" locations (e.g., "smart city") has also seen an increase. Detecting malicious network traffic is the first step in defending against a remote attack. Historically, this was handled by a variety of algorithms, including machine learning algorithms such as clustering. However, these algorithms require a large amount of sample data to be effective against a given attack. This means that defending against zero-day attacks or attacks with high variance in input data proves difficult for such algorithms. In this paper, we propose a semi-supervised generative adversarial network (GAN) to train a discriminator model to categorize malicious traffic as well as identify malicious and non-malicious traffic. The proposed solution consists of a GAN generator that creates tabular data representing network traffic from a remote attack and a classifier deep neural network for said traffic. The main goal is to achieve accurate categorization of malicious traffic with a few labeled examples. This can also, in theory, improve classification accuracy compared to fully supervised models. It may also improve the model's performance against completely new types of attacks. The resulting model shows a prediction accuracy of 91 %, which is lower than a conventional deep learning model; however, this accuracy is achieved with a small sample of data (under 1000 labeled examples). As such, the results of this research may be used to improve computer system security, for example, by using dynamic firewall rule adjustments based on the results of incoming traffic classification. The proposed model was implemented and tested in the Python programming language and the TensorFlow framework. The dataset used for testing is the NSL-KDD dataset.

Keywords: cybersecurity, network security, malicious traffic identification, machine learning, generational adversarial networks, semi-supervised learning.

Introduction. Computer networks are a key part of modern digital communications. However, these networks can be susceptible to malicious network traffic and various attacks. These attacks can be categorized by specific packet information used in them, such as the source address, service and port used, protocol used, etc. As such, network intrusion and attack detection play an important part in identifying an attack and counteracting it and are relevant areas of research.

Additionally, modern machine learning methods and algorithms can be used to categorize data or objects with great precision, provided there is a large enough training sample. A variety of statistical analysis methods are used to categorize the data. These include data clustering, which attempts to group data points based on their similarity to each other. A well-known example of such an approach is k-means clustering, which attempts to assign N data points to K clusters. A variation of this algorithm called k-nearest neighbors is often used for classification problems. However, accurate clustering often requires a large number of data points. Another common machine learning algorithm approach is supervised learning, for example, using a decision tree classifier. In this approach, we attempt to create a relation between input data and class labels in a tree-like structure. However, as a supervised learning algorithm, this requires the input data to be labeled. A common problem with these approaches is the necessity of having a large number of labeled examples. With rapid developments in security penetration, a problem has appeared where new penetration methods appear frequently and gathering enough packet samples for model training becomes a difficult task. Therefore, the aim of this research is to develop a method for classifying network traffic with a small number of labeled examples.

Problem definition. The core problem that the research focuses on is the problem of malicious traffic identification and categorization. The first part of the problem is the identification of whether or not traffic is malicious in nature. Malicious traffic is one that can be used to attack the computer network and individual devices in the network and includes malware, DoS attacks, network scanning, data exfiltration, R2L, etc. The second part of this problem is the categorization of malicious traffic.

Relevant works. A number of researchers have tackled the problem of network attack classification [1, 2] and the effect of malicious traffic on computer networks [3, 4]. Of particular interest to this paper is the general approach to performing a network attack described in [1], as well as the classification and effects described in [2] and [3], respectively.

Additionally, research into intrusion detection and, more importantly, an analysis of malicious traffic packet contents [5–7] help connect network attacks to packet contents. This allows for the definition of features used by the machine learning algorithm.

Lastly, research in the area of applying machine learning to solve network intrusion detection problems was performed [8], where a variety of models and algorithms were used. Approaches to categorizing tabular data with ML algorithms are described in [9]. This research describes the architecture of GAN networks and semi-supervised GAN networks [10–12].

In the author's opinion, the problem of intrusion detection using machine learning algorithms when there is insufficient data remains understudied. Additionally, proposed solutions may encounter difficulty with generalization when being applied in different scenarios. A GAN-based model could be used to achieve a greater degree of generalization.

Research objective. The purpose of this work is to research methods and models of malicious network traffic detection and categorization with the use of artificial intelligence models. Additionally, the purpose of the work is to create a ML model that can be used to detect and classify malicious traffic with packet information.

Dataset information.

The dataset used in this research is NSL-KDD [13], which contains 125000 examples of network traffic packet data as well as 22 categories based on attack type. Packets labeled “normal” indicate no attack. The features used in the classification include internet protocol used, service used, login status, login attempts, attempts to take root status, file and script creation, error rate, and others, for a total of 41 features. A total of 67000 records are labeled as non-malicious traffic, and 58000 are labeled as malici-ous (fig. 1, fig. 2, fig. 3).

Fig. 1 illustrates that most attacks seem to occur via tcp and icmp protocols, whereas udp connections are less likely to be malicious.

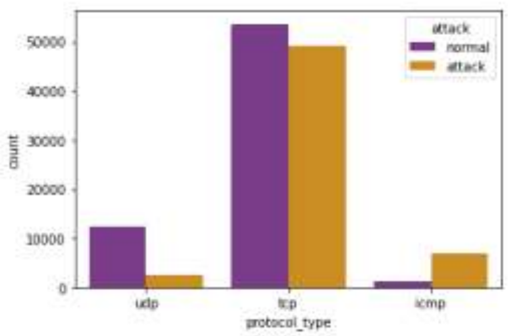


Fig. 1. Distribution of malicious and non-malicious traffic with regards to protocol used

Fig. 2 shows a small sample of dataset entries. Here we can see that most of the data is continuous numerical data. However, it should be noted that the fields “protocol”, "service,” and “flag” are categorical.

	duration	protocol_type	service	flag	src_bytes	dst_bytes
0	0	1	20	9	491	0
1	0	2	44	9	146	0
2	0	1	49	5	0	0
3	0	1	24	9	232	8153
4	0	1	24	9	199	420
...	...	...	...	...	...	...
125968	0	1	49	5	0	0
125969	8	2	49	9	105	145
125970	0	1	54	9	2231	384
125971	0	1	30	5	0	0
125972	0	1	20	9	151	0

Fig. 2. Selection of examples from the dataset [13]

```
RangeIndex: 125973 entries, 0 to 125972
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     125973 non-null  int64
 1   protocol_type                125973 non-null  int64
 2   service                      125973 non-null  int64
 3   flag                         125973 non-null  int64
 4   src_bytes                    125973 non-null  int64
 5   dst_bytes                    125973 non-null  int64
 6   land                         125973 non-null  int64
 7   wrong_fragment               125973 non-null  int64
 8   urgent                       125973 non-null  int64
 9   hot                          125973 non-null  int64
 10  num_failed_logins            125973 non-null  int64
 11  logged_in                    125973 non-null  int64
 12  num_compromised              125973 non-null  int64
 13  root_shell                   125973 non-null  int64
 14  su_attempted                 125973 non-null  int64
 15  num_root                     125973 non-null  int64
 16  num_file_creations           125973 non-null  int64
 17  num_shells                   125973 non-null  int64
 18  num_access_files             125973 non-null  int64
 19  num_outbound_cmds            125973 non-null  int64
 20  is_host_login                125973 non-null  int64
 21  is_guest_login               125973 non-null  int64
 22  count                        125973 non-null  int64
 23  srv_count                    125973 non-null  int64
 24  serror_rate                  125973 non-null  float64
 25  srv_serror_rate              125973 non-null  float64
 26  rerror_rate                  125973 non-null  float64
 27  srv_rerror_rate              125973 non-null  float64
 28  same_srv_rate                125973 non-null  float64
 29  diff_srv_rate                125973 non-null  float64
 30  srv_diff_host_rate           125973 non-null  float64
 31  dst_host_count               125973 non-null  int64
 32  dst_host_srv_count           125973 non-null  int64
 33  dst_host_same_srv_rate       125973 non-null  float64
 34  dst_host_diff_srv_rate       125973 non-null  float64
 35  dst_host_same_src_port_rate  125973 non-null  float64
 36  dst_host_srv_diff_host_rate  125973 non-null  float64
 37  dst_host_serror_rate         125973 non-null  float64
 38  dst_host_srv_serror_rate     125973 non-null  float64
 39  dst_host_rerror_rate         125973 non-null  float64
 40  dst_host_srv_rerror_rate     125973 non-null  float64
 41  attack                       125973 non-null  int64
 42  level                        125973 non-null  int64
dtypes: float64(15), int64(28)
memory usage: 41.3 MB
```

Fig. 3. Dataset information [13]

Fig. 3 shows all of the data fields used in the dataset. This presents the full set of features (for this dataset) that may be used to identify malicious traffic. For the purposes of this research, no significant data processing was performed.

Presentation of the main material.

The following data pre-processing was performed: the categorical values were converted to numerical values, the dataset was scaled using standard scaling, equation (1).

$$z = \frac{x - \overline{\mu}}{\sigma}, \tag{1}$$

where z – standardized value.

x – original feature vector.

$\overline{\mu}$ – mean of the feature vector.

σ – standard deviation of the feature vector.

The labels were one-hot encoded in order to be used for categorical classification.

For training, we make use of a 70:30 split of training to test data.

As a baseline classifier, a simple deep neural network was implemented using TensorFlow keras with two fully connected layers with 32 and 16 neurons, activation function is “relu”, batch normalization layers, and dropout layers to prevent overfitting (fig. 4). The final layer is a dense layer with “softmax” activation for categorical classification. Model metrics are “categorical_crossentropy” for loss function and “categorical_accuracy” for accuracy. The model was trained for 50 epochs on the dataset and achieved 99 %

accuracy, indicating possible overfitting (fig. 5). This classifier will be used to evaluate the performance of the GAN-based classifier.

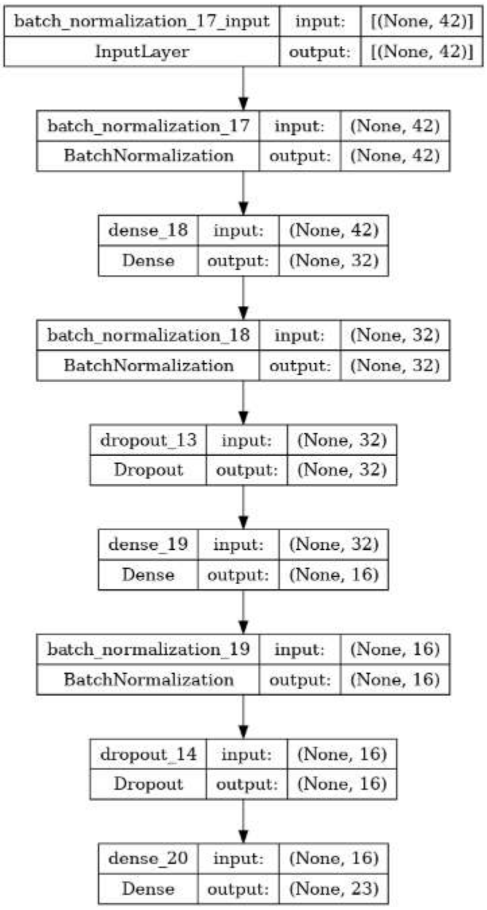
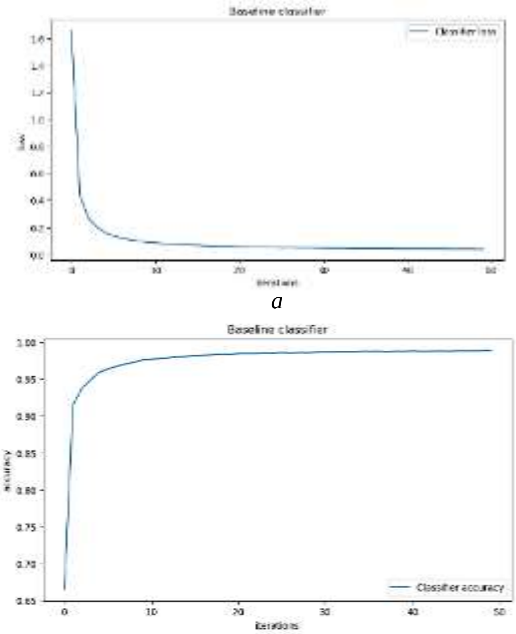


Fig. 4. Baseline classifier model



a

b

Fig. 5. Baseline classifier model training metrics: *a* – loss metric, *b* – accuracy metric

The second model is based on a generative adversarial network (GAN). These networks consist of a generator model and a classifier model. The generator uses gaussian distribution noise to generate fake information, equation (2).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \qquad (2)$$

where $P(x)$ – probability density function for x.

x – original feature vector.

μ – mean of the distribution.

σ – standard deviation of the distribution.

The classifier model of GAN is used to classify generator output as real or fake. For this, a DNN with sigmoid activation is used, somewhat similar to the baseline classifier network shown earlier. The result of the classification is used to calculate generator loss and discriminator loss (fig. 6). This allows the generator to be trained to create more believable fake data.

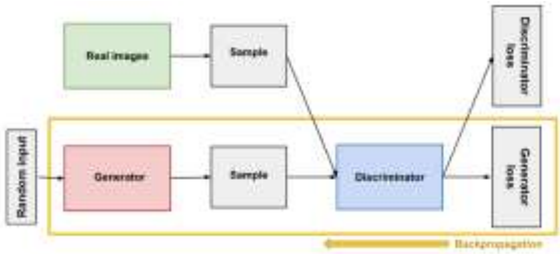


Fig. 6. General architecture of a GAN network [14]

As seen in fig. 6, the GAN network can use portions of real data and generator input to create fake data. This data is then categorized by the discriminator model. The main purpose of such a network is to train the generator model to create fake data that is similar enough to real data that the discriminator model cannot tell a difference.

A subtype of GAN networks is a semi-supervised GAN. These are often used when trying to create a generator with few real samples available. In this case, the discriminator predicts N+1 classes, with an additional label being used for fake data classification. Of particular interest to this research is the efficiency of the categorical discriminator, not the generator model. The approach used is to feed a small number of labeled samples to the classifier on each iteration alongside a large number of unlabeled samples, partially by removing labels from real data and by using generated data.

In our implementation, we use two discriminator mo-dels, one for real/fake categorization and another for attack categorization. The target of the research is the attack categorization model. The models share weights to ensure correct categorization for real and fake as well as attack class. We use two dense layers of size 256 and "relu" activation, as well as batch normalization and dropout layers. Output layers are "softmax" for the categorical classification model and "sigmoid" for binary classification. Loss functions and metrics are "categorical_crossentropy", "binary_crossentropy", "categorical_accuracy" and "binary_accuracy" for the categorical discriminator and the binary discriminator, respectively (fig. 7, fig. 8). Since all of our input data is labeled, we only use a small sample of labeled entries, between 100 and 500 samples, as input for the categorical classifier model. For the generator, a model with three dense layers was used with 128, 256, and 512 nodes and “relu” activation. Additionally, batch normalization and dropout layers were used. The output layer is a dense layer with nodes equal to the number of features and “tanh” activation (fig. 9). For model training, 10 epochs were used. With final training, categorical accuracy is around 99 % and binary accuracy is around 78 %. Final validation categorical accuracy is around 89 %. This indicates possible model overfitting (fig. 10–12).

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 42)]              0

 dense_3 (Dense)             (None, 256)               11008

 batch_normalization_3 (Batc  (None, 256)              1024
 hNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 256)               65792

 batch_normalization_4 (Batc  (None, 256)              1024
 hNormalization)

 dropout_3 (Dropout)         (None, 256)               0

 output_cat (Dense)          (None, 23)                5911

=================================================================
Total params: 84,759
Trainable params: 5,911
Non-trainable params: 78,848
```

Fig. 7. Categorical discriminator model

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 42)]              0

 dense_3 (Dense)             (None, 256)               11008

 batch_normalization_3 (Batc  (None, 256)              1024
 hNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 256)               65792

 batch_normalization_4 (Batc  (None, 256)              1024
 hNormalization)

 dropout_3 (Dropout)         (None, 256)               0

 output_bin (Dense)          (None, 1)                 257

=================================================================
Total params: 79,105
Trainable params: 0
Non-trainable params: 79,105
_________________________________________________________________
```

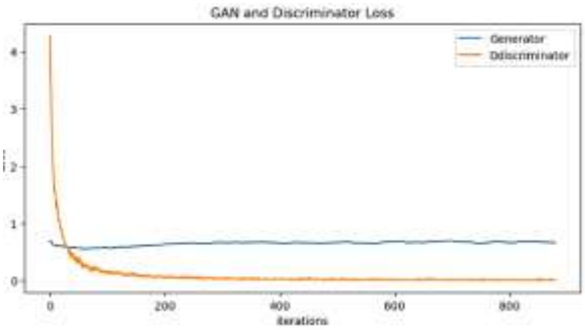
Fig. 8. Binary discriminator model

In fig. 7 and fig. 8, we can see the characteristics of the classifier models. As mentioned before, the classifier model from fig. 7 is used to predict class labels, and the binary classifier from fig. 8 is used to improve the generator model.

In fig. 9, *a* generator model is presented that takes an input of gaussian noise and real data and outputs generated data fields similar to the initial dataset. The final layer of the model goes directly into the classifier model.

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 64)]              0

 dense_5 (Dense)             (None, 128)               8320

 dropout_4 (Dropout)         (None, 128)               0

 batch_normalization_5 (Batc  (None, 128)              512
 hNormalization)

 dense_6 (Dense)             (None, 256)               33024

 dropout_5 (Dropout)         (None, 256)               0

 batch_normalization_6 (Batc  (None, 256)              1024
 hNormalization)

 dense_7 (Dense)             (None, 512)               131584

 dropout_6 (Dropout)         (None, 512)               0

 batch_normalization_7 (Batc  (None, 512)              2048
 hNormalization)

 dense_8 (Dense)             (None, 42)                21546

 model_1 (Functional)        (None, 1)                 79105

=================================================================
Total params: 277,163
Trainable params: 196,266
Non-trainable params: 80,897
_________________________________________________________________
```

Fig. 9. GAN (generator and discriminator) model



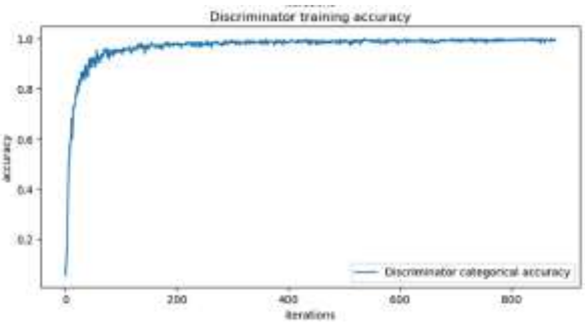
a

b

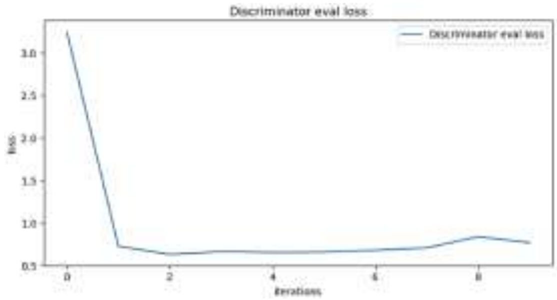
Fig. 10. GAN model metrics: *a* – generator and binary discriminator loss metrics; *b* – binary discriminator accuracy

In fig. 10, *b* we can see that the binary discriminator accuracy can fluctuate a lot, whereas in 10, *a* we can see that the loss of the binary discriminator is steady and decreasing. This means that the generator is producing data similar to real samples.

In fig. 11, a we can see that the training accuracy for the categorical classifier is high, reaching over 90 %; however, in fig. 11, b and fig. 12 we can see that the loss and accuracy on validation samples are lower, at 88 %. This may be explained by overfitting, perhaps due to model complexity or the values of the generated data being too disjointed from one another compared to real data. For example, a combination of certain values in data fields in real samples may indicate an attack, whereas in generated samples, these values may differ; however, the data would still be created under the “attack” label.



a



b

Fig. 11. Categorical classifier metrics: *a* – classifier training accuracy; *b* – discriminator loss on validation data

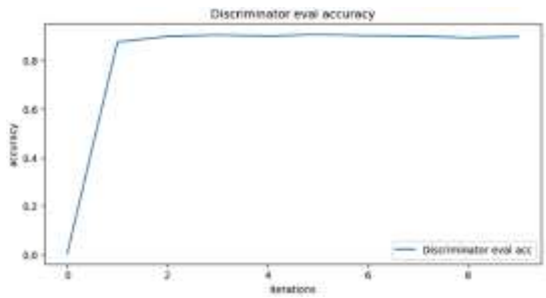


Fig. 12 Classifier model accuracy on validation data

Conclusions. This research proposes the use of semi-supervised GAN model to train a classifier network for categorizing malicious network traffic with a limited number of labeled entries. For comparison, we also used a baseline classifier DNN with a full dataset. The baseline classifier managed to achieve a validation accuracy of 99 %, whereas the SGAN discriminator only achieved 88 %. The SGAN discriminator shows signs of overfitting with a training accuracy of 99 %. While the results are subpar compared to a full dataset classifier, it is worth noting that the SGAN model only received a small portion of the dataset labels, between 100 and 500 samples, in different tests while still achieving a relatively high accuracy score. It should also be pointed out that GAN networks generally have trouble generating entirely new information; instead, they create slight variations of existing data. As such, it may not be able to be used to train a network to predict entirely unknown threats.

Overall, SGAN networks may not be an effective solution to training network attack classifiers; however, additional research may be conducted. In particular, the question of network hyperparameter tuning remains open, as it may allow us to prevent overfitting and improve model accuracy. Additionally, since the research was conducted only on a single dataset, it is worth exploring additional datasets to further evaluate the proposed solution.

References

1. Chasaki D., Wu Q., *Wolf T. Attacks on network infrastructure. 2011 20th international conference on computer communications and networks (ICCCN) Lahaina, HI, USA, 31 July – 4 August 2011*. 2011. URL: https://doi.org/10.1109/icccn.2011.6005919 (date of access: 01.08.2023).
2. Anderson R. *Security engineering: a guide to building dependable distributed systems*. 2nd ed. Indianapolis, IN : Wiley Technology Pub., 2008.
3. Kun-chan L., Alefiya H., Debojyoti D. *Effect of malicious traffic on the network.* The ANT Lab: Analysis of Network Traffic. URL: https://ant.isi.edu/~johnh/PAPERS/Lan03a.pdf (date of access: 03.07.2023).
4. Dubrawsky I., Noonan W. Firewall fundamentals. Cisco Press, 2006. 408 p.
5. John W., Olovsson T. Detection of malicious traffic on back-bone links via packet header analysis. *Campus-Wide information systems.* 2008. Vol. 25, no. 5. P. 342–358. URL: https://doi.org/10.1108/10650740810921484 (date of access: 14.08.2023).
6. Network traffic analysis and intrusion detection using packet sniffer / M. A. Qadeer et al. *2010 second international conference on communication software and networks, Singapore, 26–28 February 2010.* 2010. URL: https://doi.org/10.1109/iccsn.2010.104 (date of access: 12.09.2023).
7. Wang W., Gombault S., Guyet T. Towards fast detecting intrusions: using key attributes of network traffic. *2008 the third international conference on internet monitoring and protection, Bucharest, Romania, 29 June – 5 July 2008.* 2008. URL: https://doi.org/10.1109/icimp.2008.13 (date of access: 09.10.2023).
8. Panda M. A., Iqbal A., Zahid M., Siddiqui M. R. Network intrusion detection system: a machine learning approach. *Intelligent decision technologies.* 2011. Vol. 5, no. 4. P. 347–356. URL: https://doi.org/10.3233/idt-2011-0117 (date of access: 27.10.2023).
9. Kelleher J. D., D'Arcy A., Namee B. M. Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies. MIT Press, 2015. 624 p.
10. Odena A. *Semi-Supervised learning with generative adversarial networks.* arXiv.org. URL: https://arxiv.org/abs/1606.01583 (date of access: 14.10.2023).
11. Pasupa K., Tungjitnob S., Vatathanavaro S. Semi-supervised learning with deep convolutional generative adversarial networks for canine red blood cells morphology classification. *Multimedia tools and applications.* 2020. Vol. 79, no. 45–46. P. 34209–34226. URL: https://doi.org/10.1007/s11042-020-08767-z (date of access: 18.10.2023).
12. Langr J., Bok V. GANs in action: deep learning with generative adversarial networks. Manning Publications Company, 2019. 276 p.
13. Zaib H. *Nsl-kdd.* Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/datasets/hassan06/nslkdd/data (date of access: 05.09.2023).
14. Overview of GAN structure | machine learning | google for developers. *Google for Developers*. URL: https://developers.google.com/machine-learning/gan/gan_structure (date of access: 20.10.2023).

References (transliterated)

1. Chasaki D., Wu Q. and Wolf T., Attacks on network infrastructure. In: *2011 20th international conference on computer communications*

and networks *(ICCCN) 31 July–4 August 2011, Lahaina, HI, USA* [online]. IEEE. [Viewed 1 November 2023]. Available from: doi: 10.1109/icccn.2011.6005919
2. Anderson R., Security engineering: a guide to building dependable distributed systems. 2nd ed. Indianapolis, IN: Wiley Technology Pub., 2008.
3. Kun-chan, L., Alefiya, H. and Debojyoti, D., *Effect of malicious traffic on the network* [online]. The ANT Lab: Analysis of Network Traffic. 2009, [Viewed 3 July 2023]. Available from: https://ant.isi.edu/~johnh/PAPERS/Lan03a.pdf
4. Dubrawsky I. and Noonan W., Firewall fundamentals. Cisco Press, 2006.
5. John W. and Olovsson T., Detection of malicious traffic on back-bone links via packet header analysis. *Campus-Wide Information Systems* [online]. 25(5), 2008, 342–358. [Viewed 14 August 2023]. Available from: doi: 10.1108/10650740810921484
6. Qadeer M. A., Iqbal A., Zahid M. and Siddiqui, M. R., Network traffic analysis and intrusion detection using packet sniffer. In: *2010 second international conference on communication software and networks, 26–28 February 2010, Singapore* [online]. IEEE. [Viewed 12 September 2023]. Available from: doi: 10.1109/iccsn.2010.104
7. Wang W., Gombault S. and Guyet T., Towards fast detecting intrusions: using key attributes of network traffic. In: *2008 the third international conference on internet monitoring and protection, 29 June–5 July 2008, Bucharest, Romania* [online]. IEEE. [Viewed 9 October 2023], 2008, Available from: doi: 10.1109/icimp.2008.13
8. Panda M. A., Iqbal A., Zahid M., Siddiqui M. R. Network intrusion detection system: a machine learning approach. *Intelligent Decision Technologies* [online]. 5(4), 2011, 347–356. [Viewed 27 October 2023]. Available from: doi: 10.3233/idt-2011-0117
9. Kelleher J. D., D'Arcy A., Namee B. M. Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies. MIT Press, 2015.
10. Odena A. *Semi-Supervised learning with generative adversarial networks* [online]. arXiv.org. [Viewed 14 October 2023]., 2016, Available from: https://arxiv.org/abs/1606.01583
11. Pasupa K., Tungjitnob S., Vatathanavaro S. Semi-supervised learning with deep convolutional generative adversarial networks for canine red blood cells morphology classification. *Multimedia Tools and Applications* [online]. 79(45–46), 2020, 34209–34226. [Viewed 18 October 2023]. Available from: doi: 10.1007/s11042-020-08767-z
12. Langr J., Bok V. GANs in action: deep learning with generative adversarial networks. Manning Publications Company, 2019.
13. Zaib H. *Nsl-kdd* [online]. Kaggle: Your Machine Learning and Data Science Community, 2018, [Viewed 05 September 2023]. Available from: https://www.kaggle.com/datasets/hassan06/nslkdd/data
14. Overview of GAN structure | machine learning | google for developers [online], *Google for Developers* [Viewed 20 October 2023]. Available from: https://developers.google.com/machine-learning/gan/gan_structure

Received 27.10.2023

УДК 004.056:004.089

А. К. ДРЕМОВ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», аспірант кафедри обчислювальної техніки, факультет інформатики та обчислювальної техніки, м. Київ, Україна, e-mail: a.k.dremov@gmail.com, ORCID: https://orcid.org/0009-0005-7214-9458

МЕТОДИ ТА ЗАСОБИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ МОНІТОРИНГУ БЕЗПЕКИ МЕРЕЖЕВОГО ТРАФІКУ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Ця стаття має на меті запропонувати рішення для виявлення та категоризації шкідливого мережевого трафіку. Віддалені атаки на комп'ютерні системи стають все більш поширеними та небезпечними в наш час. Це пов'язано з декількома факторами, деякі з яких наведені нижче. По-перше, зростає використання комп'ютерних мереж та мережевої інфраструктури в цілому за допомогою таких інструментів, як месенджери, електронна пошта тощо. По-друге, разом зі збільшенням використання зростає і обсяг конфіденційної інформації, що передається мережами. По-третє, зросло використання комп'ютерних мереж у складних системах, таких як електромережі, хмарні обчислення, а також Інтернет речей і «розумні» локації (наприклад, «розумне місто»). Виявлення шкідливого мережевого трафіку є першим кроком у захисті від віддаленої атаки. Історично це робилося за допомогою різних алгоритмів, в тому числі алгоритмів машинного навчання, таких як кластеризація. Однак ці алгоритми вимагають великої кількості вибіркових даних, щоб бути ефективними проти певної атаки. Це означає, що захист від атак нульового дня або атак з великою дисперсією вхідних даних виявляється складним для таких алгоритмів. У цій статті ми пропонуємо напівкеровану генеративну змагальну мережу (GAN) для навчання моделі дискримінатора для класифікації зловмисного трафіку, а також для ідентифікації зловмисного і нешкідливого трафіку. Запропоноване рішення складається з генератора GAN, який створює табличні дані, що представляють мережевий трафік від віддаленої атаки, і класифікатора глибокої нейронної мережі для цього трафіку. Основна мета – досягти точної категоризації шкідливого трафіку за допомогою невеликої кількості маркованих прикладів. Теоретично це також може підвищити точність класифікації порівняно з повністю контрольованими моделями. Це також може покращити ефективність моделі проти абсолютно нових типів атак. Отримана модель показує точність передбачення 91%, що нижче, ніж у звичайної моделі глибокого навчання, однак ця точність досягається на невеликій вибірці даних (менше 1000 маркованих прикладів). Таким чином, результати цього дослідження можуть бути використані для підвищення безпеки комп'ютерних систем, наприклад, за допомогою динамічного налаштування правил брандмауера на основі результатів класифікації вхідного трафіку. Запропонована модель була реалізована та протестована на мові програмування Python та фреймворку Tensorflow. Для тестування використовувався набір даних NSL-KDD.

Ключові слова: кібербезпека, мережева безпека, ідентифікація шкідливого трафіку, машинне навчання, генеративні змагальні мережі, напівкероване навчання

Повне ім'я автора / Author's full name

Автор 1 / Author 1: Дремов Артем Кирилович, Dremov Artem Kyrylovych