2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom) | 978-1-6654-3574-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00167

A User Behavior-Based Random Distribution Scheme for Adversarial Example Generated CAPTCHA

1st Wenwen Zheng
School of Computer Science
China University of Geosciences
Wuhan, China
zhengwenwen@cug.edu.cn

2nd Weiqi Wang
School of Computer Science
China University of Geosciences
Wuhan, China
cugweiqiwang@163.com

3rd Wei Ren
School of Computer Science, China University of Geosciences, Wuhan, China
Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, China
Key Laboratory of Network Assessment Technology, CAS
(Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China 100093)
weirencs@cug.edu.cn

4th Shiyuan Feng
School of Computer Science
China University of Geosciences
Wuhan, China
cugmonster@gmail.com

5th Shiqin Liu
School of Computer Science
China University of Geosciences
Wuhan, China
Shiqin.Liu@cug.edu.cn

6th Yi Ren
School of Computing Science
University of East Anglia
Norwich NR4 7TJ, U.K
e.ren@uea.ac.uk

***Abstract*—The widespread and mature application of deep learning technology on human behavior analysis leads to the possibility that the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) may not be able to accurately determine whether or not the user is human, which makes the distributed denial-of-service (DDoS) attacks launched by deep learning-based malicious automatic computer programs become possible. The CAPTCHA generated by adversarial examples can defend against the automatic identification by the deep learning-based method, however, if the deployed CAPTCHAs are all adversarial examples solely, the adversary has the chance to collect sufficient adversarial examples, which also causes security issues. In this paper, a user behavior-based random distribution scheme for adversarial example generated CAPTCHA is proposed to tackle the above sole distribution problem. Specifically, we generated two kinds of CAPTCHAs, one is normal and the other is generated by the fast gradient signed method (FGSM), i.e., the adversarial examples. Meanwhile, user behaviors are analyzed and associated with a reasonable probability model. According to the probability selection, the normal CAPTCHAs or strong CAPTCHAs will be delivered to the users. The experimental results illustrate that our scheme has an excellent ability to distinguish computers from humans. Thus, it can protect the CAPTCHA systems from DDoS attacks.**

***Index Terms*—Deep learning, CAPTCHA, adversarial example, behavior analysis, user behavior-based random distribution.**

I. Introduction

CAPTCHA is the abbreviation of “Completely Automated Public Turing test to tell Computers and Humans Apart” that mainly includes the text CAPTCHA, the picture CAPTCHA, the interactive CAPTCHA, etc. Among these types of CAPTCHAs, the text CAPTCHA is one of the widest usages because of its simple deployment, maintenance, and identification. Generally, the text CAPTCHA is composed of several lines and several irregular characters, which is also utilized at large in user registration, login, authentication, and other similar fields. But, once the CAPTCHA is easily recognized by the automatic computer programs in order to automatically register or login, security threats and even security vulnerabilities may emerge. Therefore, generating CAPTCHAs that are difficult to be recognized by the automatic programs or software tools is of critical importance.

CAPTCHAs generation based on adversarial examples technology is an important research field which has important theoretical significance and high application value. The CAPTCHAs generated by adversarial examples can resist the recognition based on deep learning with strong robustness. However, only using adversarial examples as CAPTCHAs will inversely help the adversary obtain massive data in a short time, which can be used by adversary for enhancing the recognition performance of its deep learning model.

978-1-6654-3574-1/21/$31.00 ©2021 IEEE

DOI 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00167

To deal with the existing issues, in this paper, we proposed a user behavior-based random distribution scheme for adversarial example generated CAPTCHA. The proposed scheme generates CAPTCHAs using adversarial examples, and the CAPTCHAs are easily recognized by human users, but with a low recognition rate of deep learning model-based programs. This can avoid the malicious explosion launched by the adversary for CAPTCHA systems. In addition, by modeling users, our scheme can calculate user behaviors and built-in malicious automatic program lines. Besides, in order to maintain the quality of strong CAPTCHAs (i.e., achieving accurate delivery of a strong CAPTCHA), the probability of strong CAPTCHA occurrence is determined and updated iteratively. At the same time, it can reduce the budget consumption of adversarial examples and the recognition accuracy of neural network, and improve the proceeding speed. Specifically, the main contributions of this paper are as follows:

- We evaluate the CAPTCHA recognition capability of several mainstream deep learning network models (e.g., VGG16), meanwhile, we use FGSM to generate adversarial examples, and test the adversarial examples CAPTCHA recognition capability of the above models.
- We set the behavioral eigenvectors of automatic computer programs and humans on CAPTCHA recognition.
- We propose a random distribution scheme for adversarial examples generated CAPTCHAs, based on user behaviors, which distributes normal CAPTCHAs or CAPTCHAs generated by adversarial examples according to a probability model.

The rest of the paper is organized as follows: In Section II, we give a brief introduction to the related literature of deep learning identification. Additionally, we discuss various deep learning network models and adversarial examples generation methods. Then, in Section III, we describe our proposed scheme in detail. Next, we demonstrate the experiment and its results in Section IV. At last, Section V concludes the whole paper.

II. Related Works

A. CAPTCHA Identification

Currently, most of the mainstream CAPTCHA recognition programs adopt deep neural network (DNN) technology. In order to improve the low recognition accuracy of DNN for confusion class, Chen et al. [1] presented a new method of selective learning confusion class. The confusion sample and ordinary sample are separated, and two models are trained combined. Yu and Darling [2] applied TensorFlow object detection and the peak assignment algorithm combined with convolutional neural network (CNN), proposing a low-cost CAPTCHA cracking method based on artificial intelligence (AI), which can train the model infinitely and make the model achieve the good marking effect. In addition, Qing and Zhang [3] proposed a CAPTCHAs recognition algorithm based on the multi-label neural network to work out the connectivity problem. By modifying convolution to learn not only the character features but also the position of each character, the multi-label CNN can solve CAPTCHAs in an undivided way. Hu et al. [4] introduced the concept of multi-task learning. Specifically, in their work, it means recognizing each character is equivalent to a task and normalizing the local contrast of the image to avoid the saturation of neuron output caused by too many absolute input values.

B. Adversarial Example

Szegedy et al. [5] first proposed the concept of the "adversarial example". They mentioned an intriguing property of neural networks that it is simple to cause the network to misclassify an image by applying a certain hardly perceptible perturbation, which is found by maximizing the network's prediction error. The new images generated through the above way are called adversarial examples, and the most significant feature of this method is adding some undetectable perturbations to the original data (e.g., image, text, and audio) does not affect human recognition, however, it is possible to fool the DNN models and cause these models to make misclassification of the data. Akhtar and Mian [6] believed that the adversarial example is not a bug of deep learning, but a feature of deep learning. Features are divided into robust features perceived by humans and non-robust features in their paper, and they considered that adversarial examples change the non-robust features that affect models, not humans.

Goodfellow et al. [7] suggested the existence of adversarial examples is not the previously proposed nonlinearity, but the linearity. They argued that the higher-dimensional space is sufficient to generate the adversarial examples. Then they proposed FGSM based on gradient descent, which is used in our work. The DeepFool model [8] is also a gradient-based white-box attack and directional attack algorithm. Compared with FGSM, the DeepFool algorithm can calculate smaller perturbations without specifying the learning rate. Also, Papernot et al. [9] proposed generating adversarial examples by limiting the L_0 norm, and obtaining the better attack effect by iterating to modify the information of only a few pixels at a time.

Adversarial examples are mainly utilized in image classification and recognition, but they also play an important role in other fields. For example, Tabacof et al. [10] proposed a method that disturbs the input image so as to misguide the autoencoder to produce a different image again. Apart from that, they also found that the robustness of the autoencoder is better than that of the general neural network. Kos et al. [11] applied adversarial examples on the deep generative model, specifically, the variational autoencoder composed with a generative adversarial network (VAE-GAN). Moreover, they proved that the existence of adversarial examples is a general phenomenon of neural network structure. Moosavi-Deztraili et al. [12] used adversarial examples to semantic segmentation and object recognition, and proved that image-independent quasi-imperceptible perturbations can be used to dupe neural networks to generate false image segmentation.

In addition, Rozsa et al. [13] proposed an approach for applying generated adversarial examples to change facial attribute recognition and explored the stability of deep learning models in the field of facial recognition. Athalye et al. [14] proposed that 3D models can also fool neural networks by changing different angles and perspectives. Besides, Yang Zhang et al. [15] study the effect of adversarial examples on CAPTCHA robustness (including image-selecting, clicking-based, and text-based CAPTCHAs). Hyun KWON et al. [16] propose a method to generate CAPTCHAs images by using FGSM, iterative FGSM, which effect the recognition rate full sharply. Chenghui Shi et al. [17] identify the difference between adversarial CAPTCHA generation and existing hot adversarial example (image) generation research. Then, they design an adversarial CAPTCHA generation and evaluation system, called aCAPTCHA.

III. Proposed Scheme

In this section, we will describe our scheme in detail. The proposed scheme mainly consists of two modules: (1) normal and strong CAPTCHAs generation (NSCG) module, and (2) user behavior analysis (UBA) module. The NSCG module is used to generate normal CAPTCHAs and adversarial examples of CAPTCHAs with high security and user-friendly design. The UBA module is utilized to receive and analyze the user's behaviors, additionally, it also calculates the confidence of the user as a malicious automatic program and judges the user as a human or a malicious automatic program. Finally, the UBA module takes the confidence as the feedback to choose a CAPTCHA.

Besides, our scheme displays the identification results of different CAPTCHAs (i.e., normal CAPTCHAs and strong CAPTCHAs) under different models training. At the same time, the identification result of strong examples is used as the basis of user identification to determine whether it can be further operated. We hereby give five definitions below:

- **Identification time** (t_i): The time from the user's input in the i^{th} operation to the display of CAPTCHAs.
- **User behavior vector** ($\vec{u}$): The user behavior vector is used to quantify and model the user's behavior. The vector is a triple of the average identification time (t_{avg}) in milliseconds, recognition times (n) and identification accuracy (acc), which is calculated and updated by the UBA module.
- **Standard automatic computer program vector** ($\vec{s}$): The vector is the built-in standard vector of automatic computer program behavior. It is also a triple of the average identification time (t_{avg}) recognition times (n) and identification accuracy (acc), and is determined by the training results of deep learning model.
- **User confidence** ($conf_i$): The similarity between users and built-in programs measures the probability that users are malicious automatic computer programs.
- **Strong sample distribution probability** (P_i): The probability that the CAPTCHA image distributed by the user is a strong sample in the i^{th} operation.

A. Normal and Strong CAPTCHAs Generation

We construct two CAPTCHA sets called normal CAPTCHA set and strong CAPTCHA set respectively. First, we use Python library CAPTCHA to generate text CAPTCHAs randomly as normal CAPTCHAs set. There are 36 characters in text CAPTCHAs including 10 digits and 26 uppercase letters. The text CAPTCHA's shape is (170, 80, 3). We generated 30,000 text CAPTCHA as normal set.

The adversarial examples can hardly affect human recognition but seriously cheat the deep neural networks. We use FGSM model as mentioned in Section II to generate strong CAPTCHAs set. For each CAPTCHA, we calculate the gradient of identification model such as CNN, Xception, VGG16, get the gradient orientation using the sign function, then mutiply a step length on the orientation, add it on the original input as the adversarial example. FGSM model is descripted as follows.

Let θ be the parameters of a model, x be the input to the model, y be the targets associated with x and $J(\theta, x, y)$ be the cost used to train the neural network. We can linearize the cost function around the current value of θ, obtaining an optimal max-norm constrained pertubation $\eta = \epsilon sign(\bigtriangledown_x J(\theta, x, y)))$. It is recorded that FGSM of generating adversarial examples. It is noted that the required gradient can be computed efficiently using backpropagation.

Then we use normalized correlation coefficient (NC) to measure the similarity between normal CAPTCHAs and strong CAPTCHAs. The equation of NC is shown as Eq. (1) below:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{1}$$

B. User Behavior Analysis Process

In this part, we use the UBA module to determine the user's identity. The process includes the following steps, and the whole flow chart of our scheme is illustrated in Fig. 1.

Step 1: In the first step, in order to prevent adversarial examples from constantly entering the method to obtain a large number of strong CAPTCHAs to train adversary's model, we default the first CAPTCHA as the normal one. By recording the user's recognition time, recognition times and recognition accuracy, we assign the initial value to the user model, calculate the initial user behavior confidence, and then determine the distribution probability of the initial strong sample CAPTCHA.

The user confidence is the cosine similarity between the user vector $\vec{u}$ and the built-in standard automatic computer program vector $\vec{s}$, and the calculation formula is shown in Eq. (2) as follows:

$$conf_i = \frac{\vec{u} \cdot \vec{s}}{||\vec{u}|| \cdot ||\vec{s}||}, \tag{2}$$

more specifically, Eq. (2) that is formed as Eq. (3) below:

$$conf_i = \frac{t_s \cdot t_u + n_s \cdot n_u + acc_s \cdot acc_u}{\sqrt{{t_s}^2 + {n_s}^2 + {acc_s}^2} \cdot \sqrt{{t_u}^2 + {n_u}^2 + {acc_u}^2}}, \tag{3}$$

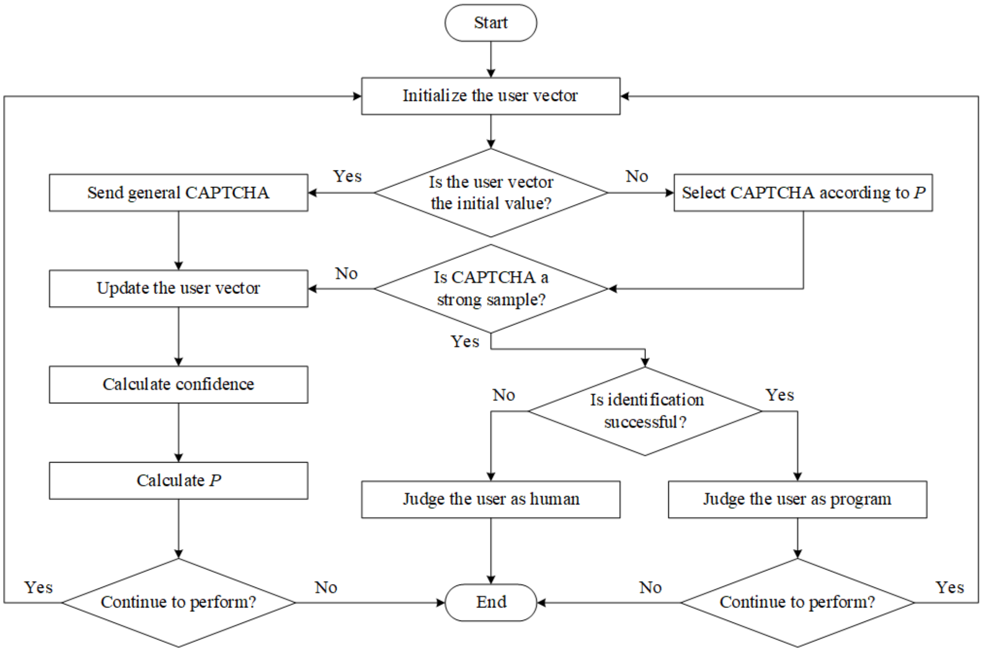

Start
Initialize the user vector
Is the user vector the initial value?
Yes
No
Send general CAPTCHA
Select CAPTCHA according to P
Update the user vector
No
Is CAPTCHA a strong sample?
Yes
Calculate confidence
No
Is identification successful?
Yes
Calculate P
Judge the user as human
Judge the user as program
Yes
Continue to perform?
No
End
No
Continue to perform?
Yes

Fig. 1. The flow chart of proposed scheme

where the initial distribution probability is $P_1 = conf_1$.

Step 2: According to the distribution probability of the last operation, this probability is used as the distribution probability of the strong CAPTCHA for this operation, and the user triples are updated according to the user behavior of this operation, and the new user confidence and the distribution probability of the next operation are calculated. The distribution probability is determined by the Eq. (4):

$$P_n = 1 - \prod_{i=1}^{n}(1 - conf_i). \tag{4}$$

Step 3: If the CAPTCHA image is a strong CAPTCHA image and the user cannot recognize it successfully, it is determined that the user is an automatic computer program and all subsequent operations are prevented; if it can be recognized successfully, it initializes the user model, confidence and distribution probability and returns to **Step 1**, otherwise it returns to **Step 2**.

IV. Experimental Setup and Results

For scheme evaluation, we use the RGB 4-character CAPTCHA with the size of 170 × 80 generated by Python CAPTCHA library as the original CAPTCHA image, and use four popular deep learning models, namely CNN, Xception, and VGG16, as the recognition model, and use FGSM algorithm to generate adversarial examples.

According to the experimental results of human user and program identification test, we find that our proposed scheme has a high recognition probability for malicious automatic computer programs.

A. Experimental Setup

The discrimination index of the method is evaluated by the distribution probability P of the strong CAPTCHA and the number n of the first strong CAPTCHA was given. When the user is human, the lower the similarity between human behavior and the built-in malicious automatic computer program, and the lower the number of the first strong CAPTCHA, that is, the closer P is to 0 and the greater n is, the better the discrimination effect of human behavior is. When it is an automatic computer program, the higher the similarity between the automatic computer program behavior and the built-in malicious automatic computer program, and the higher the number of the first strong verification code, that is, the closer P is to 1 and the smaller n is, the better the discrimination effect of automatic computer program behavior is.

In this experiment, we set the maximum recognition times as 10. If the user fails to recognize the strong CAPTCHA distributed within 10 operations, we determine the user as an automatic computer program; otherwise, we determine the user as a human. That is to say, the user can successfully recognize all the strong CAPTCHA distributed within 10 operations, or all the strong CAPTCHAs distributed within 10 operations are ordinary ones. In addition, in the experiment, we also record the number of distributed pictures n when strong samples appear for the first time in each experiment, as an index to evaluate the effectiveness of the method.

B. Identification Model Training

We choose 20,000 normal CAPTCHAs as the training set and 5,000 normal CAPTCHAs as the test set, by constructing

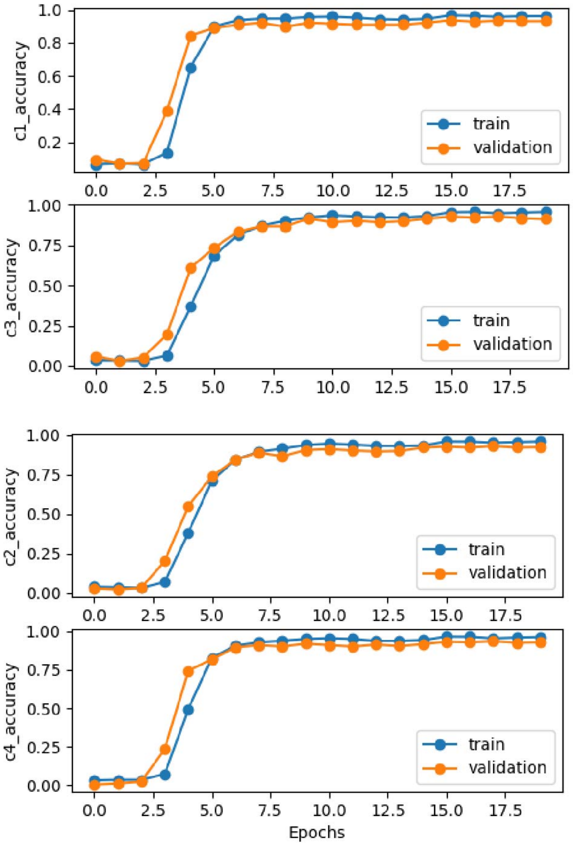


Fig. 2. VGG16 training result

four classifiers. The VGG16 model training process is shown in Fig. 2.

As we can see in Fig.2, the accuracy rate of the VGG16 model has been greatly improved in the 3 or 4 epoch, converges around 10 epochs at 95%. The specific identification model (i.e., CNN, Xception, and VGG16) test results are shown in Fig. 3.

Fig. 3 shows that almost all deep learning models identify CAPTCHAs with over 80% accuracy. This means there are certainly security threats that existed in the CAPTCHA recognition field. Moreover, we can see automatic computer program accuracy, taking VGG16 as an example and focusing on 0.8031. Therefore, we define origin standard automatic computer program vector's ($\vec{s}$) third component acc_s as 0.8031.

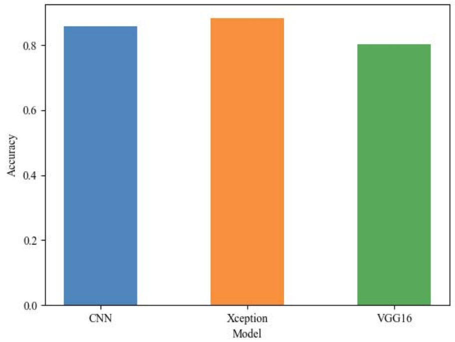
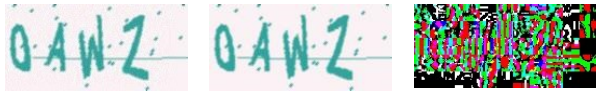


Fig. 3. Test results of identification model

C. FGSM Generation Results

Taking the VGG16 model as an example to show the results, we use the FGSM model to generate adversarial examples. The normal CAPTCHA, adversarial examples CAPTCHA, and the added noise is shown in Fig. 4.



(a) Normal CAPTCHA (b) Strong CAPTCHA (c) CAPTCHA with noise

Fig. 4. A part of FGSM generation results

As can be seen from Fig. 4(a) and Fig. 4(b), there is no difference from human vision system. Then we use NC to measure the similarity between Fig. 4(a) and Fig. 4(b). The result is $NC = 0.95$, which means the difference between 4(a) and Fig. 4(b) is negligible.

D. Adversarial Examples CAPTCHAs Identification Test

After generating adversarial examples, we test how they influence the performance of the trained models. Fig. 5 indicates that deep learning models only have 20%∼30% accuracy according to the test except for Xception (but with high accuracy in strong samples) when recognizing adversarial examples CAPTCHAs. It means that FGSM-generated CAPTCHAs can dramatically degrade the model performance without hindering human recognition. In contrast, strong CAPTCHAs can defend against malicious attacks in a way.

E. Results of Human Experiments

In order to test whether the method can distinguish computers from human users and whether the adversarial examples are user-friendly enough, we recruited 43 volunteers aged 18

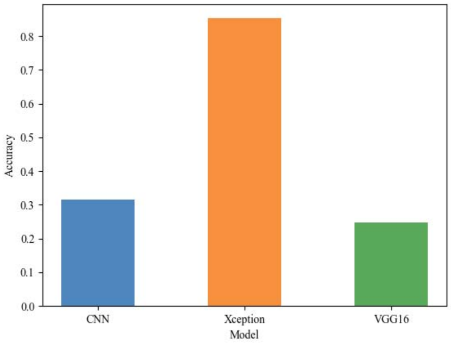


Fig. 5. Identification test results of the CAPTCHA generated by the adversarial example

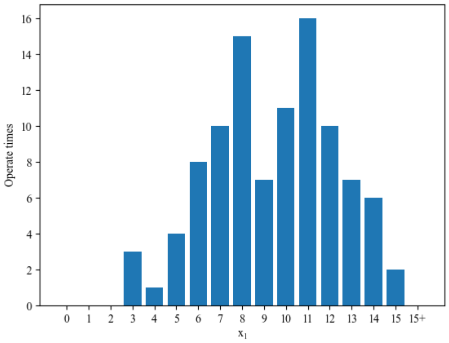


Fig. 6. Human identification time test results

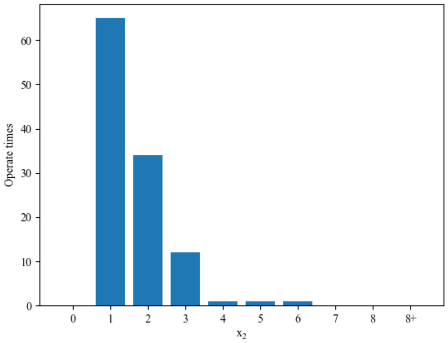


Fig. 8. Using CAPTCHA number when judging as an automatic computer program

to 45 who are familiar with CAPTCHAs, for our experiment. Each participant tested the method 1 to 3 times, a total of 100 times. The bar graph below shows the total number of attempts when the first strong CAPTCHA is delivered in Fig. 6. The x-axis (x_1) represents the number of CAPTCHAs used by the method to identify as human. In the 500 tests, the average number of strong CAPTCHA first appeared is 10.21, that is, when the user is a human, an average of 10.21 ordinary CAPTCHA will be distributed before one strong CAPTCHA is distributed, which means the user is suspected as an automatic computer program.

As for the recognition accuracy, In the 100 tests, 8 users were misjudged as malicious automatic computer programs. The human identification time test result is 92%, which means that the method has a high accuracy rate for detecting users.

F. Results of Automatic Computer Program Experiments

Fig.7 shows automatic computer programs identification time test results that simulated by the VGG16 model. We can see the automatic computer program time focuses on 0.015s, thus, we define origin $\vec{s}$ first component t_s as 0.015s. Then, we normalize the $\vec{s}$ to a unified measure.

From Fig.8, the x-axis (x_2) represents the number of CAPTCHAs used by the method to identify as an automatic computer program. we can find that the number nearly focuses on $1 \sim 2$. The average value is 1.59. Compared with Fig. 6, we can find that the number of the method judged as a person using CAPTCHAs almost focuses on 10. That is to say, the method can effectively distinguish human and automatic computer programs using small CAPTCHAs.

In Fig. 9, we find the distribution probability is almost up to 1, which means the similarity with a real automatic computer program is up to 1. The larger value is, the more similar with the automatic computer programs.

TABLE I
CONFUSION MATRIX

	automatic program	People
Judged as automatic program	100	4
Judged as Human	0	96

According to Table I, we calculated four important values to measure our method, that are shown as follows:

- $Accuracy = \frac{96+100}{200} = 98\%$
- $Precision = \frac{100}{100+4} = 96.15\%$
- $Recall = \frac{100}{100} = 1$
- $F1-score = \frac{2 \cdot 100}{2 \cdot 100+4} = 98.04\%$

The results above illustrate our proposed scheme can distinguish humans and automatic computer programs effectively.

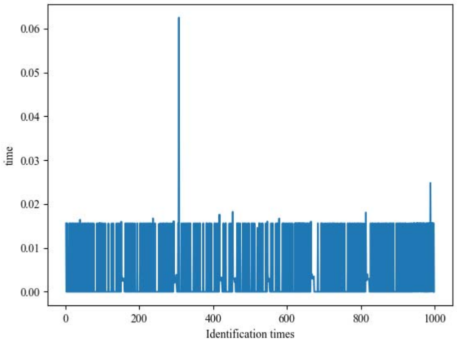


Fig. 7. Automatic computer program identification time test results

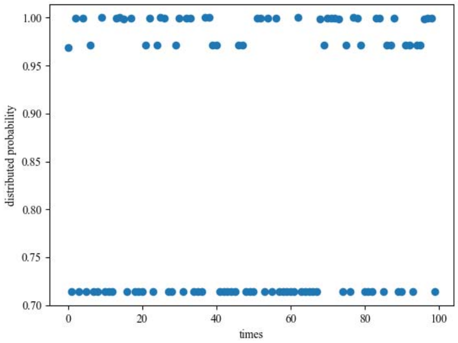


Fig. 9. The distribution probability

V. Conclusion

In this paper, for defending against automatic computer program-based DDoS attacks and abusing of adversarial example generated CAPTCHAs, we proposed a user behavior-based random distribution scheme for adversarial example generated CAPTCHA. We generated two CAPTCHA sets, one is the normal CAPTCHA set, and the other is FGSM generated CAPTCHA set. At the same time, the user behavior is analyzed and associated with a reasonable probability model. According to the probability model, the strong samples generated by normal CAPTCHAs or adversarial examples are alternatively displayed to the user to identify the user's identity in few CAPTCHAs. The experimental results showed that the proposed scheme has an excellent ability to distinguish humans and automatic computer program behaviors.

Acknowledgement

The research was financially supported by National Natural Science Foundation of China (No. 61972366), the Foundation of Henan Key Laboratory of Network Cryptography Technology (No. LNCT2020-A01), the Foundation of Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences (No. KFKT2019-003), Major Scientific and Technological Special Project of Guizhou Province (No. 20183001), and the Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No. 2018BDKFJJ009, No. 2019BDKFJJ003, No. 2019BDKFJJ011).

References

[1] J. Chen, X. Luo, Y. Liu, J. Wang, and Y. Ma, "Selective learning confusion class for text-based captcha recognition," *IEEE Access*, vol. 7, pp. 22 246–22 259, 2019.

[2] N. Yu and K. Darling, "A low-cost approach to crack python captchas using ai-based chosen-plaintext attack," *Applied Sciences*, vol. 9, no. 10, p. 2010, 2019.

[3] K. Qing and R. Zhang, "A multi-label neural network approach to solving connected captchas," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, 2017, pp. 1313–1317.

[4] Y. Hu, L. Chen, and J. Cheng, "A captcha recognition technology based on deep learning," in *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2018, pp. 617–620.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[6] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[8] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 372–387.

[9] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, 2016, pp. 49–54.

[10] P. Tabacof, J. Tavares, and E. Valle, "Adversarial images for variational autoencoders," *arXiv preprint arXiv:1612.00155*, 2016.

[11] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 36–42.

[12] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[13] A. Rozsa, M. Günther, E. M. Rudd, and T. E. Boult, "Facial attributes: Accuracy and adversarial robustness," *Pattern Recognition Letters*, vol. 124, pp. 100–108, 2019.

[14] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 284–293.

[15] Y. Zhang, H. Gao, G. Pei, S. Kang, and X. Zhou, "Effect of adversarial examples on the robustness of captcha," in *2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2018, pp. 1–109.

[16] H. Kwon, H. Yoon, and K.-W. Park, "Robust captcha image generation enhanced with adversarial example methods," *IEICE TRANSACTIONS on Information and Systems*, vol. 103, no. 4, pp. 879–882, 2020.

[17] C. Shi, X. Xu, S. Ji, K. Bu, J. Chen, R. Beyah, and T. Wang, "Adversarial captchas," *IEEE Transactions on Cybernetics*, 2021.