Text Captcha Is Dead? A Large Scale Deployment and Empirical Study

Chenghui Shi[1,*], Shouling Ji[1,*,#,(✉)], Qianjun Liu[*], Changchang Liu[†], Yuefeng Chen[‡], Yuan He[‡], Zhe liu[§], Raheem Beyah[⋆] and Ting Wang[£]

[*]Zhejiang University, [#]Alibaba-Zhejiang University Joint Institute of Frontier Technologies,
[‡]Alibaba Group, [§]Nanjing Unversity of of Aeronautics and Astronautics,
[†]IBM Research, [⋆]Georgia Institute of Technology, [£]Penn State
{chenghuishi, sji, liuqj0522}@zju.edu.cn, Changchang.liu33@ibm.com, {yuefeng.chenyf, heyuan.hy}@alibaba-inc.com,
zhe.liu@nuaa.edu.cn, raheem.beyah@ece.gatech.edu, inbox.ting@gmail.com

ABSTRACT

The development of deep learning techniques has significantly increased the ability of computers to recognize CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), thus breaking or mitigating the security of existing captcha schemes. To protect against these attacks, recent works have been proposed to leverage adversarial machine learning to perturb captcha pictures. However, they either require the prior knowledge of captcha solving models or lack adaptivity to the evolving behaviors of attackers. Most importantly, none of them has been deployed in practical applications, and their practical applicability and effectiveness are unknown.

In this work, we introduce advCAPTCHA, a practical adversarial captcha generation system that can defend against deep learning based captcha solvers, and deploy it on a large-scale online platform with near billion users. To the best of our knowledge, this is the first such work that has been deployed on international large-scale online platforms. By applying adversarial learning techniques in a novel manner, advCAPTCHA can generate effective adversarial captchas to significantly reduce the success rate of attackers, which has been demonstrated by a large-scale online study. Furthermore, we also validate the feasibility of advCAPTCHA in practical applications, as well as its robustness in defending against various attacks. We leverage the existing user risk analysis system to identify potential attackers and serve advCAPTCHA to them. We then use their answers as queries to the attack model. In this manner, advCAPTCHA can be adapted/fine-tuned to accommodate the attack model evolution. Overall, advCAPTCHA can serve as a key enabler for generating robust captchas in practice and providing useful guidelines for captcha developers and practitioners.

KEYWORDS

captcha, adversarial learning, usable study

1 INTRODUCTION

Captcha [2] is a type of *challenge-response test* in computing that aims to distinguish between human and automated programs (machines). Ever since its invention, captcha has been widely used to improve the security of websites and various online applications by preventing the abuse of online services such as phishing, bots, spam, and Sybil attacks. Existing captcha mechanisms can be generally classified as text-based captcha [10], image-based captcha [40, 47], audio-based captcha [24], game-based captcha [42] and others [20, 32]. Among them, text-based captchas have been most popularly studied and applied up to now and in a foreseeable future, which is the focus of this work as well.

Status Quo and Motivation. Over the past decade, a number of attacks have been proposed for automatically recognizing captchas. In early time, many attacks are hard-coded for specific captcha schemes [6, 15, 27], where designing the attacking heuristics and methods requires heavy expert involvement. Thus, many defense strategies that leverage varied fonts and/or background noise are proposed to make captchas more robust and challenging. Later on, to further defend against more generic attacks [5, 16] that target multiple text-based captcha schemes, advanced defense strategies (e.g., through using merged characters and character distortion [6, 44]) are proposed to increase the difficulty of character segmentation and recognition.

Recently, deep learning techniques have shown great success for text recognition [7, 34], which are naturally further being introduced to break advanced captchas. To build up a deep learning based captcha solver, attackers only need to 1) collect a certain number of labeled training data, which can be done by hiring low cost human labors or directly using the existing crowdsourcing platforms, and then 2) use mature deep learning techniques/platforms to train an end-to-end captcha solver. The end-to-end model does not require preprocessing or character segmentation, while taking a whole captcha as input and recognize it directly. As extensively demonstrated, these deep learning-based attacks can achieve significantly good recognition performance on various captchas [33, 45].

To make things even worse, captcha solving platforms [1, 2], which are the commercial sectors aiming at captcha solving, can benefit more from the state-of-the-art deep learning techniques. The previously accumulated captchas can not only help train a powerful solver directly, but also benefit the break of new captcha schemes leveraging transfer learning [45]. As a consequence, many works have declared the death of text-based captchas [5, 36]. However, small business and non-Internet firms/sectors still widely deploy

[1]Chenghui Shi and Shouling Ji are the co-first authors. Shouling Ji is the corresponding author.
[2]For readability purposes, we will write the acronym in lowercase.

the potentially vulnerable text captchas, due to their usability, low cost and high scalability.

Interestingly, recent research finds that Machine Learning (ML) models, especially deep learning models, are vulnerable to *adversarial perturbation* [38], which can be injected into a legitimate image such that the classifier predicts an incorrect label for the image. The adversarial attacks to ML models, on the other hand, provide inspiration to captcha defense against modern attacks, which are usually based on state-of-the-art deep learning techniques. Specifically, to defend against these attacks, recent works that leverage adversarial learning to inject adversarial perturbations to captchas have been proposed [29, 35]. We name these captchas with carefully designed perturbations as *adversarial captchas*. These adversarial captchas are expected to be resilient to deep learning based attacks and meanwhile preserving high usability for human.

Although the idea of adversarial captcha is promising, *there is still a huge gap between the proposed mechanisms and the actual application.* First, they usually assume priorly-known knowledge of the attack models [29, 35], i.e., the adversarial perturbations inserted to the captcha depend on the prior knowledge of the attack model (e.g., its parameters), which is, however, usually unavailable in practice. Second, previous works lack adaptivity to the evolving behaviors of attackers. The attackers, e.g., captcha solving platforms, usually have strong capability and incentives to update their models rapidly in practice. Once the attackers evolve their models, the captcha generation mechanism needs to be adapted as well. Finally, to the best of our knowledge, most, if not all, of the proposed adversarial captcha generation schemes are not actually deployed or evaluated in real world applications, as the time of writing this paper. In other words, whether the idea of adversarial captcha works on the real world platforms is not yet known. Therefore, it is important to study highly secure, adaptive and practically usable adversarial captcha generation and deployment schemes.

Our Method. To fill the blank, by collaborating with a worldwide leading Internet company, we develop an adaptive adversarial captcha generation system, namely advCAPTCHA, and deploy it on a large-scale e-commerce platform with 824 million active individual users and over 1 million commercial users. Leveraging this deployment and application, we conduct a large scale empirical study to evaluate the security, effectiveness, adaptivity, and usability of our proposed mechanism. At a high level, advCAPTCHA first trains a substitute solving model by collected captchas, then employs the substitute model to generate adversarial captchas. After that, by uniquely viewing the answers submitted by the attackers as the queries to their solving models, advCAPTCHA can fine-tune the substitute model to approximate the attack model by learning its captcha recognition behavior.

Contributions. Our main contributions can be summarized as follows.

(1) **An adaptive, practically high-usable adversarial captcha generation system.** We propose advCAPTCHA for generating adversarial captchas against ML based captcha solving attacks in practice. advCAPTCHA can achieve high usability while improving the security of text-based captchas. Moreover, by viewing captcha pictures as queries to the attack model and using these queries to fine-tune the substitute model, advCAPTCHA presents good adaptivity to the evolution of real world attack models.

(2) **Deployment of adversarial captchas on a large-scale online e-commerce platform.** We deployed advCAPTCHA on a large-scale international e-commerce platform. Extensive experiments on the real world platform show that advCAPTCHA can evidently decrease the captcha breaking rate of the underground market by 50%, i.e., adversarial captchas are practically more secure. We also show the usability and robustness of advCAPTCHA in practical deployment.

(3) **Useful guidelines for captcha developers and practitioners.** Through uniquely interacting with the attackers from the underground market, we demonstrate that 1) practical attackers are more inclined to use end-to-end solving models to directly recognize a captcha as a whole; 2) multiple solving models may be simultaneously leveraged by attackers to mitigate the security of adversarial captchas; 3) the adaptivity of captchas in accommodating the updates of attack models is important in practical deployment. These important observations can be leveraged by captcha developers and practitioners to guide the design and deployment of advanced adversarial captchas in the future.

2 BACKGROUND AND RELATED WORK

2.1 ML Model Vulnerability

Recent works have discovered that the existing ML models including neural networks are vulnerable to various attacks. We introduce two types of attacks that will be used in our methods.

Model Evasion Attack. In an evasion attack, an adversary aims to inject carefully crafted perturbations into a legitimate instance such that the classifier predicts an incorrect label for the instance. Such kind of instances are named *adversarial examples*.

In [38], Szegedy et al. first found that DNNs are vulnerable to imperceptible perturbation in the image domain. Then, many works [8, 9, 11, 13, 17, 21, 28, 31] further explored how to generate adversarial examples. Based on the adversary's knowledge, these works can be classified into two categories: white-box attacks [8, 17, 31], where the adversary has full knowledge of the model including the model architecture and parameters, and black-box attacks [9, 11, 13, 21, 28], where the adversary has none, or has limited knowledge of the model.

In the black-box setting, there are two popular methods to generate adversarial examples. The first method depends on the transferability [30] of adversarial examples. An adversary first trains a substitute model, leverages the substitute model to generate adversarial examples and transfers them to attack a target model (i.e., victim model). The second method is the gradient estimation method. Chen et al. noted that transferability is not always reliable and they explored the gradient estimation method as an alternative to the substitute model [11]. In gradient estimation methods [11, 21, 28], the adversary can query the target model by sending any input and receive the predicted class probabilities. Then, the adversary approximates the gradient information of the target model by leveraging the query results. Finally, such estimated gradient information will be further leveraged to generate adversarial examples. However, these methods need a large number of queries to generate an adversarial example.

Model Extraction Attack. In an extraction attack, adversaries aim to leverage the target model predictions to construct another

model as an approximation to the target model. Existing model extraction methods usually consist of the following steps: 1) construct a dataset X where the samples could be randomly selected or carefully designed within the input space; 2) leverage the target model F to predict X and obtain the predicted results Y; 3) consider Y as the ground truth of X and train an approximation model $\hat{F}$.

In [14], Fredrikson et al. developed a method that exploits the confidence values revealed along with predictions. In [41], Wu et al. formalized model extraction from a game-theoretic perspective. In [39], Florian et al. designed equation-solving attacks, which could extract near-equivalent models from multiple models, including SVM, decision trees and neural networks. However, it is inefficient to directly apply these methods to Deep Neural Networks (DNNs) in practice, especially when the attackers have no access to other predicted results (such as confidence scores) than the label information [30].

2.2 Modern Attacks on Captchas

Over the past decade, a number of attacks have been proposed to solve captchas, which are summarized in Table 2. Yan et al. [43] discovered the fatal design errors in many captcha schemes and used the simple pattern recognition algorithms to break captchas with high success rates. Early works [6, 15] typically follow three steps: preprocessing, segmentation and recognition. Specially, they first use heuristic methods to filter background patterns, e.g., line noise in the background of the captcha, then use segmentation techniques, e.g., color filling segmentation, to segment the captchas, and finally use a ML model to recognize the segmentation. Later works [5, 16] combine segmentation and recognition together. They use ML models to score all possible ways to segment a captcha, then find the most likely way as output. Most recently, by leveraging deep learning techniques, Ye et al. [45] used a DNN model to directly recognize captcha images without character segmentation. In particular, they first generate synthetic captchas to learn a base solver and then fine-tune the base solver on a small set of real captchas by leveraging transfer learning. Thus, their method can achieve good recognition performance with a significantly smaller set of real captchas, as compared to previous methods.

In addition to academic research, captcha solving platforms [1, 2] have also used deep neural networks to break captchas. As a commercial sector aiming at captcha solving, they have already accumulated a huge number of captchas generated from a variety of captcha schemes. With these data, they can train a very powerful solver and break a range of captcha schemes.

2.3 Motivating the Design of advCAPTCHA

As stated in Section 2.2, most attacks on text captchas are based on ML techniques, which are vulnerable to various adversarial attacks (recall Section 2.1). Therefore, we aim to exploit the vulnerability of ML models to actively defend against ML based captcha solvers, i.e., generating adversarial captchas to fool the ML based captcha solvers. However, in general practical scenarios, we usually cannot access the attack models. Therefore, it is difficult for us to directly generate adversarial captchas against them. One intuitive method is to first generate adversarial captchas against a substitute model (which is trained by using our own captcha scheme) that performs the same task as the attack model, and then use the generated adversarial captchas targeting the substitute model to defend against the real unknown attack model.

However, this intuitive method may not perform well in practice due to the following two reasons. 1) Through our careful observations in practical applications, we found that attackers usually target at multiple captcha schemes simultaneously instead of only one. Therefore, the substitute model we trained by our own captcha scheme may deviate from the attack model. As a result, the adversarial captchas generated by the substitute model may not demonstrate transferabiltiy against the real attack model. 2) In practice, attackers have strong capability and incentives to update their attack models, which may further degrade the transferabiltiy of the adversarial captchas. In the worst case, the security of the adversarial captchas may degrade to that of the ordinary ones. Therefore, the captcha generation mechanism needs to be updated as well to defend against the evolving attack models.

The security of adversarial captchas closely depends on the similarity between our substitute model and the attack model. Thus, to generate effective adversarial captchas requires an effective channel to observe and understand the real attack model. To address the above challenges, we incorporate the idea of model extraction into constructing the substitute model. In detail, we uniquely view the answers that the attackers submit as queries to their solving models. Then, we use these queries to extract the real attack model. It is worthy noting that we do not directly construct our substitute model from these queries considering the efficiency issues of DNN (recall Section 2.1). Instead, we first train a solving model by using pre-collected captchas and the corresponding labels, then employ such model as a baseline substitute model to generate adversarial captchas, and finally transfer them to defend against the real attack model. Once an update of the attack models is observed, we can query the attack models and fine-tune the baseline substitute model by leveraging these queries. As a result, we update our substitute model dynamically to accommodate the evolving models of the attackers.

2.4 Comparison with Prior Work

We first compare our method with closely related work [29, 35] in Table 2. In [29], Osadchy et al. proposed a method to generate *Immutable Adversarial Noise* (IAN), which is specifically designed for image captchas. It is easier to generate adversarial image captchas compared to adversarial text captchas, since the perturbation space of image captchas is much bigger. Technically, the idea of IAN is simple: injecting enough perturbations to prevent existing image filtering techniques. In [35], Shi et al. proposed an adversarial captcha generation and evaluation system, aCAPTCHA, which can be applied for text captchas. The core idea of aCAPTCHA is to inject perturbation into the frequency domain instead of the space domain. We note that aCAPTCHA does not directly inject perturbations into captchas. Instead, it first injects perturbations into a single character image, and then combines different character images into one captcha, which would limit the efficiency of captchas in practice.

Based on our analysis above, we can summarize the main difference between advCAPTCHA and [29, 35] as below. 1) Benefited by

Table 1: Summary of recent works on solving captchas.

Captcha Solver	Year	Preprocessing	Segmentation	Recognition Method	Training Data
Yan et al. [43]	2007	none	vertical segmentation snake segmentation	pattern recognition	none
Bursztein et al. [6]	2011	image processing	color filling segmentation	SVM KNN	real captchas
Gao et al. [15]	2013	image processing	color filling segmentation	CNN	real captchas
Bursztein et al. [5]	2014	none	cut point detector	KNN	real captchas
Gao et al. [16]	2016	none	Log-Gabor Filter	KNN CNN	real captchas
Ye et al. [45]	2018	GAN	none	CNN	real captchas synthetic captchas

Table 2: Comparison of Adversarial Captchas.

Method	Captcha Scheme	Perturbation Domain	Substitute Model	Adaptivity	Practical Deployment
IAN [29]	image captcha	space	CNN	no	no
aCAPTCHA [35]	text captcha image captcha	frequency space	CNN CNN	no	no
advCAPTCHA	text captcha	space	CRNN	yes	yes

using an end-to-end solver (recognizing captchas without segmentation), our method can directly inject perturbations into the whole captcha, which is more efficient in practice; 2) To the best of our knowledge, we are the first to consider the evolving behaviors of attackers. By incorporating the idea of model extraction, we employed the answers of the attack model to make our substitute model better fit the actual attack model; 3) We are also the first to deploy adversarial captchas on a large-scale online platform. By applying adversarial learning techniques in a novel manner, advCAPTCHA can generate effective adversarial captchas to significantly reduce the success rate of attackers, which has been demonstrated by a large-scale online study. Furthermore, we also validate the feasibility of advCAPTCHA in practical use, as well as its robustness in defending against various attacks.

Our proposed perturbation algorithms in advCAPTCHA are also novel as compared with previous perturbation methods. 1) Compared to previous methods that only construct the substitute model by the training dataset, our method can fine-tune the substitute model to approximate the real attack model, which is more effective and efficient in generating high-quality adversarial captchas; 2) Compared to the gradient estimation method that requires a large number of queries for every adversarial example, our method only needs a small number of queries to fine-tune the substitute model, based on which we can generate unlimited adversarial captchas; 3) Previous methods in model extraction, which aim to construct the approximate model, cannot be applied to extract DNNs, especially when only the label information is available. In comparison, our method starts by building an existing substitute model that performs a similar task with the target model and uses the queries to make the existing model similar to the target model. Therefore, our method is more feasible in practice.

3 DESIGN OF ADVCAPTCHA

In this section, we describe the design details of advCAPTCHA as an effective defense against deep learning based captcha solvers.

3.1 Overview

We provide the workflow of advCAPTCHA in Figure 1, which consists of the following 4 important steps.

Step 1. Training the substitute solving model. In practice, we usually have no prior knowledge of the attack model, thus we need to train an alternative solving model and employ it as a substitute for the real attack model. Then, the substitute model can be leveraged to generate adversarial captchas to defend against the captcha solving attacks. There are two factors we need to concern in this step: model structure and training data. For the model structure, we choose a model which can simultaneously recognize all the characters in a captcha picture without image preprocessing and character segmentation. For the training data, we consider multiple different captcha schemes, in order to enhance the generalizability of our substitute model. With these data and model structures, we can train a substitute solving model. This process is detailed in Section 3.2.

Step 2. Generating adversarial captchas. With the substitute solving model in place, we carefully design perturbation algorithms to generate adversarial captchas. Motivated by the evasion attacks developed in the domain of image classification, we propose several perturbation algorithms which are applicable for perturbing captchas. These algorithms can achieve a good tradeoff between security and usability. After adversarial captchas are generated, we can use them to defend against ML based attacks. This process is described with more details in Section 3.3.

Step 3. Querying the attack model. This step is optional which is only conducted when there is a need to fine-tune the substitute model (e.g., an update of the attack model is observed). In this step, we aim to query the attack models by collecting the captchas sent to

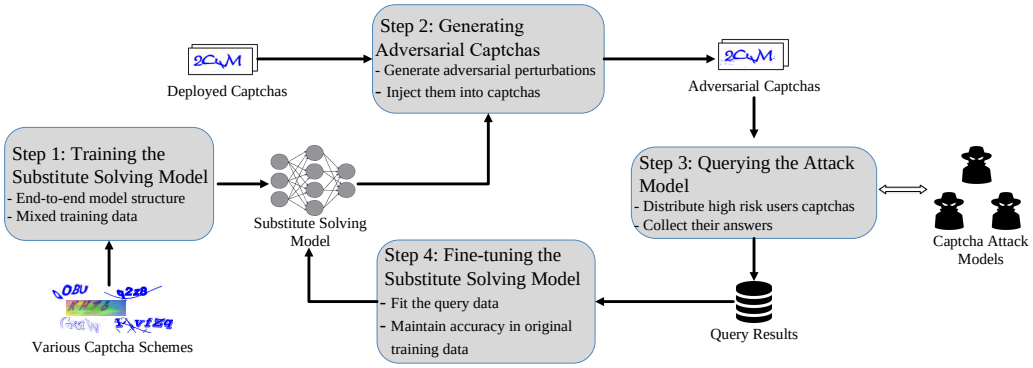


Figure 1: Workflow of advCAPTCHA.

them and the answers they submitted. The key issue in this step is how to exactly distribute captchas to the attackers. Luckily, we can take advantage of the existing user risk analysis system [36], where users who are identified as abnormal users with high confidence are classified as attackers. Finally, we distribute the captchas to these attackers and collect their answers which will be used in the next step. This process is detailed in Section 3.4.

Step 4. Fine-tuning the substitute solving model. Motivated by the model extraction attacks, we propose a novel mechanism to fine-tune our substitute model to approximate the real attack model. Specifically, we fine-tune the substitute model by using a set of labeled captchas that are collected from step 3. Here, we view the answers submitted by the attackers as queries to the attack model, and each answer conveys partial information of the attack model. Model extraction allows us to fine-tune our substitute model to approximate the attack model automatically. In addition, compared to constructing a new substitute model from scratch, fine-tuning the existing substitute model is more efficient. Leveraging this fine-tuning step, we can effectively adapt our defense system after observing an update of the attacker's model. This process is described in detail in Section 3.5.

3.2 Constructing Substitute Solving Model

3.2.1 Model Structure. Our captcha solving model aims to directly recognize the captcha images, which is different from many previous works [5, 16] that solve text-based captchas by following three steps: preprocessing, segmentation and recognition. We use an end-to-end recognition model, where the input is a captcha image without preprocessing and the output is the sequence of characters in the image. Specifically, we employ a Convolutional Recurrent Neural Network (CRNN) [33] as our captcha solving model, which requires less expert involvement and is more convenient for generating adversarial captchas. We defer the detailed structure of CRNN to Appendix A for brevity. Generally, this model first extracts features from the input image, then auto-segments features (full captcha) and recognizes each segmentation (single character), and finally seeks the best combination. This process can significantly reduce the amount of data required to train the model. In our evaluation, we can use 100K samples to train a solver with 92% accuracy for the four-character captcha scheme (which has 62^4 types of label sequences).

3.2.2 Training the Substitute Model. Based on the CRNN model in Section 3.2.1, we train a substitute solving model for the target captcha scheme deployed on real world websites. Note that we aim to build a substitute model which is close to the actual attack model that may be trained by the captchas generated from multiple captcha schemes. Therefore, we train the substitute solver by leveraging the target captcha scheme, as well as other captcha schemes. To collect different captcha schemes, we can either crawl from different websites or generate by open source softwares. Then we can use these mixed data as the training dataset. Each training sample consists of a captcha image (without preprocessing) and an integer vector that stores the character IDs of the captcha. Note that we assign a unique ID to each candidate character of the target captcha scheme. The trained substitute model can then be applied to generate adversarial captchas.

3.3 Generating Adversarial Captchas

3.3.1 Notations. We first present necessary notations in the context of generating adversarial images. We represent a solver as a function $F_\theta(\mathbf{x}) = \mathbf{y}$, where F_θ is the neural network, $\mathbf{x} \in \mathbb{R}^{n \times m}$ is the input image and $\mathbf{y} \in \mathbb{R}^r$ is the corresponding output. Note that for the CRNN model, $\mathbf{y}$ is a label sequence and r is the number of characters in the captcha. Define $\mathbf{x'} \in \mathbb{R}^{n \times m}$ as the adversarial image and let $L(\theta, \mathbf{x}, \mathbf{y})$ be the objective function of F_θ. As in [31][8], we use L_p norm to measure the similarity between $\mathbf{x}$ and $\mathbf{x'}$, which is defined as $L_p = ||\mathbf{x} - \mathbf{x'}||_p = (\sum_{i=1}^{n} \sum_{j=1}^{m} |\mathbf{x}_{i,j} - \mathbf{x'}_{i,j}|^p)^{1/p}$.

3.3.2 Perturbation Algorithm. Recently, many evasion attacks have been proposed to generate adversarial images [4, 46]. However, these methods cannot be directly applied to perturb captchas, based on the following two reasons. 1) Previous methods for perturbing images focus on classification models whose prediction contains the probability of the input for all labels. Then, they can rely on these confidence information to generate adversarial examples, e.g., to increase the probability of the target label or to decrease the probability of the ground truth label. In comparison, for captcha, we leverage an end-to-end model which uses a decode step to compute the output. In particular, our method utilizes the CTC loss to predict the probability of the input for a given label sequence. Thus, in our method, we generate adversarial captchas by decreasing the CTC loss of the ground truth label sequence. 2) We need to pay special

attention to restrict the location of perturbations injected into a captcha to maintain its high usability. In particular, we use a mask to control the position where the perturbations to be injected and the perturbation can be calculated as

$$\mathbf{x}' = \mathbf{x} + \epsilon \cdot ||\Delta_{\mathbf{x}} L(\theta, \mathbf{x}, \mathbf{y}) \cdot \mathbb{M}||_p \tag{1}$$

where $\mathbb{M} \in \mathbb{R}^{n \times m}$ is a 2D matrix named as *mask*, deciding which area of the original captcha the perturbations need to be injected into. $\mathbb{M}$ is a binary matrix where noise can only be injected to the pixel of (i, j) if $\mathbb{M}_{i,j} = 1$. We can easily choose an appropriate $\mathbb{M}$ (such as background areas) to maintain the usability of the adversarial captchas. ϵ is a constant which controls the total amount of injected perturbations. Obviously, a higher ϵ requires more perturbations to be added to the captchas. This method computes the adversarial perturbation by only one step following Equation 1, which is thus named as a *direct method*.

We also consider the *iterative method* to generate adversarial captchas. This method iteratively takes multiple small steps (as shown in Equation 2) to compute the adversarial perturbation while adjusting the direction after each step. Compared to the direct method, the iterative method can result in more misclassification (i.e., more robust captchas from the defense perspective) under the same perturbation level [23].

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \alpha \cdot ||\Delta_{\mathbf{x}} L(\theta, \mathbf{x}, \mathbf{y}) \cdot \mathbb{M}||_p \tag{2}$$

where α is a constant to control the amount of injected perturbations in each step. For an iterative method containing s steps, the total amount of perturbations in the iterative method is $\epsilon = s \times \alpha$. In the iterative method, the default step number s is 10 and the default perturbation level α is one-tenth of ϵ.

At last, in order to generate more randomized adversarial captchas, we propose a *mixture method* to inject various perturbations into a captcha image, where

$$\mathbf{x}' = \mathbf{x} + \sum_{i=1}^{k} GeneratePerturbation(L(\theta, \mathbf{x}, \mathbf{y}), p, \epsilon, \alpha, \mathbb{M}_i) \tag{3}$$

We show the process of the mixture method in Algorithm 1. Basically, we first randomly choose k masks. Note that different perturbations inserted to the same area may conflict with each other, thus we make sure these multiple masks have no spatial overlap. For each mask, adversarial perturbations are generated by different methods (direct and iterative methods) and parameter settings (different distance measures and perturbation levels). Finally, we integrate these perturbations together and inject them to the original captcha.

3.4 Querying the Attack Model

Next, we need to distribute adversarial captchas to attackers and record their answers. It seems paradoxical that if we could figure out which users were attackers, we could simply deny the access of them instead of sending them captchas. We emphasize that 1) this step is optional which is only conducted when the substitute model needs to be fine-tuned; 2) in practice, we can only figure out a small part of high-confidence attackers. By engaging with these high-confidence attackers, we can improve the security of adversarial captchas and eventually defend against more potential attackers. Now, a key challenge here is to recognize attackers.

Algorithm 1: Mixture method.

```
i = 1 ;
while i ≤ k do
    Randomly choose a mask M_i;
    Set distance metric p;
    if direct method then
        Set perturbation level ε;
        Compute perturbation Δx_i by Equation 1;
    if iterative method then
        Set step number s;
        Set perturbation level α;
        Compute perturbation Δx_i by Equation 2;
Δx = Σ_{i=1}^{k} Δx_i;
x' = x + Δx
```

We do not submit adversarial captchas to the commercial services to be solved, since we do not know which commercial services are employed by the attackers. Instead, we take advantage of existing user risk analysis system [36] to identify high-confidence attackers. When a user operation needs to be judged, the front-end web collects the user's behavior information and the environment information and submits them to the risk control engine at the back-end. The follow-up risk control engine will make a decision based on the features of these collected information, such as mouse movement, IP address and Cookie. It can determine whether to block the operation or make a second judgment. We note that the user risk analysis system in our work is different from [36]. In addition to the user environment information and high frequency information (which is similar to [36]), our user risk analysis system pays more attention to user behavior information, e.g., mouse sliding and page detention time, which is more difficult to forge and thus more reliable.

We also worthy to note that identifying high-confidence attackers may not be a real time process. In practice, we send captchas to the risk users (including high-confidence and low-confidence attackers). At the same time, we record their answers and corresponding features which are collected by the risk analysis system. After a few days (e.g., a week) deployment, the domain experts can further analyze the records to identify the high-confidence attackers by leveraging multiple tools and domain knowledge. In this way, we can recognize attackers with high accuracy. We detail this process in Appendix B. Finally, we select answers submitted by identified attackers and view them as queries to the attack model.

3.5 Fine-tuning the Substitute Model

The final step is to fine-tune the substitute model. Our fine-tuning process aims to make the substitute model fit the collected query data, as well as maintain accuracy for the original captcha data. Therefore, we formulate a multi-objective optimization task by optimizing the weighted sum of the two objectives.

$$\underset{\theta}{\text{minimize}} \quad L(\theta, \mathbf{x}_q, \mathbf{y}_q) + \lambda \cdot L(\theta, \mathbf{x}_o, \mathbf{y}_o) \tag{4}$$

where $(\mathbf{x}_q, \mathbf{y}_q)$ is the query data generated by step 3 and $(\mathbf{x}_o, \mathbf{y}_o)$ is the original training data used by step 1. θ are substitute model parameters, e.g., weights and bias. $L(\cdot)$ is the loss function measuring the error in classification, which is the CTC loss in our experiment. λ is the weight to balance the tradeoff between the loss of the new

query data and the original captcha data. In our experiments, we adjust λ dynamically to ensure that a reasonable proportion of the original training data can be correctly recognized. We use the Adam optimizer [22] to effectively solve Equation 4.

Note that in order to improve the efficiency of the fine-tune process, we query the attack model by adversarial captchas rather than ordinary ones. The reasons are as follows. Suppose that $\mathbf{x}_q$ is a captcha used in the query process, $\mathbf{y}' = F_\Theta(\mathbf{x}_q)$ is the prediction of the substitute model for $\mathbf{x}_q$, and $\mathbf{y}''$ is the prediction of the actual attack model for $\mathbf{x}_q$. In the fine-tuning process, we aim to make the substitute model fit the collected query data. In other words, we leverage the difference between $\mathbf{y}'$ and $\mathbf{y}''$ to update the substitute model. If we use ordinary captchas to query the attack model, for most cases, $\mathbf{y}''$ is the same as $\mathbf{y}'$ since both the attack model and the substitute model have high recognition rates for ordinary captchas. Such queries are thus less informative to fine-tune the substitute model. In comparison, leveraging adversarial captchas to query the attack model, which, according to our observations, usually generates different $\mathbf{y}'$ and $\mathbf{y}''$. Therefore, under the same number of queries, compared to ordinary captchas, using adversarial captchas can achieve better model fitting performance. Experiment evaluations in Section 6.5 confirm this conclusion.

Moreover, the accuracy of the updated substitute model may be degraded especially when the labels in the query datasets are deviated from the ground truth. To overcome this issue, we add the second objective of maintaining the accuracy of the original training data to Equation 4. In Sections 5.3, 6.1, 6.2, we verify the effectiveness of the fine-tuning process and show its robustness against various attack models.

4 DEPLOYMENT AND EVALUATION SETUP

4.1 Application Platform

We deploy advCAPTCHA on a large-scale international commercial platform, which provides a broad range of services for corporations and individual users. The current number of commercial users has exceeded 1 million, covering government, consumer, aviation, financial and other fields, and the current number of monthly active users is 824 million.

During online studies, we only conduct experiments on a website provided by the cooperative company for the sake of risk control. This website is a government service website from which some public information (such as traffic violations, driving license, etc.) can be queried. When a user logins account or queries the information, he/she must pass the human machine recognition test. From the user's point of view, he/she needs to slide a bar firstly, which is similar to unlocking the phone screen. Then the user risk analysis system will judge the risk level of the user and determine whether to send him/her a captcha. The captcha scheme deployed on this website is text captcha consisting of four characters, with merged characters while no background noise. According to domain experts in the company, the website is approximately suffered from 100,000 to 1,000,000 attacks per day.

4.2 Deployment

We deploy advCAPTCHA on the above platform as follows. 1) Our cooperative company generates N ordinary captchas each week

Figure 2: Twelve captcha schemes used in our paper. We use all the captcha schemes in the training phase and only one scheme (as shown in the left which is deployed in the real world platform) for the test phase.

by its existing captcha generation algorithm, where $N = 100,000$ in our evaluation for each experiment. 2) According to the algorithms in Section 3, we leverage the substitute model to generate N adversarial captchas from the N ordinary captchas. 3) When there is an authentication request, i.e., the risk analysis system determine he/she is a risky user (including high-confidence and low-confidence), we send the adversarial captcha to the user. At the same time, we record his/her corresponding features and answer. 4) Each week, the domain experts involve and further analyze the stored features to confirm the high-confidence attackers. This process is detailed in Appendix B. 5) When we observe an update of the attack model, e.g., the overall success rate of the attack increases sharply, we use the answers submitted by the high-confidence attackers to fine-tune our substitute model. The whole process of deployment and evaluation continues for 9 months, based on which we formulate our analysis for online user studies in Section 5. Note that, only the results submitted by these high-confidence attackers are used to measure the security of our adversarial captchas.

4.3 Evaluation Setup

In total, we use 12 different captcha schemes in the evaluation, where 7 of them are real captcha schemes provided by our cooperative company and the other 5 schemes are from open source softwares [3]. These schemes can generate captchas that contain different background, different length and other defense factors. We show the example captchas generated by the employed schemes in Figure 2. In the training phase, we use all the 12 captcha schemes to train the substitute model, which can make the solver more general. In the test phase, i.e., online study, we only use one captcha scheme (the actually used captcha scheme by the cooperated company) to generate adversarial captchas, since our objective is to protect the captcha scheme that is deployed on the real website.

5 EVALUATION

In this section, we conduct a large-scale online empirical study to evaluate the performance of advCAPTCHA. Firstly, we analyze the behavior of the identified attackers in online experiments. Then, we show the effectiveness of advCAPTCHA, as well as its robustness under different model parameters and different substitute models.

5.1 Analysis of Attackers (High Risk Users)

In the captcha security scenario, attackers are defined as the ones that solve captchas by using well-trained ML models or other well-designed automatic means. Thus, they may exhibit different behaviors from normal users.

Different Answer Lengths. When providing captcha solving service, a captcha solver usually receives captchas generated from different schemes, e.g., different font styles, different background, and captchas with different lengths. In order to maximize profits,

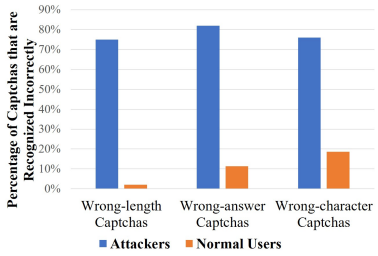


Figure 3: The incorrect recognition rate for different sets of adversarial captchas.

Figure 4: Examples of adversarial perturbations located around a specific character, which would impact human recognition.

captcha solvers in practice usually train a model that can be used for attacking different captcha schemes. Therefore, such model may return answers with different lengths even for a fixed-length captcha scheme. In comparison, for the answers submitted by humans, whether correct or not, the corresponding character length is very likely to be correct, especially for the fixed-length captcha scheme. Thus it is possible that the length distribution of the answers submitted by bots and real users has difference. Therefore, in the experiment, we 1) leverage the substitute model to generate adversarial captchas, 2) select 100,000 adversarial captchas that have abnormal length based on the substitute model's prediction, 3) send these adversarial captchas to the the attackers and the normal users, and 4) measure the lengths of their answers. The results are shown in Figure 3. We observe that the lengths of answers submitted by 76% of the attackers are different from four as defined by our captcha scheme, while only 2% of the normal users return answers with different lengths. This phenomenon also demonstrates that the captcha solvers developed by current attackers are very likely end-to-end models, otherwise attackers would return the same length of answers for the fixed-length captcha scheme.

Different Performance to Adversarial Captchas. ML models are vulnerable to adversarial examples, while in comparison human beings are rarely influenced (this can be assured in the design of adversarial examples, which takes the usability/utility into account). In this experiment, we first leverage the substitute model to generate 100,000 adversarial captchas. Based on the substitute model's prediction, we classified these adversarial captchas into two categories. 1) *Wrong-character captchas:* the captcha that has only one character different from the ground truth. e.g., 2DUA is recognized as 2DUS. 2) *Wrong-answer captchas:* the captcha that has more than one characters different from the ground truth. e.g., Agz3 is recognized as NGE3. Then we send these captchas to the attackers and the normal users, and Figure 3 shows the recognition error rate of the attackers and normal users for adversarial captchas. We observe that attackers are more likely to recognize the adversarial captchas incorrectly, as compared to that of the normal users. In other words, attackers are more vulnerable to adversarial captchas. We also observe that the recognition rate of the normal users for wrong-answer captchas is higher than that for wrong-character captchas. After carefully analyzing the captchas in different settings, we find that L_0-based perturbations added to wrong-character captchas would be located around the character (as shown in Figure 4), which may also affect human recognition. Instead, L_∞-based perturbations would not affect human recognition. This observation motivates us to carefully analyze the location of the injected perturbations in the algorithm. During the practical deployment, we prefer to use a mixture method (using both the L_0 and the L_∞ distance to generate adversarial captchas). We inject L_∞-based perturbations to the entire captcha while only injecting the L_0-based perturbations to the background. In this way, we tend to generate highly usable and secure adversarial captchas.

Table 3: Classification accuracy of 6 model structures. Classification accuracy indicates the SR of the model for ordinary captchas.

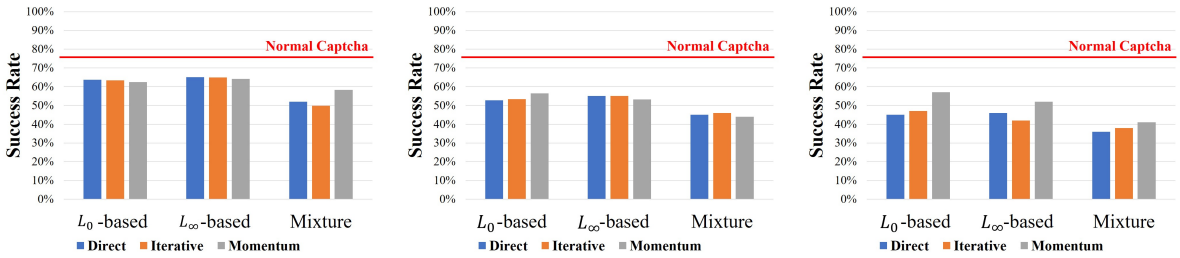
NO.	Model Structure	Classification Accuracy
1	LeNet+LSTM+CTC	92.4%
2	ResNet50+LSTM+CTC	94.1%
3	LeNet+Bi-LSTM+CTC	93.5%
4	ResNet50+Bi-LSTM+CTC	93.2%
5	LeNet	82.3%
6	ResNet50	75.6%

Our analysis above demonstrates that the recognition behavior of the attackers are to some extent distinguishable from that of the normal users. 1) The lengths of the answers submitted by attackers are more likely to be different even for a fixed-length captcha scheme, which indicates current attackers may employ deep learning based solvers. 2) Adversarial captchas tend to have a significant impact on the performance of the attackers while having little impact on the normal users.

5.2 Effectiveness of advCAPTCHA

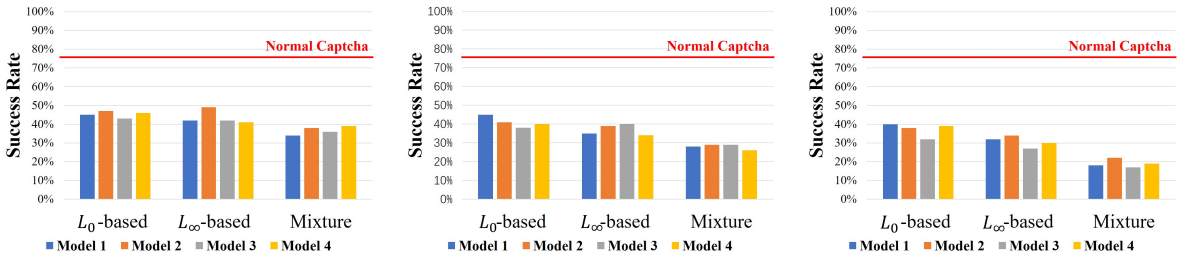
Impact of the Perturbation Algorithm. First, we evaluate the performance of our adversarial captchas under different algorithm parameters in Section 3.3: distance metric L_p, which controls the basic shape of perturbation, and perturbation level ϵ, which controls the total amount of perturbations. We calculate the Success Rate (SR) of attackers to quantify their performance. For distance metrics, we choose L_0, L_∞ and the mixture method (using both the L_0 and the L_∞ distance to generate adversarial captchas); Note that L_0 and L_∞ are the common distance metrics for generating adversarial examples [17, 31]. For perturbation levels, we choose $\epsilon = 0.1, 0.2, 0.3$ for the L_∞-based method and $\epsilon = 25, 50, 100$ for the L_0-based method, according to our usability study in Section 5.4. We also consider the direct and iterative methods, which contains 10 steps and the perturbation level α for each step is one tenth of ϵ (this setting is consistent with all the following experiments). Note that we use the same mask size through all the experiments, which is one quarter of the captcha area. We use Model 1 (in Table 3) as the substitute model and show the corresponding experimental results in Figure 5.

From Figure 5, we have the following observations. 1) Adversarial captchas can effectively reduce the SR of attackers. The lowest SR for adversarial captchas (reduced to 36%) happens when the mixture method is applied with $\epsilon = 100$ for L_0 distance and $\epsilon = 0.3$ for L_∞ distance, and the perturbations are generated by the direct method, given that the SR of attackers for ordinary captchas is 76%. 2) The perturbation level ϵ is the most important factor among different parameters. The SR of the attackers drops noticeably when the perturbation level is raised. Other parameters, e.g., the distance



(a) $\epsilon = 25$ for L_0 and $\epsilon = 0.1$ for L_∞ (b) $\epsilon = 50$ for L_0 and $\epsilon = 0.2$ for L_∞ (c) $\epsilon = 100$ for L_0 and $\epsilon = 0.3$ for L_∞

Figure 5: Performance comparison under different algorithms and perturbation levels for adversarial captcha generation.



(a) only the target captcha scheme (b) all real captcha schemes (c) all captcha schemes

Figure 6: Performance comparison under different model structures and training data for the substitute model.

metric, direct and iterative methods, have less impact on the SR of the attack model. Although the best performance is achieved by the direct method, it cannot always outperform the iterative method. 3) The mixture method, which injects the integration of different perturbations into a captcha, can improve the security of adversarial captchas. We observe that compared with only injecting L_∞ or L_0 perturbations into a captcha, the mixture method, which injects both L_∞-based and L_0-based perturbations into a captcha, can reduce SR with additional 8% on average. One possible reason is that the mixture method can leverage the advantage of both L_∞- and L_0-based methods in black-box scenarios.

Impact of the Substitute Model. Now, we evaluate the performance of our adversarial captchas under different substitute models. There are two factors need to take account in constructing the substitute model: model structure and training data (recall Section 3.2). Thus, we consider several different model structures and training data in the experiment. For the model structure, we use the CRNN model as the target model which contains three components: CNN part, RNN part and CTC part. For each CNN and RNN part, we consider two model structures, ResNet and LeNet for CNN, and LSTM and Bi-LSTM for RNN, respectively. Note that the CTC part is fixed structure. Thus, we use 4 types of models (Model 1-4 in Table 3) as the substitute model. For training data, we collect the captchas generated by different captcha schemes (recall Section 4.3) and divide them into three types: *only the target captcha scheme*, which only contains the captcha scheme deployed on the website, *all the real captcha schemes*, which contains 7 captcha schemes owned by our cooperative company, and *all the real and synthetic captcha schemes*, which contains 5 captcha schemes synthesized by ourselves in addition to the above 7 schemes. Based on our analysis in Figure 5, we choose the best setting as follows. For the distance metrics, we choose L_0, L_∞; for the perturbation levels, we set $\epsilon = 100$ for L_0 and $\epsilon = 0.3$ for L_∞, respectively. We use the mixture method to generate adversarial captchas with L_0,L_∞ distance and direct, iterative methods. We use these settings as default settings for all the following experiments in the paper. The results are shown in Figure 6.

From Figure 6, we have the following observations. 1) The model structure is not an important factor that influences the performance of adversarial captchas. There is no substitute model that always maintain better results than other models. One potential reason is that solving text-based captcha is not complicated, thus simple model structures such as LeNet are sufficiently strong to handle it. In the subsequent experiments, we still employ Model 1 as the substitute model for convenience. 2) Training data is important to the security of adversarial captchas. Compared to the first and second types (only the target captcha scheme and all the real captcha schemes) of training data, when we use the third type (all the real and synthetic captcha schemes) of training data to train a substitute model, the security of adversarial captchas has been substantially improved (where the SR of the attack model is reduced by 10% to 20% as compared to the best result in Figure 5). This demonstrates that using mixed training data to train the substitute model can increase the security of adversarial captchas. One possible reason is that the attackers may train their attack models for various captcha schemes. In other words, the training data that attackers used is also the mixed training data. As a result, the model we trained by the captchas generated using different captcha schemes would be more consistent with the attack model.

Improving Adversarial Examples versus Improving Adversarial Captchas. In [12], the authors proposed a method which integrates the momentum techniques into the iterative method for

Table 4: Usability of advCAPTCHA.

	Adversarial																				Ordinary
	L_0 method without mask				L_∞ method without mask				L_0 method				L_∞ method				Mixture method				
Perturbation level	50	100	200	260	0.1	0.2	0.3	0.4	50	100	200	260	0.1	0.2	0.3	0.4	L_0 L_∞	25 0.1	50 0.2	100 0.3	130 0.4	
Success rate (%)	91.4	75.2	60.6	41.4	89.8	83.2	73.2	64.6	89.7	85.5	75.3	75.9	89.6	83.9	81.4	78.1		90.4	87.5	85.5	75.9	89.8
Average time (s)	7.3	9.5	9.4	9.6	9.4	10.9	10.8	11.7	7.5	8.6	8.9	9.2	7.8	8.4	9.1	9.8		7.6	7.8	9.6	10.4	7.5
Median time (s)	6.4	8.1	9.1	8.7	8.3	9.7	9.6	9.9	6.5	7.6	7.9	8.2	6.7	7.5	8.3	8.6		5.9	6.3	8.2	9.1	6.1

the purpose of escaping from poor local maximum during iterations. They found that the *momentum method* can improve the transferability of adversarial examples, i.e., improving the security of adversarial captchas from the defense perspective, under the same perturbation level. Therefore, we further evaluate the momentum method in adversarial captcha generation. However, from Figure 5, we find that the performance of the momentum method is not consistent with the previous experience [12]. After using the momentum method in captcha generation, the security of the generated captchas cannot be improved in several scenarios. From Figure 5, we observe that under a low perturbation level (e.g., $\epsilon = 25$ for L_0, $\epsilon = 0.1$ for L_∞), the momentum method can reduce the SR of the attack model. In comparison, under a high perturbation level (e.g., $\epsilon = 100$ for L_0, $\epsilon = 0.3$ for L_∞), the momentum method increases the SR of the attack model, which indicates that the momentum method may even have negative effect. One possible reason is the difference between adversarial example generation and adversarial captcha generation. Compared to generating adversarial examples that inject human-imperceptible perturbations, we generate adversarial captchas by injecting human-tolerable perturbations and controlling the areas to inject perturbation. Therefore, it is expected to study the dedicated method to improve the quality of adversarial captchas.

5.3 Impact of Fine-tuning the Substitute Model

Now, we evaluate whether fine-tuning the substitute model by the query of the attack model can improve the security of adversarial captchas.

According to Section 5.2, we employ Model 1 as the substitute model and its training data is generated by all the real and synthetic captcha schemes. In the experiment, we first leverage the substitute model to generate 100,000 adversarial captchas. Then we send these adversarial captchas to identified attackers and collect their answers, which will be further leveraged to find-tune the substitute model, generate corresponding adversarial captchas and send back to the identified attackers for evaluation again. After fine-tuning the model, we observe that the SR of attackers has been further reduced to 12% (as compared to 17% before the fine-tuning process according to Figure 6). This shows that the model extraction technique and the fine-tuning can effectively improve the substitute model to make it more similar to the attacker model, thus increasing the security of the generated adversarial captchas. Moreover, fine-tuning the substitute model can play a greater role when the attackers update their attack models. Therefore, our fine-tuning process can provide adversarial captchas strong adaptivity against the update of attack models. We will further explore the fine-tuning performance of advCAPTCHA under the updates of attack models in the Section 6.1.

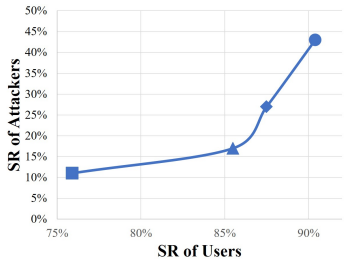


Figure 7: The tradeoff between the SR of users and the SR of attackers under the same parameter settings for the mixture method. From right to left, these four points correspond to $\epsilon = 0.1$ for L_∞ method, $\epsilon = 25$ for L_0 method; $\epsilon = 0.2$ for L_∞ method, $\epsilon = 50$ for L_0 method; $\epsilon = 0.3$ for L_∞ method, $\epsilon = 100$ for L_0 method; and $\epsilon = 0.4$ for L_∞ method, $\epsilon = 130$ for L_0 method, respectively.

5.4 Usability Analysis

Finally, we conduct experiments to evaluate the usability of advCAPTCHA in practical applications. In the experiment, we set five groups of adversarial captchas. 1) *The captchas generated using the L_0 method without mask,* where we inject L_0 perturbations into the captchas and the size of the mask is the full captcha area. Note that the mask is used to restrict the area of injected perturbations, and the full captcha area for the mask means no restriction for injected perturbations. 2) *The captchas generated using the L_∞ method without mask,* where we inject L_∞ perturbations into the captchas and the size of the mask is the full captcha area. 3) *The captchas generated using the L_0 method,* where we inject L_0 perturbations into the captchas and the size of the mask is one quarter of the captcha area. 4) *The captchas generated using the L_∞ method,* where we inject L_∞ perturbations into the captchas and the size of the mask is one quarter of the captcha area. 5) *The captchas generated using the mixture method,* where we inject both L_0 and L_∞ perturbations into captchas and the size of each mask is one eighth of the captcha area. Then, we send these captchas to the normal group (recall Section 5.1), and collect their answers and the time consumption for solving each captcha. We show the collected data in Table 4, including the average successful probability, the average time consumption, and the median time consumption of all the users to finish the corresponding captcha, respectively.

From Table 4, we observe that 1) the mask in the perturbation algorithm is very helpful for maintaining the usability of adversarial captchas. Without using mask in the adversarial captcha generation, the SR of human when recognizing adversarial captchas is obviously lower than that of the ordinary captchas, especially under high perturbation level. In comparison, if we restrict the location of perturbations by the mask, the SR of human when recognizing adversarial captchas is similar to that of the ordinary captchas. 2) The mixture method, which injects various perturbations into a

captcha, almost has no negative impact on human recognition. 3) $\epsilon \leq 0.3$ for L_∞ and $\epsilon \leq 100$ for L_0 are good choices in practice for maintaining the usability of the adversarial captchas. These selected values of ϵ have also shown good performance in balancing the SR of normal users and the SR of attackers (shown in Figure 7). Within this range, the SR of normal users decreases much more slowly than the SR of attackers. Therefore, we consider $\epsilon \leq 0.3$ for L_∞ and $\epsilon \leq 100$ for L_0 in our experiments.

Evaluating advCAPTCHA with volunteer participants who are recruited in the real world would be interesting, which, however, may lead to other limitations. For instance, the scale of users would be much smaller compared to that of the online test in our experiments. Another challenge lies in the selection of participants to prevent bias due to demographics. Therefore, following the best-in-practice evaluation manner, we choose to conduct online experiments for evaluating the usability of advCAPTCHA.

In summary, according to our evaluation, the adversarial captchas generated by advCAPTCHA, have similar usability as the ordinary ones. Combining with the security evaluation of advCAPTCHA in Section 5.2, they together demonstrate that advCAPTCHA is resilient to ML based attacks while preserving high usability for human.

6 FURTHER ANALYSIS OF THE FINE-TUNING PROCESS

In this section, we further analyze the fine-tuning process of advCAPTCHA through offline experiments, including security analysis after attack model updating, robustness in defending against different attack models, the impact of human labor in the query process and the influence of fine-tuning parameters. We analyze the worst-case performance of advCAPTCHA by using Model 1 as the baseline substitute model and Model 4 as the default attack since they present the largest difference in structure. To be consistent with the experiments in Section 5, we generate adversarial captchas by using the mixture method which injects L_0 perturbation with $\epsilon = 100$ and L_∞ perturbation with $\epsilon = 0.3$.

6.1 Security Analysis after Attack Model Update

First, we investigate the fine-tuning process of advCAPTCHA under attack model updating. Specifically, we compare the security of the adversarial captchas generated by the original substitute model and the fine-tuned substitute model. In the experiment, we retrain the attack model by using 100,000 adversarial captchas and the corresponding labels to simulate the process that captcha solving services update their attack models (assuming that the attackers obtain the ground-truth labels of the adversarial captchas using human labor or other means). Then we query the retrained attack model (by sending 100,000 adversarial captchas to it) and use the query results (the answers submitted by the attack model) to fine-tune the substitute model. Finally, we leverage the original substitute model and the fine-tuned substitute model, respectively, to generate adversarial captchas and measure the corresponding SRs of the attack model.

We observe that the SR of the attack model has increased from 7% to 43% when we use 100,000 adversarial captchas to retrain the attack model. After we fine-tune the substitute model by the 100,000 answers submitted by the attack model, its SR has decreased to 13%. These observations demonstrate that advCAPTCHA can adaptively defend against attack model update. In practice, we can fine-tune the captcha generation model regularly, e.g., each week, or when a substantial increase of the overall SR is observed.

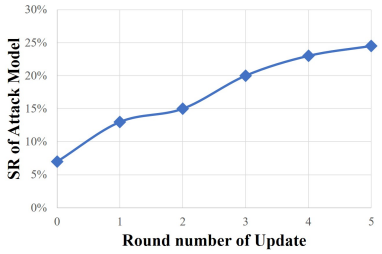


Figure 8: The SRs of the attack model for the adversarial captchas, which are generated by fine-tuned substitute models at different rounds of attack model update.

Next, we conduct five rounds of attack model update to further measure the security of adversarial captchas. Each round follows the same setting listed above, i.e., using 100,000 adversarial captchas to retrain and query the attack model. After each round of update, we obtain a new fine-tuned model. Figure 8 shows the SRs of the attack model for the adversarial captchas, which are generated by fine-tuned substitute models at different rounds of attack model update. From Figure 8, we can observe that the SR of the attack model grows slowly with the number of rounds of attack model update. This observation indicates that the security of adversarial captchas will not decrease rapidly with the update of the attack model. Considering the actual time and cost that the attackers need to upgrade their models, adversarial captchas can greatly improve the security and vitality of the original captcha scheme.

6.2 Robustness of advCAPTCHA in Defending against Various Attack Models

In practice, there may exist multiple attack models that simultaneously break captchas. Here we consider multiple attack models (Model 2, 3, 4, 5, 6 with the corresponding SR for the ordinary captchas shown in Table 3). In the query process, we query each of the five attack models for the same number of times (i.e., we send them the same number of adversarial captchas) ranging from 10K to 100K. Figure 9 shows the performance of advCAPTCHA under the five different attack models. From Figure 9 and Table 3, we observe that 1) the SR of all the attack models are within [41%, 46%], which indicates that advCAPTCHA can defend against various attack models; 2) our fine-tuning process can still effectively reduce the SR of all the attack models (from 43% to 23% on average across models). We also observe that under multiple attack models, the degradation of SR is less than that of a single attack model (as shown in Figure 11). In particular, for 100,000 query rounds, the SR is 13% under the single model attack and 23% under the attack of five models. This result is within our expectation since it is more difficult to use a single substitute model to fit multiple different attack models.

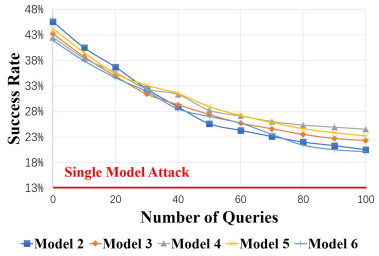


Figure 9: The performance of the fine-tuning process under 5 attack models.

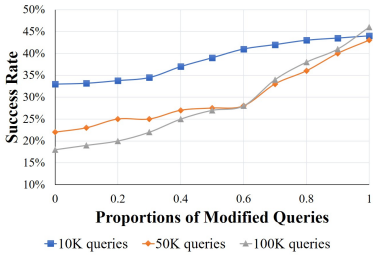


Figure 10: The success rates of the attack model after modifying different proportions of query results to fine-tune the substitute model.

6.3 Impact of Human Answers in Query Process

In the querying process, we use the risk analysis system to identify attackers. However, the queries we collect from the identified attackers may contain results submitted by human, which may be caused by the false positives of the risk analysis system. Here, we investigate whether these human answers would influence the fine-tuning process.

Adversarial captchas are designed for defending against automated solving, while can still be identified correctly by human beings. Thus the answers submitted by human are often correct. Hence, in order to simulate human answers, we change the query answers of the attack model as the correct captchas. Specifically, in the experiment, we first use adversarial captchas to query the attack model and obtain answers from it. Then, we modify the query answers to the correct ones by different proportions (e.g., 0%, 10%, 20%, ..., 100%). Here, different proportions correspond to the different numbers of human answers in all of the queries. Next, we use the modified query data to fine-tune the substitute model. Finally, we leverage the fine-tuned substitute models to generate adversarial captchas, and measure the SR of the attack model.

Figure 10 shows how human answers in the queries affects the performance of adversarial captchas. We can observe that 1) a small proportion of human answers in the queries has no impact on model fine-tuning. The SR of the attack model increases slightly when the proportion of human answers $\leq 20\%$. Moreover, we can query more times to achieve a similar performance to that without human answers if the proportion of human answers $\leq 30\%$; 2) as the proportion of human answers increases, the SR of the attack model grows. However, even under a high proportion of human answers, the SR is similar to that without fine-tuning. These results demonstrate that advCAPTCHA is resistant to human answers in the query process. One reason is that, in the fine-tuning process, we ensure the model accuracy for its original training data (recall Equation 4), which can prevent the substitute model from deviating from the right direction. We further analyze the impact of the parameters in Equation 4 to the fine-tuning process in the following experiment.

6.4 Performance under Different λ

In order to better illustrate the role of the second item $L(\theta, \mathbf{x}_o, \mathbf{y}_o)$ in the fine-tuning objective function (Equation 4), we conduct an experiment to investigate the fine-tuning performance under various values of λ. Specifically, we fine-tune the substitute model under different values of λ and different proportions of human-submitted answers, given the total number of queries is 100,000. We show the corresponding results in Table 5.

Table 5: Original accuracy (classification accuracy on the original training data) and success rate of the adversarial captchas generated by the fine-tuned models under different λ and proportions of modified query results.

λ	Proportion	Original Accuracy	SR
0	0	80.1%	40.3%
	0.1	81.2%	50.5%
	0.7	83.2%	70.2%
1	0	83.7%	25.5%
	0.1	85.3%	32.5%
	0.7	90.2%	42.8%
10	0	87.3%	41.3%
	0.1	90.3%	43.6%
	0.7	92.1%	51.5%

From Table 5, we have the following observations. First, when only 0% or 10% of the answers are submitted by human, the second item in the fine-tuning objective function can prevent the degradation of model accuracy on the original data and meanwhile can decline the SR of the attack model. Second, it is important to select an appropriate value of λ. A large value of λ would easily make the model pay too much attention to the original training data in the fine-tuning process, instead of fitting the attack model. In practice, according to our experience, we found that dynamically adjusting its value so that the model accuracy on the original training data is around 90% can achieve a good trade-off between fitting the attack model and preventing model crash.

6.5 Impact of Captchas in the Query Process

According to the analysis in Section 3.5, we know that using adversarial captchas (instead of ordinary captchas) can accelerate the fine-tuning process. Here, we further verify this conclusion using empirical evaluations. In the experiment, we first send both the ordinary captchas and the adversarial captchas to the attack model, respectively. The number of captchas ranges from 10K to 100K for each type of captcha. Then we fine-tune the substitute models by using the answers corresponding to the ordinary and adversarial captchas, respectively, and obtain two new substitute models. Finally, we generate adversarial captchas by leveraging the two fine-tuned substitute models to defend against the attack model. Figure 11 shows the relationship between the SR of the

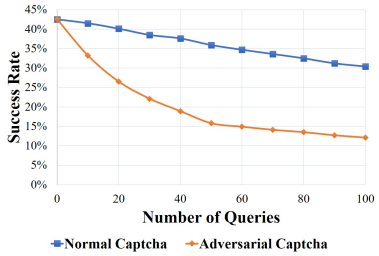


Figure 11: Impact of captchas for fine-tuning the substitute model.

attack model and the number of queries. We observe that when fine-tuning the substitute model, using ordinary captchas requires more queries (about 10 times) to achieve similar performance as that of the adversarial captchas. This is because both the attack model and the substitute model have high recognition rates for ordinary captchas. For a captcha, if the query result (the output of the attack model for the captcha) is similar to the output of the substitute model, it is less informative for fine-tuning the substitute model. Therefore, it is recommended to leverage adversarial captchas in the fine-tuning process.

7 DISCUSSION AND LIMITATION

In the daily operation of websites, since the high-confidence attackers can be relatively easily blocked by a risk analysis system, we setup our objective of developing advCAPTCHA as to defend against low-confidence attackers. Certainly, it is impossible to obtain the perfect ground truth of attackers in practice. Therefore, in the deployment and evaluation, to make our experiments and results more convincing, following a best-in-practice manner, we leverage the high-confidence attackers to evaluate the security enhancement of advCAPTCHA, i.e., in our evaluation period, before directly blocking them, we leverage them to evaluate advCAPTCHA first. Moreover, in our experiments, these high-confidence attackers are also manually confirmed by the domain experts (shown in Appendix B). Thus, to some extent, the high-confidence attackers can be considered as known attackers. It is reasonable to believe that our experimental results can reflect the performance of advCAPTCHA under actual attacks.

We would like to highlight the generalizability of advCAPTCHA, as well as its advantages over existing works. advCAPTCHA and other captcha schemes are not competitive but complementary. The key objective of adversarial captchas is to increase the difficulty of recognition for computers. Therefore, advCAPTCHA can be extended to captcha schemes that perform similar recognition tasks, e.g., image captchas. Specifically, there are two steps for generalizing advCAPTCHA to other captcha schemes: 1) replacing the substitute model (the CRNN model in Section 3.2) with the model that could recognize the target captcha scheme, e.g., CNN for recognizing image captchas; 2) designing an appropriate perturbation mechanism to inject noise into the target captcha images. In fact, we have implemented additional evaluation of advCAPTCHA for click-based captcha schemes where the users are asked to click a word in a picture according to the prompt. Due to the space constraints, we do not include this design in the paper. Experimental results verify the effectiveness of advCAPTCHA in this scenario as well, which further demonstrates its generalizability in various captcha generation schemes and application scenarios.

advCAPTCHA is also a potential enhancement to the user risk analysis system, e.g., Google noCaptcha. This popular captcha scheme can provide a great user experience, as normal users' operations are not interrupted in most situations. However, the security of the overall system has not been improved as compared to traditional captchas, since the adversaries can simply ignore the risk analysis in the first phase and only focus on captcha solving in the second phase. In this case, our advCAPTCHA can degrade the performance of captcha solvers through dynamically generating adversarial captchas.

The usability of advCAPTCHA can be affected by the ordinary version of a captcha scheme. The failure rates of advCAPTCHA for normal users highly depend on the ordinary version of the captcha scheme itself. Therefore, advCAPTCHA cannot enhance a captcha scheme if it is already poorly designed. Moreover, similar to many other captcha schemes which perform recognition tasks, advCAPTCHA cannot defend against the captcha solving services that employ human users. We believe this is acceptable, since captchas are designed to be recognized by human beings.

Limitations and Future Works. As an attempt to design secure, adaptive and practically high-usable adversarial captchas, we believe our approach can be improved in many perspectives, especially when considering the practical deployment and application. We discuss the limitations of this work along with future directions below.

Generate more secure and usable adversarial captchas. In our adversarial captcha generation algorithm, the total amount of perturbations is a key parameter to control balance between the security and usability of adversarial captchas. How to generate more secure and usable adversarial captchas with less perturbation is a meaningful, yet challenging, question. Meanwhile, given an amount of perturbation, how to elegantly distribute it over an adversarial captcha (instead of directly injecting perturbations into the background areas) is another interesting topic to study.

Generate more effective query data. In the paper, we only use ordinary captchas and adversarial captchas to query the attack model. In fact, we can deliberately construct well-designed data to accelerate the model extraction process. For example, according to previous study [39], using data closer to the model classification boundary can benefit the model extraction process. Therefore, how to generate more effective query data for the captcha solvers would be an interesting future work.

Leverage multiple substitute models. In Section 6.2, we only use a single substitute model to generate adversarial captchas. However, when facing several different attack models, it is difficult to use a single substitute model to fit all these models. As an interesting future work, we plan to generalize our method to incorporate multiple substitute models so that they can better approximate a broad range of attack models in practice.

8 CONCLUSION

In this paper, we present the design and evaluation for advCAPTCHA, which is the first large-scale deployment of adversarial captchas on an international e-commerce platform, in order to defend against

ML based captcha solving attackers. advCAPTCHA fills the gap between the scientific innovations of adversarial captchas and its adoptions in practical applications. Specifically, we propose novel perturbation methods to generate adversarial captchas to degrade the performance of captcha solvers. We further incorporate the idea of model extraction to make our substitute model fit the actual attack model. This strategy can automatically adapt our defense system after observing an update of the attacker's model. Extensive experiments on the real world platform demonstrate the effectiveness of advCAPTCHA which can significantly reduce the success rate of actual attackers.

ACKNOWLEDGEMENT

We would like to thank our shepherd Jelena Mirkovic and the anonymous reviewers for their valuable suggestions for improving this paper. This work was partly supported by NSFC under No. 61772466, U1936215, and U1836202, the National Key Research and Development Program of China under No. 2018YFB0804102, the Zhejiang Provincial Natural Science Foundation for Distinguished Young Scholars under No. LR19F020003, the Zhejiang Provincial Natural Science Foundation under No. LSY19H180011, the Zhejiang Provincial Key R&D Program under No. 2019C01055, the Ant Financial Research Funding, and the Alibaba-ZJU Joint Research Institute of Frontier Technologies. Ting Wang is partially supported by the National Science Foundation under Grant No. 1910546, 1953813, and 1846151.

REFERENCES

[1] [n. d.]. https://www.deathbycaptcha.com.
[2] [n. d.]. http://www.captchatronix.com.
[3] [n. d.]. https://pypi.org/project/captcha/.
[4] N. Akhtar and A. Mian. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access* (2018).
[5] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C. Mitchell. [n. d.]. The End is Nigh: Generic Solving of Text-based CAPTCHAs. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*.
[6] Elie Bursztein, Matthieu Martin, and John Mitchell. [n. d.]. Text-based CAPTCHA Strengths and Weaknesses. In *CCS '11*.
[7] Michal Busta, Lukas Neumann, and Jiri Matas. 2017. Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.
[8] Nicholas Carlini and David Wagner. [n. d.]. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP) 2017*.
[9] N. Carlini and D. Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. *AISec* (2017).
[10] Kumar Chellapilla and Patrice Y. Simard. 2005. Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In *Advances in Neural Information Processing Systems 17*. MIT Press.
[11] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. [n. d.]. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks Without Training Substitute Models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec '17)*.
[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
[13] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2017. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945* (2017).
[14] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. [n. d.]. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *CCS '15*.
[15] Haichang Gao, Wei Wang, Jiao Qi, Xuqin Wang, Xiyang Liu, and Jeff Yan. [n. d.]. The robustness of hollow CAPTCHAs. In *CCS '13*.
[16] Haichang Gao, Jeff Yan, Fang Cao, Zhengya Zhang, Lei Lei, Mengyun Tang, Ping Zhang, Xin Zhou, Xuqin Wang, and Jiawei Li. 2016. A Simple Generic Attack on Text Captchas. In *NDSS 2016*.
[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. [n. d.]. Explaining and harnessing adversarial examples. In *ICLR 2015*.
[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* (1997).
[20] K. Hwang, C. Huang, and G. You. [n. d.]. A Spelling Based CAPTCHA System by Using Click. In *International Symposium on Biometrics and Security Technologies 2012*.
[21] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. [n. d.]. Black-box adversarial attacks with limited queries and information. In *ICML 2018*.
[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[23] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. [n. d.]. Adversarial examples in the physical world. *ICLR 2017* ([n. d.]).
[24] Jonathan Lazar, Jinjuan Feng, Tim Brooks, Genna Melamed, Brian Wentz, Jonathan Holman, Abiodun Olalere, and Nnanna Ekedebe. [n. d.]. The SoundsRight CAPTCHA: an improved approach to audio human interaction proofs for blind users. In *CHI 2012*.
[25] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* (1998).
[26] Keaton Mowery and Hovav Shacham. [n. d.]. Pixel perfect: Fingerprinting canvas in HTML5. ([n. d.]).
[27] Yoichi Nakaguro, Matthew Dailey, Sanparith Marukatat, and Stanislav Makhanov. [n. d.]. Defeating line-noise CAPTCHAs with multiple quadratic snakes. *Computers Security 2013* ([n. d.]).
[28] N. Narodytska and S. P. Kasiviswanathan. 2017. Simple black-box adversarial perturbations for deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.
[29] Margarita Osadchy, Julio Hernandez-Castro, Stuart Gibson, Orr Dunkelman, and Daniel Pérez-Cabo. 2017. No bot expects the DeepCAPTCHA! Introducing immutable adversarial examples, with applications to CAPTCHA generation. *IEEE Transactions on Information Forensics and Security* (2017).
[30] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. [n. d.]. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*.
[31] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. [n. d.]. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P) 2016*.
[32] S. K. Saha, A. K. Nag, and D. Dasgupta. 2015. Human-Cognition-Based CAPTCHAs. *IT Professional* (2015).
[33] B. Shi, X. Bai, and C. Yao. 2017. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).
[34] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. 2016. Robust scene text recognition with automatic rectification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[35] Chenghui Shi, Xiaogang Xu, Shouling Ji, Kai Bu, Jianhai Chen, Raheem A. Beyah, and Ting Wang. 2019. Adversarial CAPTCHAs. *CoRR* abs/1901.01107 (2019). arXiv:1901.01107
[36] S. Sivakorn, I. Polakis, and A. D. Keromytis. [n. d.]. I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs. In *2016 IEEE European Symposium on Security and Privacy (EuroS P)*.
[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.
[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[39] Florian Tramer, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. [n. d.]. Stealing machine learning models via prediction apis. In *Usenix Security 2016*.
[40] H. Weng, B. Zhao, S. Ji, J. Chen, T. Wang, Q. He, and R. Beyah. 2019. Towards understanding the security of modern image captchas and underground captcha-solving services. *Big Data Mining and Analytics* 2, 2 (2019), 118–144.
[41] X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. [n. d.]. A Methodology for Formalizing Model-Inversion Attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*.
[42] Y Xu, G Reynaga, Sonia Chiasson, J.-M. Frahm, Fabian Monrose, and Paul van Oorschot. [n. d.]. Security and Usability Challenges of Moving-Object CAPTCHAs: Decoding Codewords in Motion. In *Usenix Security 2012*.
[43] J. Yan and A. S. E. Ahmad. [n. d.]. Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms. In *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*.

[44] Jeff Yan and Ahmad Salah El Ahmad. [n. d.]. A Low-cost Attack on a Microsoft Captcha. In *CCS '08*.
[45] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. [n. d.]. Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. In *CCS '18*.
[46] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems* (2019).
[47] Bin B Zhu, Jeff Yan, Qiujie Li, Chao Yang, Jia Liu, Ning Xu, Meng Yi, and Kaiwei Cai. [n. d.]. Attacks and design of image recognition CAPTCHAs. In *CCS '10*.

A CRNN MODEL STRUCTURE

Here, We describe each part of CRNN in detail. CRNN mainly consists of three parts: *Convolutional Neural Network (CNN) part* which extracts a feature sequence from the input image, *Recurrent Neural Network (RNN) part* which predicts a label distribution for each feature representation and *Connectionist Temporal Classification (CTC) part* which translates all label distributions into the final label sequence, as shown in Figure 12.

CNN Part. The component of convolutional layers is constructed by taking the convolutional and pooling layers from a standard CNN model (fully-connected layers are removed). Such component is used to extract a sequential feature representation from an input image. Before being fed into the network, all the images need to be scaled to the same height (width could be different). Then a sequence of feature vectors $\mathbf{f}_1, ..., \mathbf{f}_r$ is extracted from the feature maps produced by the component of convolutional layers, which is the input for the recurrent layers. The value of r is related to the width of the input image. Obviously, a wider input image will induce a longer feature sequence. Note that our method can accommodate any type of CNN models, e.g., Lenet [25], ResNet [18] and Inception [37]. Table 6 shows the structure of our solver which has four convolutional layers, where each of the convolutional layer is followed by a max-pooling layer. This structure is simple and effective in practical applications which can achieve high accuracy with a small amount of training data and a short inference time.

RNN Part. The recurrent layers predict a label distribution $\mathbf{l}_t$ for each feature representation $\mathbf{f}_t$ in the feature sequence $\mathbf{f}_1, ..., \mathbf{f}_r$. The traditional RNNs are vulnerable to the vanishing gradient problem, while the Long Short Term Memory (LSTM) method is more effective in tackling this issue [19]. The special design of LSTM can capture long-range dependencies, which often occur in image-based sequences. Table 6 shows the structure of our RNN part which has two LSTM layers.

CTC Part. CTC layer decodes the predictions made by the RNN part into a label sequence. The objective of the CTC part is to find the label sequence with the highest probability conditioned on the predicted label distributions. By leveraging the CTC part, we only need captcha images and their corresponding label sequences to train the network, without requiring labeling positions of individual characters.

B GROUND TRUTH CONSTRUCTION

Now, we describe how domain experts label a user as an attacker. There are many kinds of features or signals that experts can utilize. We introduce three categories of features in the following.

1) *User Environment Information.* Normal users use browsers to browse websites, while attackers often use scripts to crawl information. Thus, their browser attributes are different. The web-front

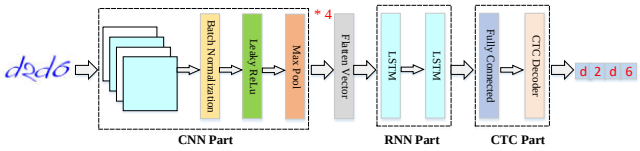


Figure 12: Architecture for our substitute solving model.

Table 6: CRNN model details

Layer Type	Configurations
Input	30 × 100 gray-scale image
Convolution 1	filters=64, kernel=3,strides=1
BatchNorm	
Max Pooling 1	kernel=2, strides=2
Convolution 2	filters=128, kernel=3,strides=1
BatchNorm	
Max Pooling 2	kernel=2, strides=2
Convolution 3	filters=128, kernel=3, strides=1
BatchNorm	
Max Pooling 3	kernel=2, strides=2
Convolution 4	filters=64, kernel=3,strides=1
BatchNorm	
Max Pooling 4	kernel=2, strides=2
LSTM 1	hidden units=256
LSTM 2	hidden units=256
Fully Connect	
CTC	

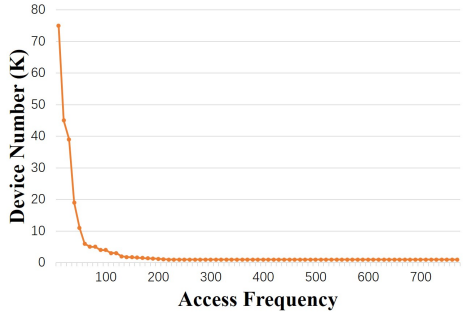


Figure 13: The statistic of the device number with respect to access frequency.

client of the user risk analysis system executes a series of checks for collecting the user environment information, e.g., web driver, user-agent and Cookie. The experts can detect various items, for example, whether a user uses an automate web driver, whether the user-agent contains the complete information, or is misformated. Moreover, in order to obtain real browser attributes, web fingerprint techniques, e.g., webGL fingerprinting [26], are employed. The discrepancies between fingerprints with the reported user-agent are also important factors considered by the experts.

2) *High Frequency Information.* A significant characteristic of machine traffic is the high access frequency compared to the normal users. The experts can analyze the access frequency for different entities, e.g., IP, account and device, to find attackers. Moreover, they can set a strict threshold, e.g., 100 times per second, to avoid misclassification. Figure 13 shows the statistic of the device number with respect to the access frequency during a week. We can find many devices access too frequently.

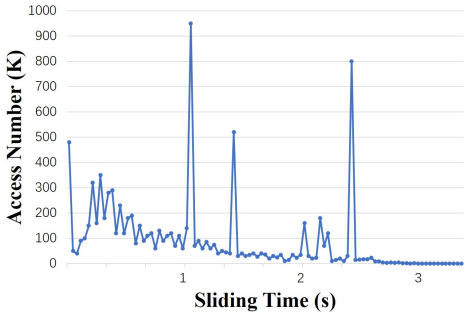
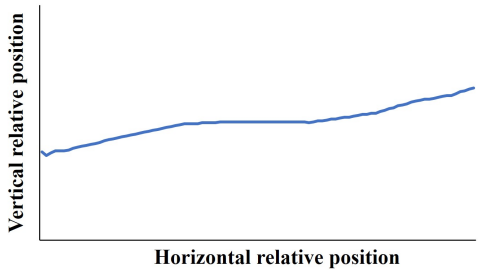


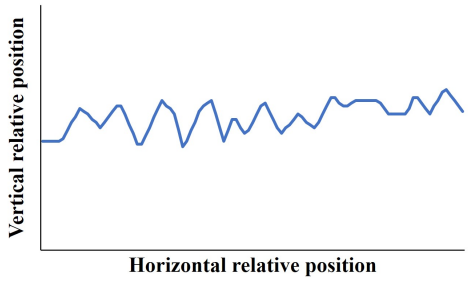
Figure 14: The statistic of the access number with respect to sliding time.

3) *User Behavior Information.* As we stated in Section 4.1, a user needs to slide a bar for the authentication. Such sliding behavior information is an important feature to identify attackers. Figure 15 shows the the difference between machine mouse trajectories and human mouse trajectories. From Figure 15, experts can classify users whose mouse trajectories are significantly different from normal users as attackers. Moreover, they can even simply analyze the sliding time. Figure 14 shows the statistics of the device number with respect to sliding time during a week. Obviously, those devices which slide the bar too fast, e.g., less than 0.1 second, are machine traffic. In addition, even the attackers realize that the sliding time is a detection variable and deliberately extend the sliding time, we can still detect them leveraging other advanced means. For instance, they access too many times in a short time and their sliding time are similar, which can also be observed in Figure 14.

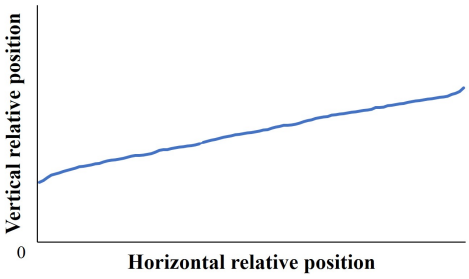
The sophisticated attackers may try to avoid detection by obfuscating some features, i.e., when they aware of mouse movement, they can deliberately simulate human trajectories. However, it is difficult for the attackers to simulate human behaviors in all possible features. Actually, there are many other features we have not mentioned in the paper due to the confidentiality requirements of the company. By analyzing various features, experts can label a attacker with high confidence.



(a) normal trajectory



(b) abnormal trajectory 1



(c) abnormal trajectory 2

Figure 15: Mouse trajectories during sliding a bar. Human trajectory in 15(a) is smooth while machine trajectory is either rough in 15(b) or linear in 15(c).