When Deep Learning Meets Captcha: A Large Scale Study Showing That Captcha Should Be Abandoned

***Abstract*—CAPTCHA is a standard secure mechanism to automated tell computers and humans, and is widely applied to prevent against malicious bot programs. In this study, we carry out a large scale systematic study on current text-based Captchas based on complex deformable characters that are augmented with both anti-segmentation and anti-recognition techniques. Unlike prior fine research that focus on the text-based Captchas with the same or similar styles, we present a novel generic hierarchical attack on different styles of Captchas. Our approach is able to translate these various styles of Captchas to a unique one using conditional Generative Adversarial Networks (cGANs). Using our CNN recognition engine, we can accurately identify the translated Captchas. We thoroughly evaluated our approach using real-world Captchas, and found all current text-based Captchas are vulnerable to our automated attack with an average speed of 224.35ms on an ordinary server (with a 3.2-GHz Inter Xeon CPU with 100-GB RAM and a TITAN Xp GPU). We also discovered that, the Captcha with complex noisy background do not defend against segmentation under our attack. This is demonstrated by the fact that we are able to clean out any sophisticated background from the Captcha image. Since our attack can break a widely of current text-based Captchas, this paper calls the community to revisit the risks of using text-based Captcha to defend against malicious bots.**

Keywords*—*Captcha, Security, Generic Attack, Text-based

I. Introduction

A CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Human Apart) is a automated test that humans can pass but computer programs cannot [46]. It provides a effective approach for automatically distinguishing humans from computer systems, and therefore is used to defend against automatic spam, registration or malicious bots [43, 45].

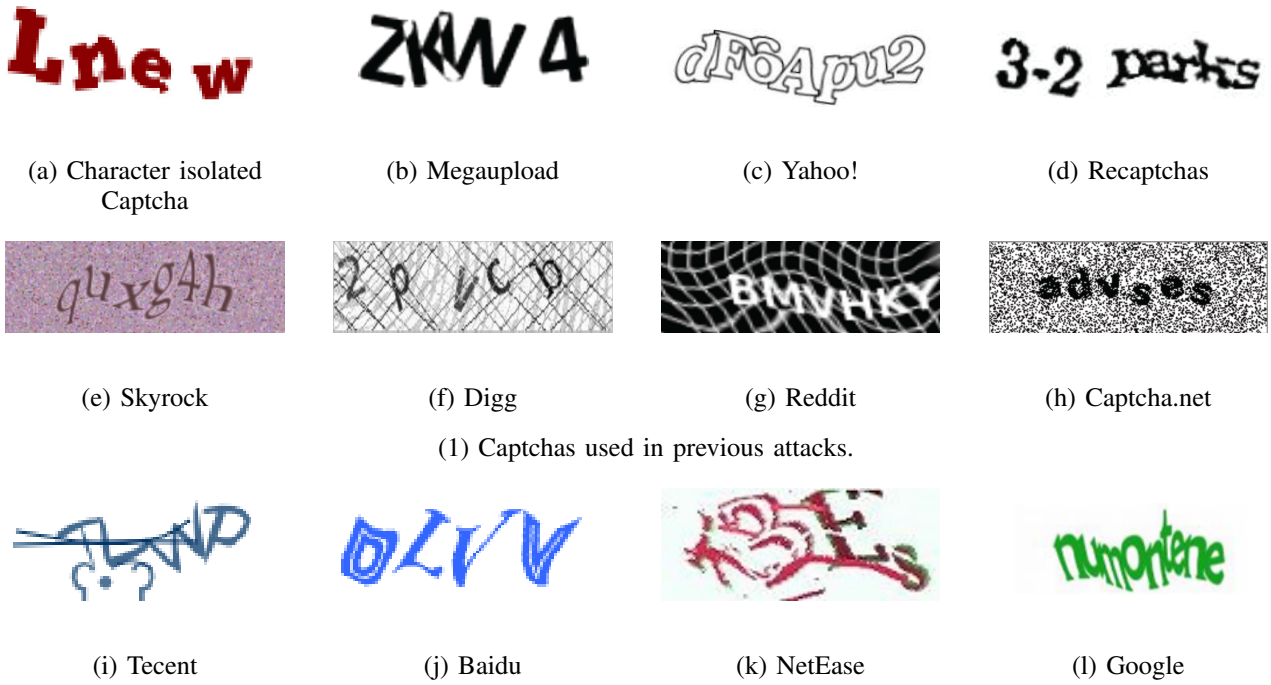
The most widely deployed CAPTCHA is the so-called text-based scheme [51], which mainly consists of distorted English letters and Arabic numerals. The popularity of this scheme is due to its obvious advantages [10, 11]: (1) most people around the world can recognize English letters and Arabic numerals; (2) the space of the text-based Captcha are huge so that the brute-force attack can be defeated. Given its pervasive usage, a security breach of the text-based Captcha could lead to serious consequences.

The robustness of text-based Captchas is the significant concern in the research communities. Over the past decade, researchers have uncovered a number of ways to recognize text-based Captchas. Many fine prior researches just focus on attacking an unique Captcha scheme [14, 15, 35, 50]. This limits their applicability. Recently the generic attacks have been proposed by Gao *et al.* [16] and Bursztein *et al.* [5, 8]. They claim that they can break a wide range of text-based Captchas using a generic attacking method. However, these defeated Captchas possess relative simple noisy background or uniform font style. Although the security of text-based Captchas have been proven frail, many companies such as Google, Microsoft and Baidu still use such scheme as current text-based Captcha scheme have more complex background or distorted characters. This makes previous attacks invalid. Recent studies [9, 44] also demonstrate that text-based Captcha is still a secure mechanism.

The key factor of previous attacks on text-based Captcha lies in the difficulty of segmenting characters [11]. Therefore, the basic design principle of text-based Captchas should be anti-segmentation. To this end, current text-based Captchas with more complex noisy background and distorted characters have been proposed and deployed by many companies. In order to increase the difficulty of finding where each character is, such Captcha shorten the distance between characters. All attacks proposed above were invalid due to cannot successfully segment the characters. It is therefore driving us wondering: is current sophisticated Captcha scheme as secure as it is expected? This fundamental question precipitated our study.

In this paper, we present a novel generic attack on current text-based Captchas using deep learning technology. Our attack employs a variant of the conditional generative adversarial networks (cGANs) [25] to transform the distorted Captchas to the regular ones. To do so, we developed a hierarchical generative model using the cGANs. The hierarchical generative mode consists of three components: background removing model, segmentation model and regularization model. The background removing model is used to clean out the complicated noisy background of Captchas and outputs the clean Captchas. The segmentation model aims to expand the space between the adjacent characters of the clean Captchas. This step outputs the Captchas with larger inter-character space. Note that until this step, the characters of Captchas are still distorted. Such distorted characters can be translated to the regular ones by our regularization model. At last, the transformed regular Captchas are recognized by our recognition engineer trained using the Convolutional Neural Network (CNN).

We thoroughly evaluate our approach using real-world captchas collected from some websites. We show that our approach is effective in recognizing current text-based Captchas and as a results, we can defeat almost all current text-based



(a) Character isolated Captcha (b) Megaupload (c) Yahoo! (d) Recaptchas

(e) Skyrock (f) Digg (g) Reddit (h) Captcha.net

(1) Captchas used in previous attacks.

(i) Tecent (j) Baidu (k) NetEase (l) Google

(2) Current Captchas deployed in some popular websites.

Figure 1. This shows the text-based Captcha schemes used in prior works and current Captchas deployed in some famous websites. The scheme in (a) shows the character isolated Captcha that is deployed originally.

captchas with a success rate range from XX% to XX%. We demonstrate that, the sophisticated noisy background cannot offer stronger protection in term of anti-segmentation under our attack. Our finding suggests that text-based captchas are insecure under the age of artificial intelligence.

Contributions This paper makes the following specific contributions:

In order to design a more security Captcha scheme, currently Captchas deployed by many companies

II. Background

A. Text-based Captchas

CAPTCHA is also called inverse Turing test [37], and aims to determine whether or not the user is human. The most popularity of Captchas is text-based Captchas. Initially, they were composed of deformed English letters and Arabic numerals which human can recognize while computers cannot. The usage of English letters and Arabic numerals makes text-based Captchas being welcomed world-wide [10, 11]. Due to such advantages, text-bsaed Captchas have been widely used in various of Internet security access tasks [46]. These tasks include defending against malicious scraping web contents, preventing automatic registration of free accounts for spam posting or forums, or mitigating the impact of DDoS attacks.

1) Captcha Security Features: To summary out a representative security features, we collected a wide range of Captchas that had ever been or being used by the top 50 most popular web sites which are listed by Alexa [1]. Figure 1 gives some representative examples of our collected Captchas [2]. It obviously shows that current text-based Captchas have more complex security features than previous Captchas. These features can be classified into two categories: anti-segmentation and anti-recognition features. The anti-segmentation feature aims to increase the difficulty of character segmentation. In order to implement such feature, the Captcha is needed to be confused on the whole with sophisticated background, extra connecting lines and overlapping characters. Thus, the anti-segmentation features can be regarded as the global security features as it makes the Captcha complex on the whole. In contrary, the anti-recognition features belong to local features. It can defense against automated recognizing characters by changing the font style of characters. We list the two fratures as follows:

Anti-segmentation features 1. Sophisticated Noisy Background Try to make the background have little difference with the text for confusing the solver. **2. Connection Lines** Add extra lines on the text to prevent solver from automatic finding the single character. **3. Overlapping** Decreasing the space between the adjacent characters to make them collapsed.

Anti-recognition features 1. Character Set Which charset the text-based Captchas scheme uses. Some schemes only include English letters while some schemes consist of both English letters and Arabic numerals. **2. Font Style** Using multiple font styles such as solid font, hollow font or font compromise of dots. **3. Font Size** Using random font size when generating Captcha. **4. Font Color** Using variable font colors.

[1] http://www.alexa.com/topsites

[2] Some previous used Captchas come from the related works [8, 15, 16].

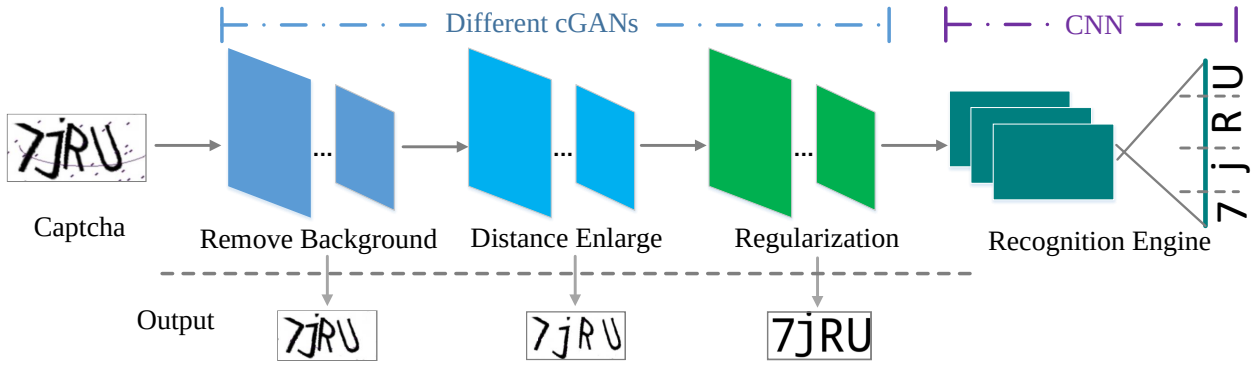


Figure 2. The overview of our attack.

5. Distortion Distorting characters using attractor fields. **6. Rotating** Rotating characters with random angles. **7. Waving** Distorting the characters in a wave fashion.

2) Previous Captcha Scheme: According to the [16], the previous text-based Captchas can be classified into three categories: character isolated Captcha (Figure 1 (a)), hollow character Captcha (Figure 1 (c)) and Connecting Characters Together (CCT) Captcha (Figure 1 (b) and (d)) based on font style and the space between adjacent characters. Obviously, most of previous text-based Captchas use a single anti-segmentation or anti-recognition features. For examples, Figure 1 (a) only uses one single anti-recognition feature (slight rotating) and Figure 1 (b) uses one anti-segmentation feature (character overlapping) and one anti-recognition feature (character waving). Of course, some Captcha schemes use a simple combination of the above security features such as Figure 1 (f) uses rotating, background and connection lines. Although using multiple anti-segmentation and anti-recognition features, the text have the same font style or obvious difference comparing to the background.

Thus, the robustness of text-based Captchas is an active topic in the research communities. Researchers have comprehensively evaluated the security of text-based Captchas [5, 8, 16]. They demonstrate that text-based Captchas are vulnerable under their attacks as they are able to effectively segment each character. Although the security of text-based Captcha scheme are suspicious by research communities, such scheme is still primary authentication mechanism and are widely deployed most mainstream websites. This wide-spread usage, we think, is due to the following reasons: (1) the written text on Captcha image may be presented in many styles such as distorted, rotated, hollow, or overlapping characters, complex noisy background or noise interference. Each of these styles can be designed more complex than previous. For examples, the value of character's rotating angle can be set more larger than previous, and the background can be became more complex. (2) The conbination of these styles is able to increase the security strength of Captchas so that it can defeat against current attacks. Current Captchas (Figure 1 (2)) combine many anti-segmentation and anti-recognition features than previous Captchas (See Figure 1 (1)). This increases the security of text-based Captchas. Recent studies suggests that text-based Captchas are still a security mechanism if they are properly designed [9, 44]. In the following, we will introduce the current Captcha scheme.

3) Current Captcha Scheme: Current Captcha scheme combines many anti-segmentation and anti-recognition features. This makes current text-based Captchas more complex than previous Captcha schemes. Figure 1 (2) shows some representative current Captcha schemes. Obviously, it presents that each current Captcha scheme is compromised of many security features such as different font style, character overlapping, rotating and distorting, connecting lines and so on. Take Figure 1 (k) for an example, this Captcha using two anti-segmentation features (complex background and connecting lines) and five anti-recognition features (Character Set: both English letters and Arabic numerals, Font Style: both solid font and hollow font, Font Size: different font size, Font Color: different font color, Rotating: rotating character with different angles.). Likewise, other Captchas presented in Figure 1 also uses many anti-segmentation and anti-recognition features.

Previous attacking methods on Captchas cannot directly crack current complex text-based Captchas. The pixel counting approach proposed by Yan *et al.* [49] will lose their efficiency in current Captchas. This is because the character sizes of current Captchas are different, which leads to the number of pixels is not fixed. Color Filling Segmentation (CFS) method proposed by Yan and used by Gao [15, 50] are also cast into the shade as sometimes the characters on current Captcha are overlapped, and CFS method cannot segment the connected character. The powerful generic attacks which claim to crack a wide range of text-based Captchas have been proposed [5, 16]. The attack described in [5] is a state-of-the-art way to segment the characters. However, this approach cannot effectively segment the characters when the space between adjacent characters is -3 pixel which is more than the space (generally about -4 pixel) between the adjacent characters of current Captcha. The work in [16] exploits Gabor filters to extract character components, and then rank the adjacent character components to recombine the individual characters. The major issue of this method is the high error ranking rate whiling cracking hollow characters. This is because the hollow characters will be extracted many possible tiny components, which results in many candidate combinations, increasing the error rate. Thus, comparing to previous Captchas, current Captcha scheme deployed by many popular web sites are more robustness, and can defeat against previous attacks.

In terms of the above complex Captcha schemes, this paper presents a hierarchical attack method using deep learning technology. Our approach is able to dynamic adapt the different

Table I. MAPPINGS FROM SECURITY FEATURES TO ITS CORRESPONDING NUMBERS

Security Features	**Anti-segmentation Features**			**Anti-recognition Features**						
	Background	Connection Lines	Overlapping	Char Set	Font Style	Font Size	Font Color	Distortion	Rotating	Waving
Feature Number	⓪	①	②	③	④	⑤	⑥	⑦	⑧	⑨
Param Value	Fixed or Random	Number, Location	Fixed or Random	Fixed or Random						

styles of Captchas. Specifically, to the Captchas with confusing background, our method can effectively clean up the complex background. To the Captchas with overlapping characters, our method can expand the space between adjacent characters. To the Captchas with both cases, our method first remove the background and then enlarge the space of adjacent characters. Our work aims to call the community to revisit the security of text-based Captchas under the age of Artificial Intelligence.

B. Generative Adversarial Networks

Generative Adversarial Networks (GANs) is a special type of Artificial Neural Network (ANN), and it was first proposed by Goodfellow in 2014 [19]. GANs consists of two neural networks competing with each other: one is the generative model G that generates the data which similar to the true data; the other is the discriminative model D which validate whether the inputting data comes from the training data or G. By repeating this competing procedure iteratively, this game will be terminated if the discriminative model D cannot identify the inputting data comes from training data or the generative model G.

The adversarial trait of GANs makes it has been widely used in video predicting [47], image processing [25, 53], natural language processing [29, 52], code security [31, 48]. Among these fine studies, Phillip *et al.* [25] proposed an image-to-image translation method named *Pix2Pix* [13] using the conditional GANs[3] (cGANs) [33]. In their work, they can translate an image from one style to another. Differ from typical GANs, cGANs consists of conditional generator and discriminator. Both the conditional generator and discriminator take the observed image as an input image so that cGANs can converge to a stable state.

This work inspired us that can it translate the hardly recognizing Captchas to easy identifying one? That is can we effectively remove the confusing background, expand the space of adjacent characters or mitigate the deformed characters? To do this, we design an effectively segment and recognize method for Captchas using a variant of *Pix2Pix* [13]. Our method is able to clean up the complex background and connecting lines on Captcha image, expand the space between the adjacent characters and translate the distorted character to the regular one. The preliminary experiments prove this method is valid. To our best knowledge, this is the first work to attack Captchas using GANs.

III. OVERVIEW

This section gives an overview of our attacking system which exploits deep learning techniques to recognize current Captchas. The system takes in a deformed text-based Captcha with complex noisy background. It automatically generated the corresponding regular Captcha. At last, the regular Captcha is cracked by CNN recognition engine. Figure 2 depicts the overview of our attack:

Step 1. Captcha Generator and Preprocessing

The first step of our attack is to generate enough training Captchas that similar to the real ones. To do so, we developed a Captcha generator that can imitate Captcha production process, and automatically generate different styles of Captchas that are similar to those deployed in real-world websites. After that, we need to uniform the font style of characters for further processing. This is because on some Captcha schemes, the characters have different font style (see Figure 4 (a), it includes both hollow and solid characters). We have shown that our approach can automatically translate the hollow character to the solid one (Figure 4).

Step 2. Generate Regular Captcha

Once the style of character on Captcha image is uniformed, a deep learning algorithm will be applied to generate the regular Captcha shown as Figure 2. We archive this through hierarchical approaches described as the following three steps:

1 **Remove Noisy Background:** To generate the regular Captcha, the first step is to clean the complex background, and produce the Captcha with white background. To do so, a deep learning algorithm will be applied to remove the complex noisy background stay on Captcha. For each Captcha scheme, this algorithm needs to train the model for cleaning out the background.

2 **Expand Space Between Adjacent Characters:** This step aims to increase the distance between two characters on Captcha. We use the same algorithm as step 2 to enlarge the inter-character distance and generate a new Captcha. Keep in mind that the Captcha generated at this stage is distorted.

3 **Regularization:** In this step, our system is able to automatically translate the distorted Captcha to a regular one.

Step 3. Recognize Captcha

In this final step, we use a radical CNN model as the recognition engine to identify the text of the regular Captcha translated from last step.

IV. IMPLEMENTATION DETAILS

A. Captcha Generator and Preprocessing

1) Captcha Generator: A key success factor for deep learning system is large amount of data to train. Intuitively, the simple yet direct approach in data collecting is to mine the Captchas from target websites and then paid to mark the

[3]Conditional GANs learn a mapping from observed image with random noise to output image while GANs learn the mapping from only random noise to output image.

Algorithm 1 Unifying the character font style

Input:
CI: Captcha Image
$numChar$: The number of characters on Captchas
Output:
UC: Captcha with the same font style

$grayCI \leftarrow getBinaryImage(CI)$
$positions[] \leftarrow getHollowPositions(grayCI)$
$LEN \leftarrow getPositonsLen(positions[], numChar)$
$meanThick \leftarrow getMeanCharThick(CI, positions[])$
for $i = 1 : LEN$ **do**
  $cFI \leftarrow FillRedColor(CI, positions[i])$
  $cCI \leftarrow ChangeFillColor(cFI, positions[i])$
  $charThick \leftarrow getCharThick(cCI, positions[i])$
  if $charThick > meanThick$ **then**
    $UC = cropChar(cCI, positions[i], meanThick)$
  end if
end for

labels to them artificially. Obviously, it is a time- or financial-consuming work with a high error rate during manually marking labels. Further, some websites limit the accessing frequency, making it hard to mine their Captchas. In addition, there is never such a public Captcha generator which can produce enough different styles of Captchas that match the requirement of the training process. Thus, the first concern of our work is to develop a flexible, well-calibrated training data generator.

Generative Model: To generate enough training samples, we design an generation model to imitate Captchas production process, and automatically generate different styles of Captchas that are similar to those deployed in real-world websites. Given a unique text-based Captcha scheme x, we first manually analyze the number of the characters and their corresponding security features, $S_{1:N}$, with N parameters such as font style, size and color, rotating, distortion and waving *et al.* described in Section II-A1. Here we ranked the security features from No.1 to 10 shown as Table I. We specifics this analysis results as $\{M, S_{1:N}\}$, where M and N respectively are the number of characters and its styles. Then given the content of Captcha, our generation model is able to automatically produce Captchas image y, which accords with its style. We define our generation model as follows:

$$y \mid x \sim G_s(x), x = \{M, N, L_{1:M}, S_{1:N}\} \quad (1)$$

Where $G_s(x)$ is the generation model parameterized by the unique style s, that can generate the target Captcha image y based on the target Captcha scheme x. M is the number of characters of x and $L_{1:M}$ represents the content of the characters. For each character, our generation model individually generates corresponding style defined by $S_{1:N}$, because each character on some Captchas may have different style.

Quantify Security Features: To generate the target Captchas as likely as possible, the security features of Captcha should be quantified. Given an unique Captcha scheme, we first should identify how many security features it used. Then for each used security feature, we need to quantify its parameters value. The last row in Table I shows the parameter value for each security

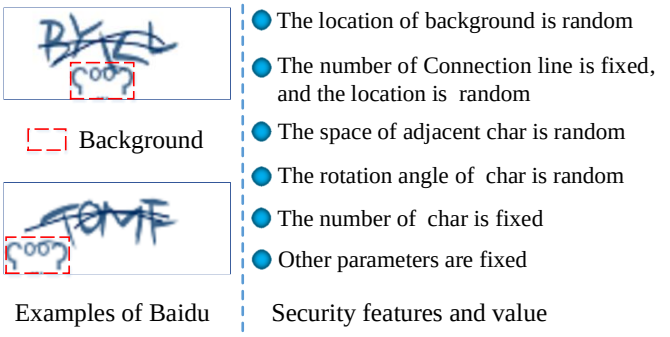


Figure 3. The figure shows the security feature of Baidu Captcha and their parameter value.

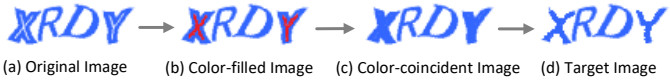


Figure 4. The procedure of Captcha preprocessing.

feature. For example, if an individual Captcha scheme used rotating and overlapping characters with background, we need to analyze whether the background is fixed or random, the rotating angle and the overlapping distance are fixed or random by observing a number of real Captchas. Figure 3 presents the security features of Baidu Captchas and the parameter value of each feature.

Example: We use the Captcha image depicted in Figure 2 as an example to describe our Captcha generator. This Captcha scheme consists of both English letters and Arabic numerals and the number of characters are fixed (4 characters). The content of the Captcha is $\{7, j, R, U\}$. It uses 2 anti-segmentation features (Complex Background and Connection Lines) and 4 anti-recognition features (Character Set, Font size, Rotating and Distortion). So the collection of security features number is {⓪, ①, ③, ④, ⑥, ⑦}. Among these security features, we observed that both the location of connection lines and the space of adjacent character are random, and other parameters value of security features are fixed. So the collection of security features can be represented as {{⓪, f}, {①, f, r}, {③, f}, {④, f}, {⑥, f}, {⑦, r}}. Considering these parameters, the variance x is $\{4, 6, \{7, j, R, U\}, \{⓪, ①, ③, ④, ⑥, ⑦\}\}$. For each character in the collection $\{4, 6, \{7, j, R, U\}\}$, our generator product the subgraph correspond to its security features. At last, the generator aggregates the subgraph to produce a Captcha image.

2) Captcha Preprocessing: Given some current text-based Captcha schemes mainly consist of both solid characters and hollow characters (see Figure 1 (j) and (k)), the first step of our attack is to unify the different font styles to a fixed style. Here we aims to fill the hollow character with solid core due to the following two reasons: (1) the hollow character can be easily transformed to the solid one but not the opposite. (2) the solid character can be extracted more stable features than hollow character at the following step according to our preliminary experiments.

To do so, we first convert the colorful image to black-and-white using the standard threshold selection method proposed

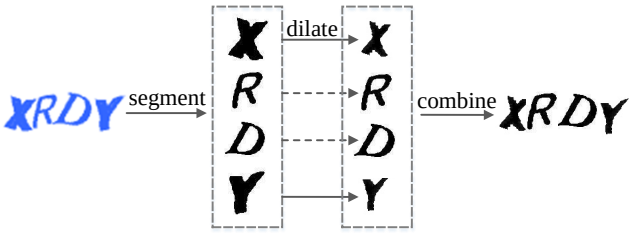


Figure 5. This figure shows how to unify the font style using the algorithm of morphological dilation.

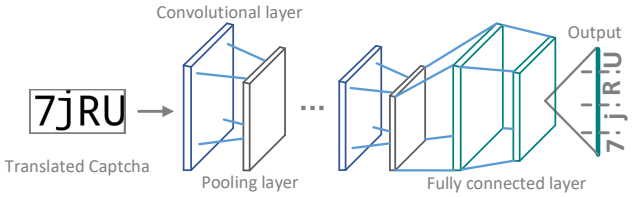


Figure 6. CNN recognition engine. The input of the recognition recognize is the regular Captcha image, and it output the text of the Captcha image.

by Otsu [38][4]. Then the Color Filling Segmentation (CFS) [50] is used to fill the hollow character with the red color (see Figure 4 (b)). Next, the color-filled image should be convert to color-coincident image by changing the red filled area to the original character color (see Figure 4 (c)). At last, the thick filled characters on the color-coincident image need to be cropped as the original character (see Figure 4 (d)).

However, how to translate the color-coincident image to the target image (shown in Figure 4) is a challenge. Fugure 5 presents our approach to address this challenge. We first segment the characters using CFS method [50]. Then we extract the color-filled characters and using morphological algorithm to dilate these characters. To effectively dilate both the edge and inflexion areas, we use the ellipse and cross kernels. We set the size of the kernel to (10, 10), an empirical setting that makes dilated characters clearly visible. After that, we combine these segmented characters to a Captcha image.

Our method for unifying the character font style is described in Algorithm 1. The input to the algorithm is the original Captcha image with different font styles and the number of characters on this Captcha, and the output of the algorithm is the Captcha with solid characters. To locate the hollow characters, we first convert the colorful original image to binarized image (line 1). According to the binarized image, we can get the positions of the each hollow characters on the colorful Captcha image (line 2) because the size of colorful image is the same as the binarized image. For each hollow character, we use CFS method described above to fill it with red color (line 6) and then replace the red filled color with the character color to get the color-coincident image (line 7). Finally, we crop the filled character on the color-coincident image to unify the font style of characters (line 10).

B. Generate Regular Captcha

After unifying the character font style, we need to generate regular Captchas for better recognizing. We achieve this by employing an image-to-image translation algorithm called Pix2Pix [13]. This algorithm automatically translate the image from the original style to the target style. In our case, the images to be translated are the Captchas with complex noisy background, overlapping and distorted characters (original style). These are supplied to the algorithm by a Captcha generator developed using a simple script (Section IV-A). The algorithm tries to generate regular Captcha with appropriate character spacing and clean background (target style).

[4]Note that we only use the binarized image to locate the position of the hollow part of the character other than furthering processing.

In order to generate regular Captcha, we propose the hierarchical methods that employ a variant approach of *Pix2Pix* to complete the progressive tasks. These hierarchical methods share the same methodology of the variant algorithm. The main differences of these hierarchical approaches are the input and output images. The inputs of the first steps are the images with complex background, overlapping and distorted characters. These inputting images are eliminated the complex background by the variant approach which outputs the Captcha images with white background, overlapping and distorted characters. According to the outputs, the second step is to expand the space between the adjacent overlapping characters using the same variant method, and produces the Captcha images only with distorted characters. Finally, the distorted characters are translated to the regular ones using the variant method.

Hierarchical Methods. The hierarchical methods are comprised of three sequenced models, and they can respectively achieve the tasks of removing complex background, expanding space between adjacent overlapping characters and translating the distorted Captcha to a regular one. The sequenced models share the same translation model. The key part of the translation model is a variant of *Pix2Pix*, which consists of a image generator and discriminator. They are competing with each other until reach Nash equilibrium when training processing (Section II-B).

In our case of removing background, the inputting data are the image pairs including the Captcha images with complex background and the images with white background. The goal is to train a generative model that can translate the Captcha image with complex background to the images with white background. During training, the image generator produces the fake image that similar to the image with white background by randomly add the noisy points to it. The fake image and the image with complex background compose an image pair. For the composed image pair, the discriminator learns to classify between the real and fake pairs. This competing process will be terminated until the discriminator cannot classify the image pairs. Unlike the *Pix2Pix*, we use the L2 distance to figure out the loss of the generator as L2 loss can capturing the overall structure of the Captcha image, which contribute to removing the background.

$$\mathcal{L}_{L2}(Gen) = \mathbb{E}_{x,y\epsilon C_O,z\epsilon N_O}\|y - Gen(x,z)\|_2 \qquad (2)$$

Where x is the Captcha image with white background, y is the image with complex background and z is the fake image with the noisy points.

C. Identify Captcha

In this step, we use a rudimentary CNN framework, *LeNet-5* [28], as our recognition engine, to identify the text of the translated Captcha image. The initial *Lenet-5* was compromised of three convolutional layers, two pooling layers and followed by two fully connected layers. The convolutional layer extracts the features using a number of filters that are trained during training process. The pooling layer aggregates the features extracted from the convolutional layer for extracting more representative features meanwhile reducing the amount of calculation. The fully connected layer classify the extracted features into target categories. The appropriate number of network layers determines the quality of the extracted features as proper number of layers will extract more representative features.

Unlike the *LeNet-5*, the goal of our recognition engine is to recognize the text of Captchas on the whole, which is more difficult than recognition a single character done by *LeNet-5*. This is because recognizing more characters need to extract more complex and abstract features. To do so, we redesign the *LeNet-5* and adding another two convolution layers and another three pooling layers. Figure 6 depicts the framework of our recognition engine. Generally, it consists of five convolution layers, five pooling layers and followed by two fully connected layers. Each convolution layer is followed by a pooling layer.

To extract more representative features, each convolutional layer uses a convolution filter of 3×3 and each pooling layer employs the max-pooling value. Other parameters are the same as the *LeNet-5*.

V. Experimental Setup

A. Data Preparation and Collection

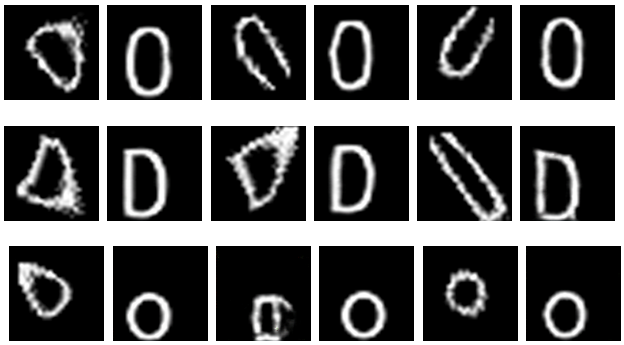
The Captchas used in our evaluation are made up of both training data and testing data, and they come from two different sources. The training Captchas are synthesized by our Captcha generator (Section IV-A) and the testing Captchas are collected from corresponding websites using a traditional mining method written through a python script.

Target Websites. We select the target websites based on the following reasons: (1) The Captchas of the websites combines at least one anti-segmentation and three anti-recognition features; (2) the websites must have a large number of active users and (3) The testing Captchas are relatively easy to mine [5]. According to the three factors, we target 5 famous websites for preparing our data: Baidu, Alipay, NetEase, VIPSHOP and TOUP. Table II presents the target websites and their corresponding Captcha schemes used in our experiments.

Synthesize Training Captchas. To ensure the synthesized Captcha as likely as the true one, we first manually analyze the traits of the true Captchas and aggregates these traits to a feature tuple as described in Section IV-A. According to the feature tuple, a large number of training Captchas can be synthesized by our generator. In our experiment, we totally synthesized 5 kinds of Captchas come from the target websites. For each website, we mined 20000 Captchas to be used for training process.

[5]Some Captcha schemes, such as Tecent Captchas, are hard to mine as each Captcha image has a different URL.

(a) MNIST dataset



(b) Single character dataset

Figure 7. This figure shows some examples of original samples and their regular one transformed by our approach.

Mine Testing Captchas. The testing Captchas used in our evalustion are automatically mined from the target websites using a python script. For each target website, we apply 2000 Captchas to test the efficiency of our attacking system. We recruited nine participators from our institution to manually mark the labels of the mined Captchas. To decrease the marking error rate, the nine participator are divided into three groups and each group has three people. We choose these Captcha which two participators mark the correct tags of the Captchas at the same time as the testing data.

Implementation. Our prototyped attacking system built upon a variant of *Pix2Pix* framework [13] in Tensorflow. The developed software ran on an ordinary server with a 3.2-GHz Inter Xeon CPU with 100-GB RAM and a TITAN Xp GPU. The operating system is Ubuntu 16.04.

B. Captchas Show

VI. Experimental Results

In this section, we first present the overall success rate for breaking the Captchas collected from the target websites. Our results shows that the overall success rates range from 45% to 90% speeding no more than 80 millseconds. We then analyze how the success rate is affected by the number of the training Captchas. Finally, we demonstrate the risks of our attacking approach by simulating a real world attacking process before evaluating the robustness of the secure features using variant types of text-based Captchas.

Table II. TARGET TEXT-BASED CAPTCHA SCHEMES USED IN OUR EXPERIMENT

Scheme	Website	Sample	Security Features	
			Anti-segmentation Features	Anti-recognition Features
Baidu	baidu.com		Connecting Lines, overlapping, only Enligh letters used	Both hollow and solid characters, varied font size, color, rotating, disortion and waving used
Alipay	alipay.com		Overlapping characters used	Both English letters and Arabic numerals, rotating and distortion used
NetEase	163.com		Complex background, connecting lines	Both hallow and solid characters, varied rotating angles used
VIPSHOP	vip.com		Complex background, overlapping characters	Both English letters and Arabic numerals, varied font colors used
TOUP	tuniu.com		Overlapping characters used	Both English letters and Arabic numerals, waving characters used

Table III. THE OVERALL SUCCESS RATE AND AVERAGE ATTACK SPEEDS FOR EACH CAPTCHA SCHEME.

Scheme	Success rate	Speed (ms)
Baidu	45%	70
Alipay	86%	72
NetEase	67%	74
VIPSHOP	90%	65
TOUP	69%	50

A. Overall Success Rate

In this experiment, the training data were synthesized used our Captcha generator, and the number of each training Captcha scheme is 20000. The corresponding testing data were collected from the target websites and its number is 2000.

1) Evaluation using current text-based Captchas: Table IV shows the success rate for breaking different text-based Captcha schemes and it ranges from 45% to 90%. For VIOSHOP scheme, we are able to successfully cracking 90% Captchas as our attacking approach can effectively remove the complex background as well as expand the space of adjacent characters. For Baidu Captchas, the success rate decreases to 45%. This is because Baidu uses almost two different styles of Captcha scheme shown in Table II. Further, it exploits both hollow and solid fonts. These designs make such scheme more complex than other Captchas and more difficult to be recognized. On average, the breaking speeds for each Captcha image is no more than 80 milliseconds. In general, our cracking success rate is far more than 1%, the secure threshold at which the Captcha is considered ineffective [8], which means our approach pose a great threat to current widely used text-based Captchas.

2) Evaluating using Previous text-based Captchas: In this section, we compare our result with previous works [5, 8, 16] using the previous text-based Captchas. For the previous Captchas, the real world testing data are hard to be collected because most websites have updated their Captcha scheme which is more complex than prior one. To acquire the true previous Captchas, we first use some online Captcha generator such as Captcha.net to produce the true Captcha images. We then employ the generators that previous papers has provided to generate Captchas. If neither of the two cases, we use our Captcha generator to produce the testing data.

Figure 8 depicts some Captcha examples of the previous real world Captchas and our generated versions that produced by our generator (Section IV-A). Recall that our Captcha generator is able to generate the similar style Captchas with the real world Captcha scheme. So the generated Captcha has some subtle differences from the real one. For example, Figure 8 (e) shows the real Captcha deployed by Microsoft, and it slightly different from the generated one shown in (f) because the each character on Captcha image is randomly rotated and distorted. Nevertheless, the generated Captchas contain the main features of the original real world Captchas.

The experimental results show that

3) Evaluation using MNIST datasets: We evaluate our approach using the MNIST datasets, and compared the results with the latest work [18] which can recognize the MNIST with a high accuracy using a small number of training data. In this experiment, we selected 100, 200 and 500 MNIST examples for each digit for training, respectively. Table V demonstrate that our approach is superior to RCN when the total number of training examples more than 5000. Indeed, RCN performs better than ours when using a small number of training data as our approach cannot correctly transformed the similar characters to the regular one.

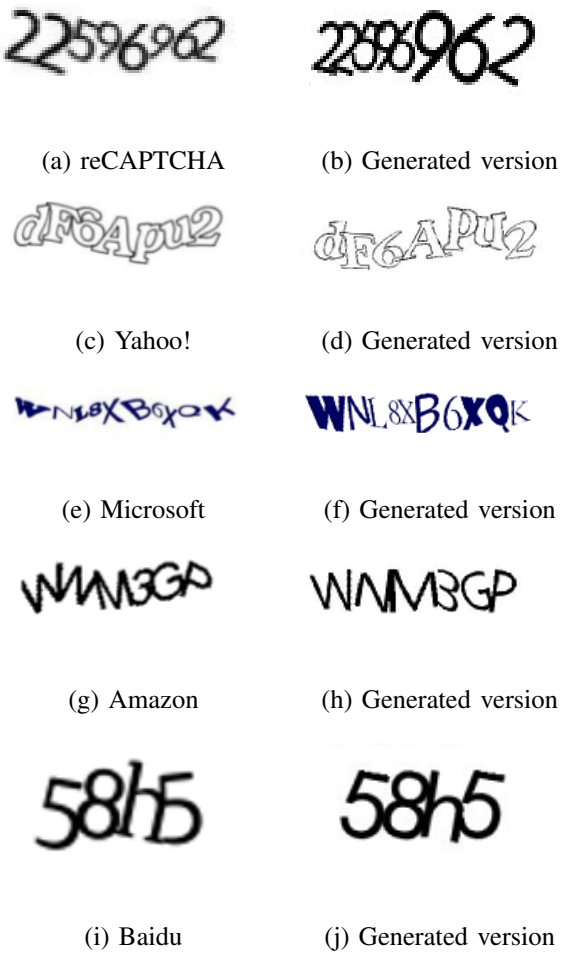
In addition, we generated single character dataset which contains digits and English characters. Each character in this dataset is rotated and distorted. The stye of the generated dataset is a little like MNIST's style. We use such dataset to evaluate both RCN and our approach, and the results are shown in Table V. It obviously show that our approach is able to recognize most characters while RCN is invalid. Figure 7 shows some examples of original data and the corresponding regular version.

B. Impact of the number of training data

We would like to know how the number of training examples affects the success rate of the attack. To do so, we used all the collected Captchas and we varied the number

Table IV. THE OVERALL SUCCESS RATE AND AVERAGE ATTACK SPEEDS FOR EACH CAPTCHA SCHEME.

Scheme	Success rate		Scheme	Success rate		Scheme	Success rate		Scheme	Success rate	
	ref. [8]	ours		ref. [16]	ours		ref. [5]	ours		ref. [18]	ours
Authorize	66%		reCaptcha	77.2%	96.62%	Baidu(2011)	38.68%		reCaptcha	66.6%	
Blizzaer	70%		Yahoo!	5.2%		Baidu(2013)	55.22%		Yahoo!	57.4%	
Captcha.net	73%		Baidu	46.6%	95.62%	CNN	51.09%		PayPal	57.1%	
Digg	20%		Wikipedia	23.8%	76.38%	eBay	51.39%	99.46%	MNIST	97.89%	97.45%
eBay	43%	98.9%	QQ	56%		ReCaptcha(2011)	22.67%				
Megaupload	93%		Microsoft	16.2%		ReCaptcha(2013)	22.34%				
NIH	72%		Amazon	25.8%	78.38%	Wikipedia	28.29%	76.38%			
Reddit	42%		Taobao	23.4%	96.44%	Yahoo!	5.33%				
Slashdot	35%		Sina	9.4%							
Wikipedia	25%	76.38%	eBay	58.8%	99.46%						



(a) reCAPTCHA (b) Generated version

(c) Yahoo! (d) Generated version

(e) Microsoft (f) Generated version

(g) Amazon (h) Generated version

(i) Baidu (j) Generated version

Figure 8. Some examples of the real world Captchas and the corresponding generated version. The Captchas on the left column are the previous real world schemes and on the right column are the corresponding generated version.

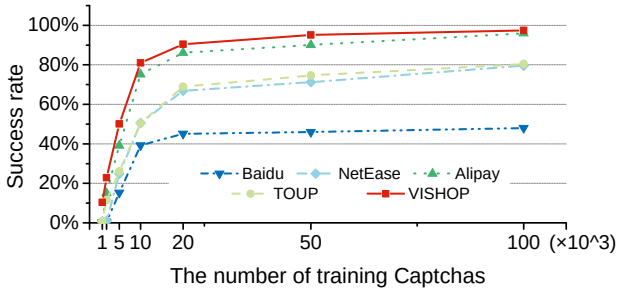


Figure 9. Impact of the number of training set.

Table V. CLASSIFICATION RESULTS ON BOTH STANDARD MNIST AND GENERATED DATASET.

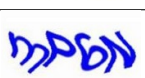
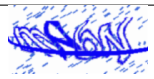
MNIST dataset			
# of per class	100	200	500
RCN	97.89%	98.10%	98.60%
Ours	**91.40%**	**94.50%**	**98.75%**
Single character dataset			
# of per class	100	200	500
RCN	82.67%	81.13%	xx.xx%
Ours	**87.68%**	**89.84%**	**92.29%**

of training Captchas from 1000 to 100000 for each scheme. Figure 9 shows how the cracking success rate changes as the number of training Captchas increases. When the size of the training examples is 100000, our approach can break 97.5% VISHOP Captchas, 96.6% Alipay Captchas, 80.4% TOUP Captchas, 79.6% BetEase Captchas and 48% Baidu Captchas respectively. The success rate drops significantly when the number of training Captchas is less than 10000. Beyond this point, our approach cannot extract representative features of the Captchas. The degradation features makes it difficult for our algorithm to successful recognize the content of the Captcha image. Due to the degradation, our approach can just crack 2.65% Baidu Captchas, 3.45% NetEase Captchas, 4.25% Alipay Captchas, 7.55% TOUP Captchas and 10.4% VISHOP Captchas respectively when the number of training examples is only 1000. Nonetheless, the success rates are higher than 1%, the security threshold that determines whether or not the Captcha is secure. Generally, our approach can achieve a high success rate when the number of training examples is 20000.

C. Impact of Security Features

Intuitively, the success rate of the attack would be influenced by the number of security features used by the Captcha because the complexity of the Captcha depends on the number of the security features. To evaluate this effect, we choose the commonly used security feature such as overlapping, rotating, distortion, waving and background for the experiment. First, we evaluate how a single commonly used security feature effect the success rate. We then evaluate our attack using our synthetic Captchas which combines of a number of commonly used security features. Note that the synthetic Captchas are more complex than current any real world Captcha scheme.

Table VI. THE SUCCESS RATE OF EACH COMBINATIONS SECURITY FEATURES

No.	Sample Captcha	Overlapping	Rotating	Distorting	Waving	Success Rate
1		√	√			74.85%
2		√		√		65.05%
3		√			√	58.8%
4			√	√		64.95%
5			√		√	82.35%
6				√	√	62.45%
7		√	√	√		57.50%
8		√	√	√	√	52.50%
9		All security features				46.30%

At last, we generate a most complicated Captcha using all security features to evaluate our approach.

1) Evaluation of single security feature

Overlapping. Overlapping decreases the space between adjacent characters and makes them collapsed, and it is considered the most effective anti-segmentation security feature [8] which is widely deployed by lots of popular websites.

We use the TOUP scheme as a case study to evaluate the security of overlapping. We modify the original TOUP Captchas through increasing character overlapping by 4, 6, 8 and 10 pixels[6], respectively. For each overlapped Captchas, we generated 20000 training examples and 2000 testing examples. We then run our attack on these synthetic overlapped Captchas, and our new success rate are 65%, 50.05%, 42.55% and 25.1%, respectively. This result demonstrates the more overlapped the characters, the less successful our approach would be. Although the success rate drops significantly when the character overlapping more than 6 pixels, it still pose a real threat to Captcha and also superior to human because human cannot recognize the Captcha when the character overlapping reaches 6 pixels [5].

Rotating. To evaluate how the character rotating effects the success rate, we choose the Alipay scheme for implementing the experiment as Alipay Captcha randomly rotates the character. This is more difficult than other scheme which rotates the character in one single direction. The rotating angle of Alipay is less than 15 degrees through our careful analysis. In this experiment, we increases the random rotating angle to 30 degrees. We use 20000 such Captchas for training and 2000 for testing, and our approach can achieve a success rate of 80.5%, a little lower than the original success rate of 86.15%. This indicates rotating does not have a positive effect in enhancing the security of Captcha under our attack.

Distorting. Character distortion is widely used by most current text-based Captcha schemes. It can confuse the bot program for recognizing the similar distorted characters. For example, the character "O" and the number "0" looks very similar when they are distorted. This makes the malicious bots misidentifying "O" as "0" or "0" as "O".

In this experiment, we generate 20000 training Captchas which are similar to the Alipay scheme. But the generated Captchas are more distorted than Alipay scheme. Using 2000 Captchas for testing, our approach is able to crack 67.5% which is lower than the original success rate of 90.10%. This indicates that although character distortion enhancing the security of Captcha to some extent, our approach can still break such Captcha with a high success rate.

Waving. In this evaluation, we use the Captcha generator proposed by Tuan *et al.* [27] to produce 20000 training examples for training. Using the trained model, our approach is able to recognize 1977 out of 2000 testing Captchas, and have a success rate of 98.85%, higher than Tuan's success rate of 93.6%. We only failed to break 23 Captchas because our model is confused by some similar waving characters such as character "O" and number "0", character "1" and "l", *etc*. This experiment demonstrates that waving transformation of characters does not contribute to enhancing the security of Captcha under our attack.

2) Evaluation of combining security features

[6]The overlapping of the original Captcha is no more than 3 pixels.

Combination security features can offer a more secure protection than a single one. In this section, we perform a number of experiments to test various combinations of the security features, and evaluate how each combination effects our attack. We conduct this experiment to search whether or not having a combination can defeat against our attack. We choose four basic security features from which we design eight different combinations and we test all of them. We select Alipay Captcha for our experiments, and the size of training and test set is 20000 and 2000, respectively.

Table VI summaries the experiment results, listing a number of combinations of security features along with its influence on the success rate of our attack. The results suggest the following insights. Firstly, the combinations of two or more security features are indeed more secure than just a single one. Secondly, character overlapping or distorting can offer a stronger protection than rotating or waving because overlapping or distorting changes the style of character which indeed enhances security. This can be proved by the results of experiments No. 1 - 6. Lastly, the combination of all security features has a positive effect on defeat against our attack. However, our approach is able to crack 46.30% such Captchas, much higher than 1% at which Captchas is considered ineffective. On the other hand, it will decreases the usability when blindly using more security features. Nonetheless, our attack still can crack all the complex schemes with a high success rate. Therefore, the text-base Captcha is no longer a secure scheme to prevent the malicious bot programs.

VII. Related Work

Our work lies at the intersection between adversarial machine learning and cracking text-based Captchas methods. This works brings together techniques developed in the domain of adversarial deep learning and character recognition to develop the attack on current text-based Captchas.

Optical Character Recognition technology Naor proposed the definition of Automated Turing Tests for automatically distinguishing humans from computer systems [37]. This definition was firstly implemented by Lillibridge *et al.* [30] who developed the first practical scheme to prevent malicious bot programs from registering web accounts. However, such scheme was soon defeated by common used Optical Character Recognition (OCR) technique. Since then, a various of text-based and corresponding attacks have been proposed.

Attacks on the individual Captchas Mori *et al.* [22] proposed an attack on Gimpy and EZ-Gimpy using complex object recognition algorithm. Their experiments shows that their approach can break 33% Gimpy and 92% EZ-Gimpy. Yan *et al.* [49] used a simple character segmentation method to break most Captchas provided by Captchaservices.org. They segmenting the character by counting the number of pixels of each individual character. Then they improved their character segmentation technique for attacking Microsoft-like Captcha schemes deployed by Yahoo!, Microsoft and Google [50]. Gao *et al.* [15] firstly proposed a generic method to break the hollow Captcha scheme. They first extracted the hollow character strokes based on color filling segmentation (CFS) method, and then searched for possible combinations of adjacent character strokes to form the individual characters. Recently they break Microsofts two-layer Captchas using similar approach with a high success rate [14]. The above solutions first need to segment characters and then recognition them. But for the skewed and overlapped letters, their methods are hard to successfully segment as the characters are gathered together. To such scheme, Stark *et al.* [42] presented an approach that can recognize the Captchas based on Active Deep Learning. Their approach can break Google's reCAPTCHA with a high success rate using a small number of labeled data.

Generic attacks on text-based Captchas Bursztein *et al.* [5] reported a generic method to break a wide of text-based Captchas in a single step using machine learning algorithm. Their approach can recognize the Captcha by addressing the segmentation and recognition problems simultaneously. It scores all possible ways of character combinations to segment a captcha using reinforcement learning algorithm and search for the best possible one as the result. More recently, Gao *et al.* [16] present another generic approach that solved text-based Captchas. They first used Gabor filter to extract character components of the Captcha in four different directions, and then the extracted character components is ranked ordered by a graph search algorithm before recognizing the ranked components using Convolutional Neural Network. The success of the two above approaches lies in the relative clean background. That is their approaches perfect poor when the Captcha has complex background as VISHOP scheme shown as Figure 1. This is because their segmentation method cannot distinguish the individual character from such background. George *et al.* [18] proposed a novel network model named RCN that can break text-based Captchs using a samll number of training data.

Study on the robustness of text-based Captchas Bursztein [8] conducted a systematic study on existing text-based Captchas based on distorted characters. Among studying 15 Captcha deployed on popular websites, they found that 13 schemes excepting for Google and reCAPTCHA are vulnerable to automated attacks. They also explored whether it is easy to recognize the Captchas for human by conducting a large scale evaluation of Captchas from human's perspective [7]. They collected 21 most widely used captcha schemes and asked the participants to manually recognize these Captchas. Their experiment results indicated that Captchas general were difficult for humans, especially for the audio Captcha schemes. Krol *et al.* [26] carried out a user study on both reCAPTCHAs and "gamified" Captchas. They found that participants preferred to reCAPTCHAs than gamified Captchas because they were familiar with reCAPTCHAs.

Attacks on other Captcha schemes In addition to text-based Captchas, there are other Captchas such as image-based Captchas [1, 2, 12, 21, 36], audio-based Captchas [4, 40]. However, researchers have proposed some attacks on image-based Captchas [17, 35, 41] and audio-based Captchas [6, 32, 43], respectively. Further, such Captcha schemes are vulnerable to side-channel attacks [23] as the limited number of example Captchas can be easily mined by the adversary.

Adversarial machine learning Recently, adversarial machine learning [24] begins to be used in the field of information security. Researchers have found such technique can be used to evading classifiers of malicious software [39, 48], constructing specific malicious examples bypass intrusion detection system or spam e-mail filtering [3] and cheat any classifiers trained by machine learning algorithms [20, 34].

VIII. Conclusion

References

[1] "Are you a human," https://www.areyouahuman.com/.

[2] E. Athanasopoulos and S. Antonatos, "Enhanced captchas: using animation to tell humans and computers apart," in *IFIP International Conference on Communications and Multimedia Security*, 2006, pp. 97–108.

[3] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *ACM Symposium on Information, Computer and Communications Security*, 2006, pp. 16–25.

[4] J. P. Bigham and A. C. Cavender, "Evaluating existing audio captchas and an interface optimized for non-visual use," in *Sigchi Conference on Human Factors in Computing Systems*, 2009, pp. 1829–1838.

[5] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: generic solving of text-based captchas," in *USENIX conference on Offensive Technologies*, 2014.

[6] E. Bursztein and S. Bethard, "Decaptcha: breaking 75captchas," in *Usenix Conference on Offensive Technologies*, 2009, pp. 8–8.

[7] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and J. Dan, "How good are humans at solving captchas? a large scale evaluation," in *IEEE Symposium on Security and Privacy*, 2010, pp. 399–413.

[8] E. Bursztein, M. Martin, and J. Mitchell, "Text-based captcha strengths and weaknesses," in *ACM Conference on Computer and Communications Security*, 2011, pp. 125–138.

[9] E. Bursztein, A. Moscicki, C. Fabry, S. Bethard, J. C. Mitchell, and J. Dan, "Easy does it: more usable captchas," in *ACM Conference on Human Factors in Computing Systems*, 2014, pp. 2637–2646.

[10] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Building segmentation based human-friendly human interaction proofs (hips)," vol. 3517, pp. 1–26, 2005.

[11] ——, "Computers beat humans at single character recognition in reading based human interaction proofs (hips)," in *Conference on Email & Anti-Spam*, 2005.

[12] J. Elson, J. R. Douceur, J. Howell, and J. Saul, "Asirra:a captcha that exploits interest-aligned manual image categorization," in *ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, Usa, October*, 2007, pp. 366–374.

[13] P. I. *et al.*, "Pix2Pix: Image-to-image translation with conditional adversarial networks," https://github.com/phillipi/pix2pix.

[14] H. Gao, M. Tang, Y. Liu, P. Zhang, and X. Liu, "Research on the security of microsofts two-layer captcha," *IEEE Transactions on Information Forensics & Security*, vol. 12, no. 7, pp. 1671–1685, 2017.

[15] H. Gao, W. Wei, X. Wang, X. Liu, and J. Yan, "The robustness of hollow captchas," in *ACM Sigsac Conference on Computer & Communications Security*, 2013, pp. 1075–1086.

[16] H. Gao, J. Yan, F. Cao, Z. Zhang, L. Lei, M. Tang, P. Zhang, X. Zhou, X. Wang, and J. Li, "A simple generic attack on text captchas," in *Network and Distributed System Security Symposium*, 2016.

[17] S. Gao, "An evolutionary study of dynamic cognitive game captchas: Automated attacks and defenses," *Dissertations & Theses - Gradworks*, 2014.

[18] D. George, W. Lehrach, K. Kansky, M. Lzaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, and H. Wang, "A generative vision model that trains with high data efficiency and breaks text-based captchas," *Science*, p. eaag2612, 2017.

[19] I. J. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.

[20] I. J. Goodfellow, J. Shlens, C. Szegedy, I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICML*, 2015, pp. 1–10.

[21] R. Gossweiler, M. Kamvar, and S. Baluja, "What's up captcha?:a captcha based on image orientation," in *International Conference on World Wide Web, WWW 2009, Madrid, Spain, April*, 2009, pp. 841–850.

[22] M. Greg and J. Malik, "Recognizing objects in adversarial cultter: Breaking a visual captcha," in *IEEE Computer Society Conferene on Computer Vision and Pattern Recognition*, 2003.

[23] C. J. Hernandezcastro, A. Ribagorda, and Y. Saez, "Side-channel attack on labeling captchas," *Computer Science*, 2009.

[24] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," *IEEE Internet Computing*, vol. 15, no. 5, pp. 4–6, 2011.

[25] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.

[26] K. Krol, S. Parkin, and M. A. Sasse, "Better the devil you know: A user study of two captchas and a possible replacement technology," in *NDSS Workshop on Usable Security*, 2016.

[27] T. A. Le, A. G. Baydin, R. Zinkov, and F. Wood, "Using synthetic data to train neural networks is model-based reasoning," in *International Joint Conference on Neural Networks*, 2017, pp. 3514–3521.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," 2017.

[30] M. D. Lillibridge, M. Abadi, K. Bharat, and A. Z. Broder, "Method for selectively restricting access to computer systems," 2001.

[31] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," 2016.

[32] H. Meutzner and D. Kolossa, "Reducing the cost of breaking audio captchas by active and semi-supervised learning," in *International Conference on Machine Learning and Applications*, 2014, pp. 67–73.

[33] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *Computer Science*, pp. 2672–2680, 2014.

[34] T. Miyato, S. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing by virtual adversarial examples," *arXiv*, 2015.

[35] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao,

N. Saxena, C. Zhang, P. Kumaraguru, P. C. V. Oorschot, and W. B. Chen, "A three-way investigation of a game-captcha:automated attacks, relay attacks and usability," in *ACM Symposium on Information, Computer and Communications Security*, 2014, pp. 195–206.

[36] M. Mohameda, S. Gaob, N. Sachdevac, N. Saxena, C. Zhangd, P. Kumaraguruc, and P. C. V. Oorschote, "On the security and usability of dynamic cognitive game captchas," *Journal of Computer Security*, pp. 1–26, 2017.

[37] M. Naor, "Verification of a human in the loop or identification via the turing test," 1996.

[38] N. Ostu, O. Nobuyuki, and N. Otsu, "A threshold selection method from gray- level histogram ieee transactions on systems," *IEEE Transactions on Systems Man & Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[39] I. Rosenberg, A. Shabtai, L. Rokach, and Y. Elovici, "Generic black-box end-to-end attack against rnns and other api calls based malware classifiers," 2017.

[40] A. Schlaikjer, "A dual-use speech captcha: Aiding visually impaired web users while providing transcriptions of audio streams," *LTI*, 2010.

[41] S. Sivakorn, I. Polakis, and A. D. Keromytis, "I am robot: (deep) learning to break semantic image captchas," in *IEEE European Symposium on Security and Privacy*, 2016, pp. 388–403.

[42] F. Stark, C. Hazirbas, R. Triebel, and D. Cremers, "Captcha recognition with active deep learning," in *German Conference on Pattern Recognition Workshop*, 2015.

[43] J. Tam, J. Simsa, S. Hyde, and L. V. Ahn, "Breaking audio captchas," in *Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December*, 2008, pp. 1625–1632.

[44] K. Thomas, D. Mccoy, A. Kolcz, A. Kolcz, and V. Paxson, "Trafficking fraudulent accounts: the role of the underground market in twitter spam and abuse," in *Usenix Conference on Security*, 2013, pp. 195–210.

[45] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, *CAPTCHA: Using Hard AI Problems for Security.* Springer Berlin Heidelberg, 2003.

[46] L. Von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Communications of the Acm*, vol. 47, no. 2, pp. 56–60, 2004.

[47] J. Walker, K. Marino, A. Gupta, and M. Hebert, "The pose knows: Video forecasting by generating pose futures," 2017.

[48] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on pdf malware classifiers," in *Network and Distributed System Security Symposium*, 2016.

[49] J. Yan and A. S. E. Ahmad, "Breaking visual captchas with naive pattern recognition algorithms," in *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, 2007, pp. 279–291.

[50] ——, "A low-cost attack on a microsoft captcha," in *ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, Usa, October*, 2008, pp. 543–554.

[51] ——, "Usability of captchas or usability issues in captcha design," in *Symposium on Usable Privacy and Security, SOUPS 2008, Pittsburgh, Pennsylvania, Usa, July*, 2008, pp. 44–52.

[52] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," 2016.

[53] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.