IMPROVED SECURITY ASPECTS ON MICROSOFTS TWO -LAYER CAPTCHA

Rachana.B.S, Dhruthi.S, Swarna.R, Chandan.A

Rachana.B.S, Asst.Prof, ISE, APSCE, B'lore, Karnataka, INDIA
Dhruthi S, Student, ISE,, APSCE, Karnataka, India
Swarna R, Student, ISE, APSCE, Karnataka, India
Chandana A, Student, ISE, APSCE, Karnataka, India

ABSTRACT

Captcha is a security mechanism designed to differentiate between computers and humans, and is used to defend against malicious bot programs. Text-based Captchas are the most widely deployed differentiation mechanism, and almost all text-based Captchas are single-layered. Numerous successful attacks on the single-layer text-based Captchas deployed by Google, Yahoo! and Amazon have been reported. In 2015, Microsoft deployed a new two-layer Captcha scheme. This appears to be the first application of two-layer Captchas. In this paper, we examine the security of the two-layer Captcha. We attack this scheme by a series of processing.

Keywords: *Captcha, Preprocessing, Partition and Recognition, Dynamic Programming, Improving Microsoft Security.*

---

1. INTRODUCTION

CAPTCHA (Completely Automated Public Turing Test toTell Computers and humans Apart) is used to prevent automated registration, spam or malicious bot programs [1], [2]. It automatically generates and evaluates a test, difficult for computers to solve, but easy for humans. If the success rate of solving a Captcha for humans reaches 90% or higher, and computer programs only achieve a success rate of less than 1%, the Captcha can be considered secure [3]. Current Captchas can be divided into three categories: text based, image-based and audio-based.

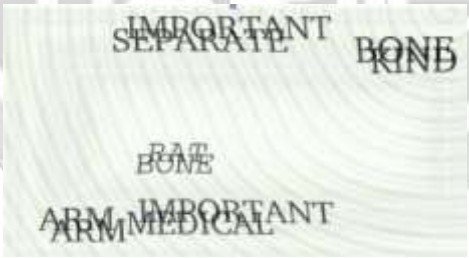

Fig-1: Captcha

Text-based Captcha is usually based on English letters and Arabic numerals, and uses sophisticated distortion, rotation or noise interference to prevent the recognition of a machine. Text-based Captcha is the most widely used scheme [4]. This wide-spread usage is due to its obvious advantages [5], [6]: users' task is a text recognition problem which is intuitive to users worldwide;

There are many examples for the failure of text based captcha like; Researchers have recently claimed that their simple generic attacks have broken a wide range of text-based Captchas in a single step .

Microsoft deployed a two-layer Captcha in 2015. It connected the sides of each character, across both top-to-bottom and right-to-left, in order to increase the difficulty of detecting where each character is. This was the first

application of two-layer Captchas, and it appeared more secure than previous schemes. In this paper, we examine the security of the two-layer Captcha. We attack this scheme by a series of processing. This appears to be the first systematic analysis of two layer Captchas. Our two-dimensional segmentation technique is innovative; it can be applied as a basis for other successful attacks.

Captcha generation process to build training sets, and using it to train the recognition engine to recognize the objects in the real world, is the first attempt in the field of analyzing Captchas' robustness. This approach can be adapted for other projects requiring a large amount of data to train machine learning systems, and for people working in Captcha design, our work provides guidelines for designing Captchas with better security and usability.

2. RELATED WORK

The Automated Turing Tests were first proposed by Moni Naor [15], but they did not provide a formal definition. Alta Vista [16] developed the first practical Automated Turing Test to prevent bots from automatically registering web pages.
This system was effective for a while and then was defeated by common Optical Character Recognition (OCR) technology. Text-based Captcha based on English letters and Arabic numerals, is still the most widely deployed scheme. Many research communities focus on developing attack approaches. For existing text-based Captchas, and then explore guidelines for better designs.

In 2006, Yan and El Ahmad [7] broke most visual schemes provided at Captchaservice.org by simply counting the number of pixels of each segmented character, even though these schemes are resistant to the best OCR software on the market.

In 2008, the same authors [8] developed new character segmentation techniques for attacking a number of text-based Captchas, which included the earlier mechanisms designed and deployed by Microsoft, Yahoo! and Google.

In 2013, Gao et al. 6 proposed a generic method to break a family of hollow Captchas, and this is the first application of solving Captchas in a single step. They used the characteristics of hollow fonts to extract character strokes, and then tested different combinations of adjacent character strokes to form individual characters.

Fig-2: various captcha (a) Character isolated captcha (b)Hollow character Captcha (c)CCT captcha (d)Captcha with noise arcs (e) Captcha with complex background

The original Captcha used by Microsoft was the character isolated Captcha (See Figure 2(a)). This character isolated scheme had been used for many years before the two-layer Captcha was deployed. When it was broken by Jeff Yan [8], the character-isolated scheme seemed to have become the most Vulnerable and other similar schemes were also shown to be insecure.

3. EXISTING SYSTEM

Since the early days of the Internet, users have wanted to make text illegible to computers. The first such people could be hackers, posting about sensitive topics to online forums they thought were being automatically

monitored for keywords. To circumvent such filters, they would replace a word with look-alike characters. HELLO could become |-|3|_|_ () or)-(3££0, as well as numerous other variants, such that a filter could not possibly detect all of them. This later became known as lee speaks. [5].

One of the earliest commercial uses of CAPTCHAs was in the Gausebeck-Levchin test. In 2001, PayPal used such tests as part of a fraud prevention strategy in which they ask humans to "retype distorted text that programs have difficulty recognizing."[6] PayPal cofounder and CTO Max Leaching helped commercialize this early use.

The first team consists of Mark D. Lilli bridge, Martín Abdi, Krishna Bharat, and Andrei Broder, who used CAPTCHAs in 1997 at AltaVista to prevent bots from adding URLs to their web search engine. Looking for a way to make their images resistant to OCR attack, the team looked at the manual of their Brother scanner, which had recommendations for improving OCR's results (similar typefaces, plain backgrounds, etc.). The team created puzzles by attempting to simulate what the manual claimed would cause bad OCR

4. PROPOSED SYSTEM

In 2015, Microsoft deployed a two-layer Captcha on its account creation page which can be seen as a vertical combination of two traditional single-layer Captchas. It requires users to recognize the upper-layer characters and then the lower-layer characters to pass the test. The upper layer is the first horizontal line of the Captcha containing three characters, and the lower layer is the second line of the Captcha with another three characters. The characteristics of the two-layer Captcha are summarized as follows:

- An image with six characters in total, three in each layer.
- It uses both hollow and solid characters randomly. Unlike
- the common solid characters, the hollow characters are
- those characters formed by contour lines, e.g. characters
- 'X', 'P' and 'R' in Figure 2(b).
- Each layer utilizes CCT scheme and the whole Captcha
- Is warped and rotated.

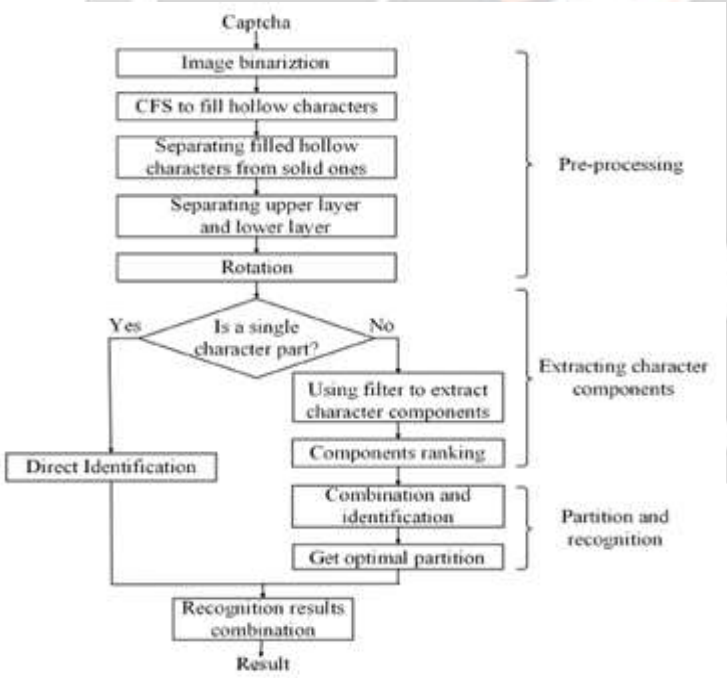

Fig 3: Flowchart

A. Pre-processing

Image binarization. First, we convert a rich-color challenge to black-and-white using the standard Otsu's threshold method 7, as Figure 4 shows. CFS [8] to fill hollow characters. This is a flooding algorithm which is used to detect connected non-black pixel blocks. Both hollow characters and noise components are picked out to

fill. Different colors are used to distinguish those components. After CFS, the background color is set to lightgray, and different components are filled with distinguishing Colors .

Unfortunately, both the strokes of hollow characters and closed regions are filled by CFS. For our purpose, some measures should be taken to remove those noise components in this step.
.
There are three types of undesirable components which are considered noise:
1) The closed part within a hollow character that does not belong to character strokes,
2) the closed part within a solid character,
3) The closed part produced by connected characters.

To remove these noise components, we define parameters and for each color component: =*C*/*S* and =*W*/*Sr*, where *C*, denotes the number of edge pixels that have a black neighbor in this color component, *S* denotes the total number of pixels in the color component, *W* denotes the number of pixels within the bounding box of the color component (an external rectangle of the color components area) but belong to the lightgray background or other color components, and *Sr* denotes the total number of pixels the bounding box of the color component contains, as Figure 4(a) and (b) show.

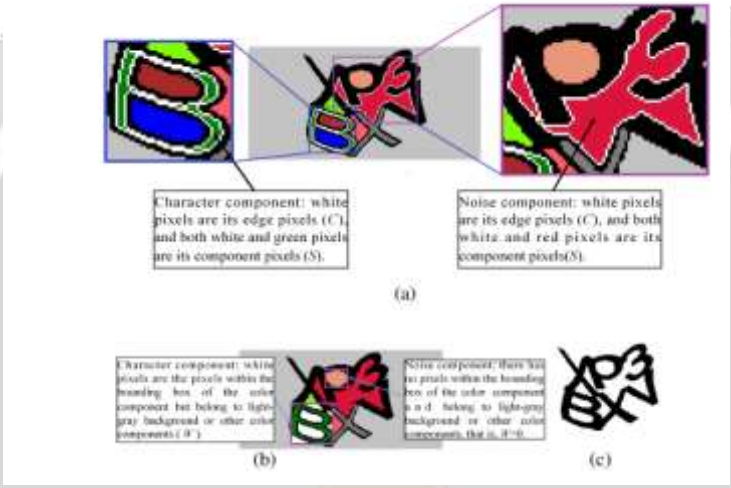

Fig-4: Pattern in pixel recognition

Letters and Arabic numerals, their color components always have a relatively larger *C* and smaller Sinise components, however, are usually simple with a plump shape. Besides, the surroundings of character components always have a large section of pixels belonging to a light-gray background or other components, whereas the remnant noise components are surrounded by character strokes and with little (or no) such pixels (see Figure 4(a) and (b)). Therefore, with properly chosen threshold values *a* and *b*, we have the following: if _ < *a* or _ < *b* for a component, we consider it as noise and remove it. The values *a* and *b* are determined by analyzing a small sample set of data.

After deleting all of the noise components, the color of the character components is changed to black, and Figure 4(c) shows the final result of CFS. The two major contributions of CFS are: first, it reduces the burden on the recognition engine since the diversity of characters significantly decreases the effectiveness of the recognition engine; second, it also avoids failed cases that possibly occur in later extraction of character components, where we adopted the state-of-the-art method in 9 to extract components from connected characters based on Gabor filters.

Noted that they received the lowest success rate on a hollow scheme deployed by Yahoo!, This extraction method breaks a long hollow text string into a large number of tiny components, and further produces a huge set of possible combinations during partition and recognition, both decreasing the success rate and slowing down the attack speed.

Learning from this prior experience, conversion of hollow characters into solid characters is done in advance. Separating filled hollow characters from solid characters.

This step is considered to achieve the highest number of complete individual characters. It makes use of the characteristics of hollow characters formed by thin contour lines that can be easily dislodged. First, we erode the binaries image (Figure 4(c)) to remove the hollow characters part. The solid characters part is also damaged during this step, but we simply recover it by dilation. Finally, by removing the solid characters part from the image received by CFS (Figure 4(c)). Process of separating the solid characters and filled hollow characters.

It may seem that this step can be skipped and that we could consider these filled hollow characters as solid ones equally. However, it is necessary in our attack. This process, as the first step of our two-dimensional segmentation algorithm, helps to receive as many individual characters as possible. [6].Suggested that the challenge of breaking a Captcha depends on the difficulty of finding where each character is. Thus we add this step to our attack in order to increase segmentation and accuracy.

Separating upper and lower layers. The two-layer structure is used as a defense measure to disguise where a single character is, and it is necessary to separate a two-layer challenge image into two one-layer images. We have found that the borderline of upper layer and lower layer ranges across challenges, but it waves along the variation trend of the envelope lines of a challenge. Based on this characteristic of Microsoft's two-layer Captcha, we start by finding the top and bottom envelope lines for each challenge image, and then generate the middle line of these two envelope lines as the borderline of a challenge image's upper and lower layers.

Then, we apply the borderline to the filled hollow characters and the solid characters parts, and finally separate the upper and lower layers shown in fig 5,

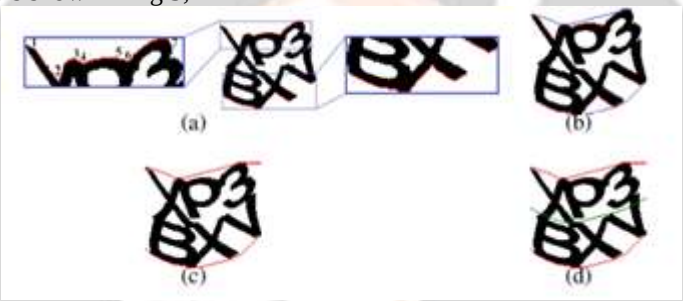

Fig-5: Solid character separation

B. Extracting Character Components

After pre-processing, we have many separated individual characters, e.g. 'V' in the example image. For images like Figure 2(b), where hollow characters and solid characters are alternatively located, we can successfully separate each individual character only via our two-dimensional segmentation approach. We then utilize CNN engine to recognize the separated single characters, and for other images containing connected characters, we adopt the method presented in 9 to break them, similar to solving a simple single-layer Captcha with three connected characters at most. Note that, we have improved the approach in 9 with regard to many details, especially the graph search algorithm and utilizing CNN to replace the template-based KNN.

C. Partition and Recognition

Graph building. We try different combinations of adjacent components to form individual characters, and build a directed graph with $n+1$ node for each image whose edges record whether a combination is feasible, where n is the total number of components

Recognizing component combinations. We use CNN as our recognition engine to determine which character each of the remaining edges in the graph is likely to be. In terms of text recognition, CNN has a much better performance than the template-based KNN .

Note that we adopted a Dynamic Programming (DP) graph search algorithm to achieve this, similar to the algorithm introduced in 9, but our algorithm is more detailed and has been improved. Instead of finding a target path with the largest confidence level sum, our algorithm manages to find a path with largest average confidence level, which makes it not only effective for fixed-string length .Captchas, but also for Captchas with a varied string length.

ALGORITHM:

```
Procedure Main ()
Begin
R =  NIL
For j=   1 to n + 1
PathList[j]   null
GetV alue (j; pahList)
Select (n + 1; pathList;R)
End

Procedure GetValue(j; pahList)
Begin
Path j =  null
if j = 1
newPath:step   0
newPath:value   0
newPath:result   null
Add(pathj; newPath)
else
foreach i in prej
foreach path in pathList[i]
n  = 1
if ExistStep(pathj; path:step + 1; n)
if path:value + con[i; j] > pathj[n]:value
pathj[n]:value   path:value + con[i; j]
pathj[n]:result   path:result + char[i; j]
else
newPath:step   path:step + 1
newPath:value   path:value + con[i; j]
newPath:result   path:result + char[i; j]
Add(pathj; newPath)
pathList[j]   pathj
End

Procedure ExistStep(pathj; step;num)
Begin
for i =  1 to path j:length
if path:step in Captcha length
if pathj[i]:step = step
num   i
return true
return false
End

Procedure Select (num; pathList; R)
Begin
averageV alue   0
foreach path in pathList[num]
if path:step in Captcha length
if path:value=path:step > averageV alue
averageV alue   path:value=path:step
R   path:result
End
```

5. CONCLUSION AND FUTURE WORK

We have for the first time systematically analyzed the robustness of a two-layer Captcha deployed by Microsoft, and answered a primary question: is the two-layer Captcha as secure as expected? on a standard desktop computer. Additionally, implementation of Captcha generating imitator to recreate the generation of Microsoft's two-layer Captcha as a training system. Compared with the common approach to access training sets, it is simple but highly effective. This concept, training a recognition engine with automatically generated data and then utilizing it to recognize real-world objects, is novel in the field of analyzing the robustness of Captchas.

The results clearly indicate that the two-layer Captcha is not secure, in contrast to early claims. To improve more on security bases we can get an idea of introducing fingerprint sensors along with captcha.

6. REFERENCES

[1] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in Cryptology‡EUROCRYPT 2003*. Springer, 2003, pp. 294–311.

[2] 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.

[3] E. Bursztein, M. Martin, and J. Mitchell, "Text-based captcha strengths and weaknesses," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 125–138.

[4] J. Yan and A. S. El Ahmad, "Usability of captchas or usability issues in captcha design," in *Proceedings of the 4th symposium on Usable privacy* s*and security*. ACM, 2008, pp. 44–52.

[5] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow captchas," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1075–1086.

[6] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[7] H. Gao, W. Wang, Y. Fan, J. Qi, and X. Liu, "The robustness of" connecting characters together" captchas." *J. Inf. Sci. Eng.*, vol. 30, no. 2, pp. 347–369, 2014.

[8] J. Yan and A. S. El, Ahmad, "A low-cost attack on a Microsoft captcha," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 543–554.

[9] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, \Vqa: Visual question answering," ArXiv preprint arXiv:1505.00468, 2015.

[10] 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 *Proc. Network and Distributed System Security Symposium (NDSS). San Diego, USA*, 2016.