[[1]](#footnote-1)

Parsing a Binary CAPTHCA

Kanav Sethi\*, Kunal Kalra, Nikhil Patwari, *SCOPE, VIT University, Vellore- 632014, Tamil Nadu, India*

***Abstract***—In this work we analyze the weaknesses of the CAPTCHA scheme used by a regular common page. Furthermore, we propose some alternatives to mitigate the weaknesses found. In the first part of this project we discuss the various methods of parsing CAPTCHA’S. Secondly, we introduce our CAPTCHA PARSER which uses python code to build a CAPTCHA training set and use that to parse the CAPTCHA. Finally, we propose the future scope and improvements on our CAPTCHA PARSER.

***Keywords***—CAPTCHA, parser, image processing, security, hacking.

—————————— ◆ ——————————

# **INTRODUCTION**

T

he abbreviation CAPTCHA stands for "Completely Automated Public Turing test to tell Computers and Humans Apart". It was coined by Louis Von Ahn et al. in 2003, and has been an important security measure in the web industry. CAPTCHA’s are based on an experiment known as the **Turing Test**. Many versions of CAPTCHA have come up since 2003, using different methods to implement the basic idea of differentiating the computer and human apart. The most widely method used is to show the user images that humans can recognize but computers cannot. Improvements include a wide variance of shapes, transformations and deformations to make the CAPTCHA recognition a difficult task for computers whereas humans face no such difficulty in recognizing the characters. Differentiating humans from computers is a very important task. Many forms of attacks can be performed by computers when they are able to act as humans. To prevent such instances and to make sure that the user submitting the query is a human and not a bot.

C:\Users\Kanav\AppData\Local\Microsoft\Windows\INetCache\Content.Word\6.gif

*Fig (1): A typical VIT FFCS CAPTCHA*

*Source: vtop.vit.ac.in/student/captcha.asp*

How a CAPTCHA works is it give the user an image with an sequence of alpha-numeric characters. The user then has to visually analyze the image and input the text sequence in the given answer box.

The text sequence is the cross-checked with the answer to that particular captcha and if they match it is believed that the user is a human and not a bot.

But as years passed CAPTCHA’s became a hassle than a security measure for some applications. CAPTCHAs are **difficult to decipher.** Although it is necessary for them to be hard but they can also decrease the data conversion rate. CAPTCHAs **carry no meaning.** People don’t understand what CAPTCHAs **are for. And** some of the users get irritated and leave the website to find a different solution. People with **lowered vision** can’t read your CAPTCHA. These reasons gave rise to the need for software that could automatically parse CAPTCHA and give accurate results. CAPTCHA parsers have alleviated these problems. They reduce the time required to access the websites. They help people who do not understand CAPTCHA to get the answer of the CAPTCHA and use the websites easily. Many companies that employ the use of bots in their operations have problems when CAPTCHA’s come into play as they hinder their operations. CAPTCHA parsers when combined with these bots help them to get past the CAPTCHA and access the sites. These are just some of the vast implementations and uses of CAPTCHA parsers.

# **Literature Review**

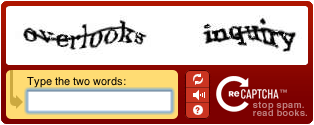
The first time that CAPTCHAs was needed was to keep out the search engine or the website abuse being done by bots. In 1997, AltaVista designed and implemented ways to discourage and block the submissions of URL that were automatically done into their search engines [1]. CAPTCHA may come in many different forms like text or image based CAPTCHA. The Bot operation is similar to the “TURING TEST” (given by Alan Turing) [2] where the other person acts like a user and the program acts like a judge. Ideally, a CAPTCHA should be highly usable by humans, provide strong security against automated attacks, and is easy to realize. [3] The CAPTCHA parser we have developed works for the CAPTCHA, which is a text based CAPTCHA. A text CAPTCHA is easy and simple to implement. Text distortions are used and the user un asked to identify the hidden text [4]. The CAPTCHA contains random strings of numbers and alphabets with lines as distortions. While this task is not much of a problem for a regular user of the Internet, it is an incredibly difficult image processing task for a program which is automated without proper context. [5] If you are interested in different types of CAPTCHAs and their working, Baljit Singh Saini et al [6] examine CAPTCHAs and their working and also describe the classification of CAPTCHAs and its areas of application and also, the guidelines for generating a CAPTCHA. A new and easy CAPCTHA implementation which is text-based is given by Xlao Ling-Zi et al. [7]. This implementation is very risky without careful design. There are many attacks on Text based CAPTCHA system because of poor security. This is highlighted by Aditya Raj et al. [8] which show that CAPTCHAs based on OCR (Optical Character Recognition) have been exploited and thus, are insecure, while the CAPTCHAs based on non-OCR are safe to use as they employ natural skill of the person’s eye to identify the picture. Shape has been deeply studied in the object recognition context so as to break the CAPTCHAs. [9] uses some of these ideas to attack Gimpy and EZ-Gimpy (CAPTCHA), two visual CAPTCHA schemes. In particular, they use the description of the features of the shapes that was introduced in [10] to build a matching system between a large database and every new query example. The method we use to break the VIT CAPTCHA is similar to the one given in [11] which is done by recognizing the characters and building a data set. Other similar methods of parsing the same CAPTCHA are given in [12]. The methods given there are somewhat similar but also different at the same time as there can be many different ways to parse a given CAPTCHA.

# **How to MAKE THE CAPTCHA STRONGER (strengthening of captcha)**

Here are some of the methods to improve in simple CAPTCHA’s and to make them stronger.

* **ReCAPTCHA**

This CAPTCHA is a part of google now. It displays a combination of random letters and words which are visually distorted to prevent dictionary or OCR based attacks. It needs to be kept in mind that the harder the CAPTCHA is to read for bots, it gets harder to be read by humans.



*Fig (2): A typical reCAPTCHA*

*Source:*[*https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/05/139961547201-recaptcha-example.gif*](https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/05/139961547201-recaptcha-example.gif)

* **Pure Audio CAPTCHAs**

These CAPTCHAS have voice or audio messages instead of text to prevent visual or OCR based attacks. reCAPTCHA has this option inbuilt.

* **Image CAPTCHAs**

Image CAPTCHAs have totally removed the possibility of OCR attacks because here the client is given a group of images and asked a question based on this group of images.



*Fig (3): A typical Image CAPTCHA*

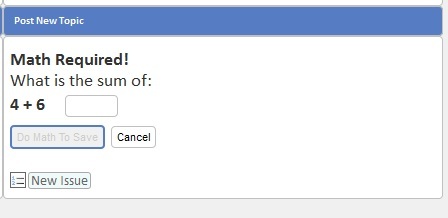
*Source:*[*https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/05/139961548102-Image-CAPTCHA.png*](https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/05/139961548102-Image-CAPTCHA.png)

* **Video CAPTCHAs**

Video CAPTCHA are the most recent but least popular version of CAPTCHA’s because they require a lot of videos and storage space and are hard to understand.

* **Simple Math/Question CAPTCHAs**

These CAPTCHA’s have a simple question and answer system where the user is asked a simple math based question like “3+3” and the user has to answer the question. Since math’s is universal and can cross the language barrier these kind CAPTCHAS are widely used.



*Fig (4): A typical Math CAPTHCA*

*Source:*[*https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/05/139961548603-Math-Captcha.jpg*](https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/05/139961548603-Math-Captcha.jpg)

* **3D CAPTCHAs**

3D CAPTCHA’s are a new concept which are more difficult to read than reCAPTCHA. They are found to be tiresome by the users.

# **EXISTING METHODS OF PARSING A CAPTCHA**

* **Client-Side Attacks**

In these type of attacks, sensitive information is communicated between the client and server in the HTML text. Due to this reason just parsing the HTML text the answers for the given CAPTCHA could be found. In some cases, only the verification component was setup at the server side resulting is the attacker just having to change the chosen CAPTCHA to bypass the security. Any type of CAPTCHA where the data set is stored in the client-side can be attacked in this way.

* **Server-Side Attacks**

This attacking methodology relies on the fact that many companies use a finite set of Captchas’ in their implementation. OCR methods can be used and Rainbow tables can be created to parse these CAPTCHA’s easily by the brute force method.

In case these CAPTCHA’s are changed periodically we can create custom CAPTCHA solvers (just like the one we made today) or outsource the problem to people on the internet.

# **PROPOSED captcha parser**

To break down any given captcha, there are some basic steps to be followed:

1. The first step is to populate a training set which consists of images of individual digits and alphabets (called mask image).
2. Mask images are produced by appropriately cropping several captcha images to form distinct alphabets and letters which are then included in the training set.
3. Prior to the cropping process, the original captcha image needs to be free of any distortions or disturbances, whose removal, has been explained later.
4. The Second step is mask matching which comprises of brute forcing and matching each letter and digit with the given captcha image.

Our captcha parsing software is divided into four modules which does each of these processes.

1. importCaptchaImages.py
2. removeDistortions.py
3. populateMaskImageSet.py
4. maskMatchtoProduceCaptcha.py

The software, as the extension of the modules suggest has been written in python language.

**importCaptchaImages.py**

This module is of prime importance as it is used to automatically download a set of captcha images from the link <https://vtop.vit.ac.in/student/captcha.asp>. This module makes use of the ‘Request’ header file which pings on the particular link and gets a byte stream as response. This byte stream is converted to an image using the BytesIO header file.

The downloaded images are saved in the directory named ‘captchaImages’. These images will be used to populate the training set upon removal of distortion in the next module.

**removeDistortions.py**

The captcha has a set of distortions, the major one being a single pixel line which cuts through several letters and numbers. This line needs to be removed in order the successfully parse the captcha.

This module also creates a binary (black and white) image which is considerably easier to work with.

Projects/Captcha%20Parser(Final)/captchaImages/15.gif

*Fig (2): ORIGINAL CAPTCHA*

Projects/Captcha%20Parser(Final)/binary1.gif *Fig (3): BINARY IMAGE WITH DISTROTIONS*

Projects/Captcha%20Parser(Final)/binary2.gif *Fig (4): BINARY IMAGE SANS DISTORTIONS*

The algorithm to create a distortion free binary image is as follows:

1. Create an image with a white background with dimensions equal to the dimensions of the original captcha image called newIm.
2. Iterate through the captcha image using a nested loop (pixel by pixel).
3. If a pixel with color green, whose upper and lower pixels have the same color as it has, we write that particular pixel in newIm with black color.

**populateMaskImageSet.py**

The distortion free image is cropped accordingly to produce a mask Image training set that is finally used to parse our captcha.

According to our code, the image is first horizontally cropped to produce all letters and alphabets and then, these images are further vertically cropped as well.

This process is automated using a well written piece of code.

Projects/Captcha%20Parser(Final)/trainingSet/3.gif Projects/Captcha%20Parser(Final)/trainingSet/2.gif

*Fig (5): VERTICALLY CROPPED MASKED IMAGES*

Projects/Captcha%20Parser(Final)/maskImages/3.gif Projects/Captcha%20Parser(Final)/maskImages/P.gif

*Fig (6): HORIZONTALLY CROPPED MASKED IMAGES*

Projects/Captcha%20Parser(Final)/maskImages/4.gif Projects/Captcha%20Parser(Final)/maskImages/5.gif Projects/Captcha%20Parser(Final)/maskImages/G.gif Projects/Captcha%20Parser(Final)/maskImages/M.gif

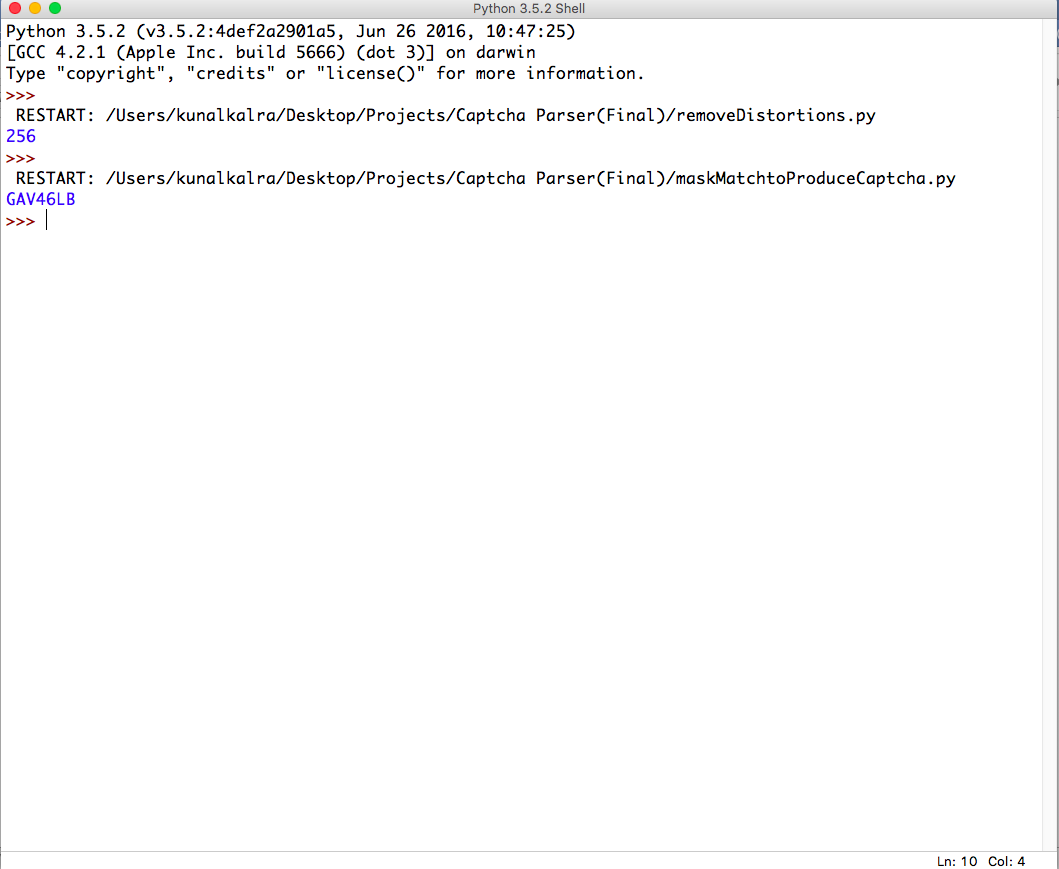
*Fig (7): EXAMPLE MASK IMAGES*

# **Result & Analysis**

**maskMatchtoProduceCaptcha.py**

This module finally is used to break down the captcha by mask matching each character from the training set with the captcha image.

We can deduce the fact that any given binary captcha can be easily parsed, given we have a complete training set for mask matching. Our Captcha parser has the ability to create training sets for any given binary captcha.



*Fig (8): Code Input and Output*

Our brute force technique, would render useless on other CAPTCHAS other than a binary captcha, unless there is a change in algorithm.

However, several machine learning algorithms have been developed to parse complex captchas, such as the ones created by Tech Giants like Google and Yahoo.

# **FUTURE SCOPE & IMPROVEMENTS**

The CAPTCHA parser demonstrated in this paper is made specifically to crack the CAPTCHA and give accurate results. Further the same code can be used to crack any text based CAPTCHA similar to the one, where in we would have tweak the code to pull the CAPTCHA data set from the owner site. This code is written in Python and works as a standalone program but can also be added to any internet application or bot via JSON scripts for automatic parsing of CAPTCHA that the bots may encounter.

**Acknowledgment**

The Authors of this paper would like to thanks Prof. Manjula R. for her support and guidance in completions of this review paper and project.

**References**

1. Aditya Raj, Ashish Jain, Tushar Pahwa and Abhimanyu Jain "Picture CAPTCHAsWith Sequencing: Their Types and Analysis," International Journal of Digital Society, vol. 1, no. 3, pp. 208-220, 2010.
2. Baljit Singh Saini and Anju Bala "A Review of Bot Protection using CAPTCHA for Web Security," IOSR Journal of Computer Engineering, 2013, pp. 36-42, 2013.
3. Clark Pope and Khushpreet Kaur, IEEE Computer Society,”Is it Human or Computer? Defending E-Commerce with Captchas”[2005]
4. Collin Hong et al., “Breaking Microsoft’s CAPTCHA”, May 2015.
5. G. Mori and J. Malik, “Recognizing objects in adversarial clutter: Breaking a visual captcha,” in Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, IEEE, vol. 1, 2003, pp. I–134.
6. Gursev Singh Kalra, “Attacking CAPTCHAs for Fun and Profit”, McAfee Foundstone Professional Services.
7. K. Arunkumar and R. Sathyaseelan, “Enhancing the Human Perceivable Image Content in CAPTCHA based BF Algorithm”, The IJES, 2015.
8. Paul Baecher et al, “CAPTCHAS: The Good, the Bad, and the Ugly”, Darmstadt University of Technology, Germany.
9. Sarika Choudhary et al, “Understanding CAPTCHA: Text and Audio Based CAPTCHA with its Applications”, IJARCSSE, June 2013.
10. S. Belongie, J. Malik, and J. Puzicha, “Shape context: A new descriptor for shape matching and object recognition,” in NIPS, vol. 2, 2000, p. 3.
11. Xiao Ling-Zi and ZHANG Yi-Chun "A Case Study of Text-Based CAPTCHA Attacks," in International Conference on Cyber-Enabled Distributed Computing and Knowledge Discover, 2012.

1. [↑](#footnote-ref-1)