Indian Journal of Economics and Business
Vol. 20 No. 3 (December, 2021)
Copyright@ Ashwin Anokha Publications & Distributions
http://www.ashwinanokha.com/IJEB.php

Analysis of Captcha's On Web Based Registration System

Mir Sajjad Hussain Talpur*[1], Masroor Ahmed Nizamani[1], Anzar Ali Shah[1], Hadi Bux Khokhar[1], Noor Ul Ain Abbasi[1], Fauzia Talpur[2], Asadullah Kehar[3], Abdul Samad Memon[4]

[1]*Information Technology Centre, Sindh Agriculture University, Tando Jam.*
[2]*University of Sindh @Laar Campus Badin*
[3]*Department of Computer Science, Shah Abdul Latif University, Khairpur*
[4]*Benazir Bhutto Shaheed University Lyari, Karachi*

Received: 18th August 2021
Revised: 19th September 2021
Accepted: 08th October 2021

Abstract: The Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA) is an integral part of the web. It is usually used to prevent a user or bot from sending multiple requests at a time, whenever a website makes a request to a server to process or to get some data. Analysis of CAPTCHAs used on various sites, how well they are implemented on testing service websites in Pakistan. In the study, the OCR engine and k-nearest neighbors algorithm (k-NN) techniques were used, which are much easier in implementation than the Vector Space Image Recognizer (VSIR). The result of the study showed that the testing service websites are vulnerable to the attack. Websites while implementing CAPTCHA don't even consider web accessibility, which makes a website inaccessible to people who have a disability.

Keywords: CAPTCHA, Automated public turning, OCR engine.

Introduction

The web has a Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA). When a website sends a request to a server process or retrieves data, it is frequently used to prevent a user or bot from sending several requests at once. CAPTCHA asks the user to recognize letters and words by displaying an image with some words or characters in it. (Ahn, et al., 2003) first invented the term CAPTCHA in their paper titled "CAPTCHA: Using Hard AI Problems for Security". Tesseract is an OCR engine that is used to decode text from an image. Tesseract requires preprocessing to improve the OCR result. The image needs to be scaled appropriately by having as many images as possible, as much contrast as possible, and having horizontally aligned text. It can be used to generate an algorithm to match the pattern of images. Image processing is a process that involves the analysis and manipulation of a digitized image. OpenCV Sharp is a cross-platform wrapper of OpenCV for the.NET Framework. OpenCV (Open-Source Computer Vision Library) is a C++ library which includes functionalities for image and video processing. OpenCV features many areas of Computer Vision. OpenCV supports many platforms, including Linux, Windows, Mac. OpenCV has a BSD license, which makes it free for academic and commercial use. OpenCV is designed for real-time applications with a strong focus on computational efficiency. OpenCV is used for image manipulations like image filtering, image segmentation, etc.

Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and cross-platform using Xamarin. It can produce both native code and managed code. Microsoft Visual Studio is the choice of development for the C # programming language.C# is a strong-typing, case-sensitive, object-oriented programming language. developed by Microsoft within.NET and standardized by the European Computer Manufacturers' Association (ECMA) and the International Organization for Standardization (ISO). C# is one of the programming languages designed for the Common Language Infrastructure (CLI).The AForge.NET Framework is an open-source framework for computer vision and artificial intelligence like image processing, neural networks, machine learning, robotics, and so on. It was used in research for image processing purposes like noise reduction, image resizing and cropping images.

K-Nearest Neighbors (KNN) is a classification algorithm for supervised learning. It searches for the closest match of the test data in feature space, caches all training samples, and predicts the response of a new sample by analyzing a certain number of the nearest neighbors of the sample using voting, calculating a weighted sum, and so on. It is referred to as learning by example, because of its prediction, as it looks for the feature vector with a known response that is closest to the given vector.

In this research, we analyze CAPTCHA's online testing services and recommend further guidelines for implementing CAPTCHA's testing services.

1. Literature Review

Brodi and Amelio (2017) explore the usability elements of image-based CAPTCHAs like the response time of the user, usage, and user preferences related to the CAPTCHA, like facial expressions. In the paper, the authors describe the role of Human Computer Interaction (HCI) in computer science and Artificial Intelligence (AI), and the main concept of HCI is usability. In the experiment, the authors used image-based CAPTCHAs in their research as it relates to HCI and cognitive psychology. The authors used four distinct types of image-based CAPTCHAs for their usability experiments. Each of them had five different images, and of those, one was related to the solution to the CAPTCHA. The authors took samples of 100 users, which were distinct by their age, internet experience, and educational level. The authors analyzed the users' response time related to the usability aspect of the CAPTCHA and the usage of statistics is connected to the time took to solve CAPTCHA including various aspects like facial expression. The rules like association were applied to the tested data. The authors concluded their research work with the co-occurrences of feature values and their association with a given response time to solve the facial expression CAPTCHA. The authors found that the facial expressions like surprised or worried were the most difficult to recognize and differentiate.

Korakakis, *et al.,* (2014), in the paper, worked on Optical Character Recognition (OCR) and Vector Space Model (VSM) techniques to break the CAPTCHA and lay down the results they got at the end of the experiment. For OCR, they used the Tesseract engine and Vector Space Image Recognizer (VSIR). In the comparative analysis, the authors used ASP.NET security image generator to generate CAPTCHA's, for use cases. The author generated three types of CAPTCHA. The first set of characters were numbers, the second one was upper-case letters, and the last one had upper-case letters and numbers. For noise removal, they converted the image to black and white pixels and then checked multiple non-white pixels in a row and changed those pixels if their sum was

less than or equal to a chopping factor that was predetermined. They had used a segmentation technique for VSIR. It was not applied for Tesseract as it uses a dictionary to identify a whole set of words as opposed to an individual character. In the end, the author concluded that the VSIR technique is good for breaking CAPTCHAs.

Gao *et al.* (2012) tried to break Yahoo CAPTCHA by being 78% successful in CAPTCHA decoding and 58% successful in individual character recognition with OCR. The authors broke Yahoo! CAPTCHA techniques by "connecting characters together". It aligns the characters together, so it is hard to make a character apart. The authors segmented the text by extracting the characters, which were previously recognized.

Pan and Zhou (2012) tried to make CAPTCHAs harder to decode by using techniques against segmentation attacks. By employing a multi-color scheme rather than the more common approach of using simple colors, colorblind individuals can view it.Their approach was to use canvas and brush coloring for CAPTCHAs so it would be much harder. To make it more difficult, they used DES encryption in their program to achieve more confusion in the brush pattern. They had proposed the empirical algorithm with the support of the Taguchi method to guarantee the quality of the color schemes. Their proposed method has at least three advantages: 1) the settings of color schemes can be fully customized by the user or developer; 2) the quality of selected colors has desirable statistical features that are ensured by the Taguchi method; and 3) the algorithm can be fully automated into computer programs. Moreover, they included examples and experiments proving the practicality and validity of their algorithm.

In Lv (2011), the author deciphered the CAPTCHA used by Baidu, using the Convolutional Neural Network to recognize the character set. The author has applied the Simard network, which is based on LeNet-5. The author used Convergence with a second algorithm called stochastic diagonal Levenberg-Marquardt method, replacing basic BP in Simard's network. By doing so, the end result they got was a recognition rate of 98.4% in the single character of CAPTCHA, and 93.5% in the overall rate.

Zhan & Wang (2010), tried to break CAPTCHAs used in Internet banking in China. Four samples were taken for this experiment. The authors presented methods for solving textual CAPTCHAs by using image processing techniques and instance learning like graying, character normalization, extracting a feature vector for each character, and recognizing, which are based on instance learning. In the first step, the authors used CAPTCHA Preprocessing like graying, thresholding, interference noise removing, character segmentation and normalization. After that, the authors implemented recognition algorithms on instance learning, like extracting feature vectors and recognition algorithms on instance learning.

Hindle, *et al.* (2008). The authorsintroduce Captcha and its features and how it not only stops bots, but also affects disabled people like the visually impaired or anyone who may have to rely on screen readers. As noted out, it is hard to create a universal captcha solver, but creating a captcha solver for similar captchas is much easier. Bitmap text-based captchas were used. The authors talk about the properties of captchas, the common ones like readability, randomness, character order, and so on. In methodology, the steps to solve captchas were like this: analysis of the captcha, determining steps to solve it, and relenting counter-techniques.

Bieniecki, *et al.* (2007), argued in their research that the image-preprocessing phase is extremely important for OCR rather than the actual recognition phase. For experimental evaluation, they used FineReader 7.0 for OCR. In their experiments, first they scanned a Polish book, then introduced various image noises and image deformations in scanned images using Paint Shop Pro 9 software. Finally, they concluded that appropriate

preprocessing algorithms should be used in the preprocessing phase before performing actual recognition. In their experiments, OCR was more sensitive to geometric deformations than noise and low resolutions.

2. Methodology

This section is about research methodology to achieve proposed research objectives. Analysis of CAPTCHAs used on various sites, how well they are implemented on testing service websites in Pakistan. In the first step, websites were discovered which were using CAPTCHAs on their websites, and after that, CAPTCHAs were downloaded so that analysis of CAPTCHAs could be done.

3.1Research Approach

To analyze how well CAPTCHAs are implemented for research purposes, they were tested by using OCR and classification techniques. In the research, the Tesseract OCR Engine and the k-nearest neighbors algorithm (k-NN) were used to break the CAPTCHAs. To reduce the noise from CAPTCHA images, the AForge.NET Library was used. The C# programming language was used as a language to write code in the Visual Studio 2017 Integrated development environment (IDE). By using OCR, these steps were followed to break CAPTCHAs:

1. Downloading CAPTCHAs
2. Image Resizing & Line Removal
3. Image Conversion to Black & White
4. Clearing Noise from the CAPTCHAs
5. CAPTCHA decoding CAPTCHAs were used for analysis.

3.2 Download CAPTCHAs

In this step, the program was written to download CAPTCHA images from the testing service websites. The following image is a sample:

Figure 3.1: Downloading CAPTCHA Image

3.3 Removing lines & Resizing Images

3.3.1 Removing lines

In this step, the lines were removed from the background so OCR engine could detect images properly. In this step A Forge libraries filter Euclidean Color Filtering was used to detect the lines color and removing them. The following image is an example:

Figure 3.2: Removing Lines from CAPTCHA

3.3.2 Resizing Images

In this step, the downloaded images were enlarged so OCR engine could detect them properly. The drawing library the one which comes pre-packaged was used. The following image is an example:

Figure 3.3: Resizing CAPTCHA Image

3.4 Converting Images into Black & white

In this step, the images were converted into black and white.

The following image is an example:

Figure 3.4: Converting CAPTCHA image into Black & white

3.5 Clearing background noise from the CAPTCHAs

In the following step, background noise from the images was removed. Using Median filter of AForge library was used.

Figure 3.5: Clearing background noise

3.6 Decoding CAPTCHAs & Saving Results

In the following step, OCR engine was used to generate results from the CAPTCHA images and save it into CSV file. Later file was converted into excel file, so that data could be compared with the images.

In the research, Microsoft Excel 365 was used to get the result. The data obtained were imported Microsoft Word 365. Excel formulas which were used in this are as under:

=COUNTIF(D1:D200, 1)

=K4/200*100

K-Nearest Neighbors algorithm was used with customized steps were followed on every website. The Captcha images downloaded in OCR were used in the K-nearest neighbors. Fallowing steps were followed.

For NTS website the used downloaded image from OCR, lines removed and Converted images and KNearest algorithm was used which come in OpenCVSharp out of the box Minimum Thresh hold was 158 and the highest one was 250

For FPSC website the download images for OCR were used, mathematical morphology Erosion was used which assigns the minimum value of surrounding pixels to each pixel of the resulting image. Threshold minimum value was 170 and the maximum value was 250.Contours were used to match the images given to train KNearest algorithm. KNearest findnearest function was used to get a result.

```
10  static void Main(string[] args)
11  {
12      string image = @"C:\Users\Masroor\Desktop\MSC\Data\NTS\";
13
14      SearchImages(image);
15      Console.WriteLine("Done");
16  }
17
18  static void SearchImages(string SourceFolder)
19  {
20      if (!Directory.Exists(SourceFolder))
21          throw new Exception("SourceFolder does not exist");
22
23      DirectoryInfo diImages = new DirectoryInfo(SourceFolder);
24      ArrayList alImages = new ArrayList();
25
26      alImages.AddRange(diImages.GetFiles("*.jpg"));
27      alImages.AddRange(diImages.GetFiles("*.png"));
28
29      foreach (FileInfo fiImage in alImages)
30      {
31          Image source = Image.FromFile(fiImage.FullName);
32          BlackWhite(source, fiImage.Name);
33      }
34  }
35
```

Figure 3.6: User defined Main function

The functions shown above in figure 3.6, the first function is the main function, it is executed when the program

start, in which there is a variable which is holding a string of folder path where images are located, after that it execute Search Images function which takes string variable as a parameter. In Search Images first it checks the directory which is given as parameter exist or not if not throw an error, after that it we created an instance of Directory Info class and passing Source Folder as a parameter to its constructor, after that created instance of Array List to hold files of specified types JPG and PNG, and looping through the array and to get image and passing it and image file name to Black White function.

```
36      static void BlackWhite(Image file, string filename)
37      {
38          Bitmap bmp = new Bitmap(file);
39          int width = bmp.Width;
40          int height = bmp.Height;
41          int[] arr = new int[225];
42          int i = 0;
43          Color p;
44
45          for (int y = 0; y < height; y++)
46          {
47              for (int x = 0; x < width; x++)
48              {
49                  p = bmp.GetPixel(x, y);
50                  int a = p.A;
51                  int r = p.R;
52                  int g = p.G;
53                  int b = p.B;
54                  int avg = (r + g + b) / 2;
55
56                  // Making Captcha solid black
57                  avg = avg < 231 ? 0 : 255;
58                  bmp.SetPixel(x, y, Color.FromArgb(a, avg, avg, avg));
59              }
60          }
61          bmp.Save(@"C:\Users\Masroor\Desktop\MSC\Thesis Data\NTS\" + filename, System.Drawing.Imaging.ImageFormat.Jpeg);
62      }
```

Figure 3.7: User defined function to convert images into Black and white

The function shown above in figure 3.7, take Image and file name as argument, create a Bitmap object of Image passed as argument, integer variables to hold the height and width of Bitmap object, after creating Integer arrays, Integer variable, declaring a color object. There are two for loops first one loop through height and the second one with width, in second loop Bitmap pixels were held in variable p which is Color object and after that Int variables are declared in which value of alpha, red, green, and blue are held, after we declare avg variable in which we add red, green, blue and divide them by 2. After that we save the value of avg if avg is less than 231 0 else 255, assigning value to set the pixels function of a bitmap variable named bmp, x and y coordinates and giving Argb value, Argb stands for Alpha, red, green and blue, it is a method of Color class, giving it Alpha value and giving other colors value average of red, green, and blue. And in the end, saving it as an image with file format for jpeg.

```
23      private static bool DownloadImages(string uri, string fileName)
24      {
25          HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);
26          HttpWebResponse response;
27          try
28          {
29              response = (HttpWebResponse)request.GetResponse();
30          }
31          catch (Exception)
32          {
33              return false;
34          }
35      #region
41          {
42              using (Stream inputStream = response.GetResponseStream())
43              using (Stream outputStream = File.OpenWrite(fileName))
44              {
45                  byte[] buffer = new byte[4096];
46                  int bytesRead;
47                  do
48                  {
49                      bytesRead = inputStream.Read(buffer, 0, buffer.Length);
50                      outputStream.Write(buffer, 0, bytesRead);
51                  } while (bytesRead != 0);
52              }
53              return true;
54          }
55          else
56              return false;
57      }
```

Figure: 3.8: User defined function to Download Captcha images

The function shown above in figure 3.8, return Boolean value, takes two strings as arguments first one is for URI which stand for Universal Resource Identifier, and the second one is filename to save download as. On line 25 HttpWebReqeust object is created and URI is given as argument and after that HttpWebResponse is declared, after that on line 29 the response to the request is saved in the response object. On line 42 the Stream object is created and response from response object is saved as a stream. On line 43 the stream object is created to create an object

```
51      public static List<CsvData> ReadCsvFile(string fileName)
52      {
53          List<CsvData> result = new List<CsvData>();
54          using (TextReader fileReader = File.OpenText(fileName))
55          {
56              var csv = new CsvReader(fileReader);
57
58              csv.Configuration.HasHeaderRecord = false;
59              csv.Configuration.Delimiter = ",";
60              csv.Configuration.Encoding = Encoding.UTF8;
61              csv.Configuration.TrimFields = true;
62              result = csv.GetRecords<CsvData>().ToList<CsvData>();
63
64              return result;
65          }
66      }
67      public class CsvData{
68          public string fileName { get; set; }
69          public string Data { get; set; }
70      }
```

Figure 3.9: User defined function to read data from CSV File

The function shown above in figure 3.9, read csv file which takes a file name as an argument and returns list object CsvData, it is class with two properties. CsvReader is a part CsvHelper, it is used to read and write Csv files.

```
33          StringBuilder csvcontent = new StringBuilder();
34          csvcontent.AppendLine("Filename, Captcha Generated");
35
36          foreach (FileInfo fiImage in alImages)
37          {
38              Image file = Image.FromFile(fiImage.FullName);
39              csvcontent.AppendLine(fiImage.Name + "," + Captcha(file));
40          }
41          string csvpath = @"C:\MS Project\NTS3.csv";
42          File.AppendAllText(csvpath, csvcontent.ToString());
43      }
44
45      static string Captcha(Image img)
46      {
47          Bitmap imagem = new Bitmap(img);
48          string reconhecido = OCR(imagem);
49
50          return reconhecido;
51      }
52
53      static string OCR(Bitmap b)
54      {
55          string res = "";
56          using (var engine = new TesseractEngine(@"tessdata", "eng", EngineMode.Default))
57          {
58              engine.SetVariable("tessedit_char_whitelist", "1234567890");
59              engine.SetVariable("tessedit_unrej_any_wd", true);
60              using (var page = engine.Process(b, PageSegMode.SingleLine))
61                  res = page.GetText().Trim().ToUpper().Replace(" ", "");
62          }
63          return res;
64      }
65  }
```

Figure 3.10: User defined function for OCR

The function is shown above in figure 3.10, in which OCR function takes bitmap as an argument and return a string as value Tesseract engine which takes a bitmap and brute force to predict word in images is used.

3. Results and Discussion

The results of this research studies indicate that the websites are vulnerable.

Table 4.1 OCR Result

Websites	Results
FPSC.GOV.PK	66%
NTS.ORG.PK	56%

Table 4.2 k-nearest neighbors Result

Websites	Results
FPSC.GOV.PK	60%
NTS.ORG.PK	84%

The research was the analysis of Captcha on Testing service websites, similar work had been done by (Zhan 2010) in which the authors conducted research on internet banking in China. In the research, resizing Captchas, removing lines, graying images, clearing noise, and image processing technique OCR and classification algorithm K-Nearest Neighbors were used in the research, in comparison with (Zhan 2010) in which they had used an image processing technique also they had used instance learning for extracting a feature vector for each character and recognizing. In the research, word segmentation was not used in OCR as it is not recommended to use in

OCR as OCR tries to predict the whole word as it is good in its rather predicting character, character segmentation was achieved for k-nearest neighbors algorithm.

4. Summary and Conclusion

This section summarizes of whole research. The research was conducted to analyze of the CAPTCHA, how well they are implemented. The result shows that the CAPTCHAs which are implemented on the testing service websites are not even secure. Even though the NTS website result shows the CAPTCHAs are strong on the web but they are small in size and they are using a similar character with little difference which would make harder to recognize for the web user, when CAPTCHA is incorrect they are using http header redirection if someone makes it false in their bot they get desired result.In the study, OCR engine and K-Nearest Neighbors algorithms were used to, which is much easier in implementation than Vector Space Image Recognizer (VSIR). The implementation of CAPTCHA even doesn't consider web accessibility which is making a website accessible to the people who have a disability. It is recommended that website administrator should upgrade their CAPTCHA to Google reCAPTCHA V2, or similar implementation and web administrators should ensure to have web accessibility on their web.

References

AForge.NET. (2017). AForge.NET :: Computer Vision, Artificial Intelligence, Robotics. Available at: http://www.aforgenet.com/ [Last visited 18 Dec. 2017].

Bieniecki, W., Grabowski, S., & Rozenberg, W. (2007, May). Image preprocessing for improving ocr accuracy. In *Perspective Technologies and Methods in MEMS Design, 2007. MEMSTECH 2007. International Conference on* (pp. 75-80). IEEE.

Brodić, D., & Amelio, A. (2017). Analysis of the Human-Computer Interaction on the Example of Image-Based CAPTCHA by Association Rule Mining. In L. Gamberini, A. Spagnolli, G. Jacucci, B. Blankertz, & J. Freeman (Eds.), Symbiotic Interaction: 5th International Workshop, Symbiotic 2016, Padua, Italy, September 29–30, 2016, Revised Selected Papers (pp. 38-51). Cham: Springer International Publishing.

C# 7. (2017). C# Guide. Available at: https://docs.microsoft.com/en-us/dotnet/csharp/ [Last visited 18 Dec. 2017].

Fpsc.gov.pk. (2017). Available at: http://www.fpsc.gov.pk/icms/user/index.php [Last visited 18 Dec. 2017].

Gao, H., Wang, W., & Fan, Y. (2012). Divide and Conquer: An Efficient Attack on Yahoo! CAPTCHA. Paper presented at the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications.

Hindle, A., Godfrey, M. W., & Holt, R. C. (2008). Reverse engineering captchas. In Reverse Engineering, 2008. WCRE'08. 15th Working Conference on (pp. 59-68). IEEE.

Korakakis, M., Magkos, E., & Mylonas, P. (2014). Automated CAPTCHA Solving: An Empirical Comparison of Selected Techniques. Paper presented at the 2014 9th International Workshop on Semantic and Social Media Adaptation and Personalization.

Lv, G. (2011). Recognition of Multi-Fontstyle Characters Based on Convolutional Neural Network. Paper

presented at the 2011 Fourth International Symposium on Computational Intelligence and Design.

Nts.org.pk. (2017). .:NTS™:.. [online] Available at: http://nts.org.pk/ [Last visited 18 Dec. 2017].

OpenCVSharp. (2017). OpenCvSharp 3.3.1. Available at: https://github.com/shimat/opencvsharp [Last visited 18 Dec. 2017].

Pan, L., & Zhou, Y. (2013). Developing an Empirical Algorithm for Protecting Text-Based CAPTCHAs against Segmentation Attacks. Paper presented at the 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. Paper presented at the Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.

Visual Studio. (2017). Visual Studio IDE, Code Editor, VSTS, & App Center. Available at: https://www.visualstudio.com/ [Last visited 18 Dec. 2017].

von Ahn, L., Blum, M., Hopper, N. J., & Langford, J. (2003). CAPTCHA: Using Hard AI Problems for Security. In E. Biham (Ed.), Advances in Cryptology — EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings (pp. 294-311). Berlin, Heidelberg: Springer Berlin Heidelberg.

Zhang, J., & Wang, X. (2010). Breaking Internet Banking CAPTCHA Based on Instance Learning. Paper presented at the 2010 International Symposium on Computational Intelligence and Design.