CaptchAll: An Improvement on the Modern Text-Based CAPTCHA

Charlie Obimbo, Andrew Halligan, Patrick De Freitas

School of Computer Science, University of Guelph, Guelph, ON N1L 1J9, CANADA

Abstract

Current CAPTCHA implementations are a source of frustration for many and yet they are an integral and necessary piece of the World Wide Web, as we know it. This paper intends to propose a solution to this ever-growing problem, CaptchAll, an easy to use and difficult to break image-based CAPTCHA. In this system, image scenes of a complex nature are presented to the user along with a piece of challenge text, asking the user to identify objects in the image with a simple click. While this task is unproblematic for the typical Internet user, it is an incredibly challenging image processing task for an automated program without proper context. Initial in-depth analysis is able to demonstrate the user-friendly nature of the system while outlining the struggle in which automated attackers will face.

Keywords: Cyber Security, CAPTCHA, Features extraction, & Radial symmetry

1. Introduction

Digital information has become extraordinarily vast while staying readily available at the same time. Though this is generally a great thing, the data's availability allows it to be subjected to malicious deeds that are not only a nuisance but can also be harmful. The sheer volume of the data being considered means that it would be impossible for a human to process and authenticate all the requests as they arrive. From this, automated software programs, or bots, have been developed to scour the web and perform activities such as post advertisements, generate email accounts for spam purposes, spam comment boards and create an uncontrollable environment in the web. Due to this dilemma, a new form of access control for web content had to be developed with the intention that humans and automated software programs could be differentiated from each other.

The CAPTCHA was invented to add the ability to protect information and actions from the malicious intents of these bots. CAPTCHA stands for "Completely Automated Public Turing test to tell Computers and Humans Apart" [1]. The general theory is that the end user will perform a relatively simple task and the computer should be able to score this attempt to determine whether the individual taking the test is in fact a human. The logic behind this is that it is incredibly difficult to teach a bot how to perform a task that involves artificial intelligence, abstract thinking, object detection or classification [2]. A very fine balance must be achieved with CAPTCHA systems since the overall goal is not to alienate users but rather legitimize them [2].

However, that fine balance is occasionally not met when users are given CAPTCHAs. There are also many other examples that ignore the usability aspect of a CAPTCHA completely and a prime example of this would be the The Math CAPTCHA [3]. The interesting point about this CAPTCHA would have to be its nature of being highly complex for the average user. Even though the user is able to lower the level of difficulty, the first
impression alone would probably scare them away completely [3]. While most mathematical based CAPTCHAs use basic operations, the Math CAPTCHA can vary with differing levels of complexity and the examples that exist range from simple arithmetic equations to partial derivatives [3].

There are a variety of different types of CAPTCHA systems available today and they can be divided into the following categories: text-based, generic image-based, specialty image-based, knowledge-based and audio-based [5]. For the discussion at hand, it is worth touching upon two of these categories:

Text-based CAPTCHAs are systems built entirely on the recognition of words and letters, typically in English, with various distortions applied [5]. They were likely the first to be introduced and possibly the first to be exploited through the use of Optical Character Recognition (OCR) systems [2]. From its inception, it has taken many forms by using different methods to increase the complexity of attack such as manipulating fonts, adding systematic noise, utilizing multi-colored backgrounds and having overlapping letters [2]. However, breaking this type of CAPTCHA usually involves two steps: letter segmentation and character recognition [2]. Also when keeping in mind the overall goal of a CAPTCHA, usability studies show a distinct difference in response times and error rates between different age groups in solving text based CAPTCHAs with different types of distortions applied to them [6].

Specialty image-based CAPTCHAs utilize image classes, which are easily distinguished by humans but challenging for machines to decipher [5]. Many difficult artificial intelligence problems can replace text-based CAPTCHAs and those, which are image-based, are usually a suggested alternative given that image recognition is considered a very difficult problem [6]. While this is desirable, it does not automatically shield this category of CAPTCHA from being exploited. To avoid exploitation, some image-based CAPTCHAs, like their text-based alternative, will use distortion via luminance, color quantization, dithering, cutting, rescaling and the introduction of noise [6]. Again, the introduction of any of these characteristics will generally contribute to poor usability, which can be detrimental to websites or services that rely on user retention.

This paper proposes an improvement to the traditional text and image based CAPTCHAs that are commercially available. The system, deemed CaptchAll, is one which combines specialty image-based CAPTCHAs with a knowledge-based component. Such a knowledge-based component is essentially a question-based method that requires common sense responses [6]. By combining these aspects, the proposed solution avoids distortion providing a quick, simple alternative with increased usability compared to the methods in use today.

2. Design

The major problem with CAPTCHA systems is the balance between making it relatively easy to solve for human but next to impossible to defeat for an automated computer program. The CaptchAll implementation addresses both of these concerns. It has excellent usability factors for end users, and yet is a very challenging subject matter for potential attackers. Since optical character recognition systems are so well researched and rapidly advancing, our system strays from the typical text-based CAPTCHA and utilizes images. Current image based CAPTCHA systems generally ask the user to select one or more objects from one or more distinct images [7]. CaptchAll intends to use a single image and ask users to select one or more different objects within this image. For the average human user, this task is something that would be easily accomplished and far less mentally strenuous than decoding garbled characters. For a computer, object recognition and classification is a very difficult problem given that programs really only are able to examine an image at the pixel level and are not able to see the entire context that such an image presents [8].

The CaptchAll system is derived of two distinct components that work in a cohesive manner to accomplish the collective goal of improving the modern CAPTCHA. These components consist of an image annotation tool and a proof of concept implementation of the CAPTCHA itself.

2.1. Database Design

The CaptchAll system uses a database of point-encoded images with a related challenge question. Users must recognize the point with a tolerable radius in order to be identified as a human. As such, some simple database
models had to be defined in order to store several pieces of information to allow the system to function. A diagram depicting these models can be seen below in Figure 1.

The ImageAnnotation model defines storage for a challenge question of up to 100 characters in length. It also allows an appropriate URL to be stored for the particular image that is to be displayed to the user. While currently the URLs that are being stored link to available image hosting providers (Flickr), it would likely be a good idea to personally store the images being used if the system were to be developed further.

The AnnotationPoint model stores the \(x\) and \(y\) coordinates for a given object inside an image. This model has a foreign key relationship to the ImageAnnotation model. While the \(x\) and \(y\) coordinates could have easily been included in the ImageAnnotation model, the system was designed in this way so that it could be extended to allow for multiple points being associated with a single ImageAnnotation in the future.

The CaptchaSession model stores a unique session ID (SHA-512 hash) which is associated with a particular CAPTCHA request by a user. Whenever the user requests to see a new CAPTCHA, a new session ID will be generated and stored for reference purposes. This model also includes a Boolean field to mark whether or not the particular session ID has already been used. The CaptchaSession model also foreign keys out to the ImageAnnotation model since each CaptchaSession must be related to a random image from those available in the database.

In the proof-of-concept implementation, an SQLite database has been used for simplicity. When deployment to production becomes a possibility, this will be replaced with a more robust and scalable alternative.

2.2. Image Annotation Tool

The Image Annotation Tool provides the backbone for the CaptchAll implementation. Since any image used in the system must be hand encoded with object points, it was determined that a tool to aid in this process would be very beneficial. The main responsibility of the annotation tool is to fetch new images from a publicly available source (The Flickr Commons project), allow a user to enter a challenge question, and allow a user to select an appropriate object in the image that represents the challenge.

The annotation tool makes use of the AnnotationPoint and ImageAnnotation models that have been discussed. The back-end for the tool translates the user’s input into the appropriate models and persists them to storage.

The decision was made to implement the annotation tool as a web-based project since it would be vastly easier to host and allow for multiple contributors. Knowing that seeding the database with a multitude of images is very important to CaptchAll’s success, a platform-independent web project seemed like the optimal choice. Django [9], a Python web framework was used to implement all of the backend request and data handling that the annotation tool provides. An annotation form is provided to the front end with a field for the challenge text and three hidden fields for the \(x\) coordinate, \(y\) coordinate, and image URL. The fields are hidden for simplicity since they don’t necessarily concern a user who is utilizing the tool.
2.3. CaptchAll Implementation

A ten pixel radius is initially selected given the fact that it is reasonable to assume that the average user will be able to pinpoint image features with this level of accuracy. In extended testing and analysis, this radius can be adjusted so as to more accurately account for click tolerance levels.

In terms of implementation, there are two main components that CaptchAll consists of which include a JSON API for generation and verification of CAPTCHAs, and a front-end proof-of-concept design.

The JSON API is the core of the CaptchAll implementation. It is very simple in the sense that it consists of a mere two endpoints. Like the annotation tool, the JSON API was constructed using the Django web framework in Python. The Django-Rest-Framework package was also used to assist in providing some base functionality for the API such as appropriate serialization and response error coding. This particular package was chosen for implementation since it provides niceties including throttling and authorization out of the box.

The first API endpoint is used when a user requests a new CAPTCHA image. A new session ID is generated upon receiving such a request from a user. This session ID is used to reference the particular CAPTCHA in the database so that it can be looked up after to perform the verification step. A random ImageAnnotation object is pulled from the database.

The second API endpoint will be used when a user intends to check their CAPTCHA response against the known encoded point in the database. The user will POST the session ID of the current CAPTCHA, and also the individual x and y coordinates of the point that they have clicked (this will be automatically posted on behalf of the user once they indicate that they want to submit). This data will be subjected to validation using a simple algorithm that has been outlined below.

As can be seen, the session ID is an important security mechanism. Whenever a session ID is submitted and found in the database, it will be marked as used. This ensures that malicious users are unable to reuse known correct responses for past session IDs. One pitfall to this method is that an abundance of stale session IDs will be left ultimately with no real purpose. A more feature filled implementation could use a pruning background process that could remove these stale entries from the database to free up disk space.

The front-end proof-of-concept system makes extensive use of the JSON API that has been detailed. Its layout has been developed using HTML and CSS while the client-side logic has been built with CoffeeScript. When the page initially loads, a call is made to the CAPTCHA generation API endpoint. The image URL in the response, along with the challenge text, is displayed to the user. The end user is able to click the presented image and a target indicator will appear at their selected point. Upon submitting, this selected point along with the received session ID is supplied to the verification API endpoint to determine whether the user can be validated as a human.
3. Analysis

Further research into other areas has allowed the authors to see how their implementation of the CaptchAll system would stand up by analyzing several different techniques in the field of image processing and by looking at the favorable probability calculations behind the solution. From these specific areas it was determined that multiple existing methods are available that try to detect certain features in an image. For further clarification, the concept of the implementation presented in this paper is to allow the user to select a feature in the image that pertains to the topic of the challenge question being asked. From here on out, this will be referred to as an “image feature”.

3.1. Brute Force

Like all other CAPTCHAs, the CaptchAll implementation is subject to brute force methods. An image is a set of pixels can be represented as an $n \times m$ matrix where each entry within the matrix represents a pixel that can be displayed on an electronic screen [10].

\[
p = \begin{pmatrix}
p_{11} & p_{12} & \cdots & p_{1n} \\
p_{21} & p_{22} & \cdots & p_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
p_{m1} & p_{m2} & \cdots & p_{mn}
\end{pmatrix}
\]

The pixels can then be randomly iterated over using programming languages such as Python while keeping track of each of the possible solutions for the actual matrix. In total the matrix would contain $n \times m$ possible solutions of which only $\approx \pi \times 5^2 \approx 78$ pixels with an error radius of about 5 pixels around the solution pixel. Given this, the probability of success would be approximately $\phi$. Now $\phi = \pi^2 / nm$.

So in the case of a 500 $\times$ 500 pixel image, the probability, $\alpha$, of successfully guessing the correct pixel would be approximately 0.0003. By adding a requirement that more than one point must be clicked, this probability becomes much lower.

3.2. Image Processing

Since the CaptchAll implementation involves the use of images it was required to consider techniques in image processing that could be used against the system to exploit it. Some of the areas of concern were those of feature detection, feature extraction, and content based image retrieval (CBIR). These areas are rather accessible and can be fairly simple to implement which makes them prime possible threats to the CaptchAll system and thus it makes sense to further discuss these in detail.

1) Feature Detection: Within the subject of feature detection there exists numerous methods of realizing points of interest in facial and non facial recognition by generally trying to mimic the human behavior [11]. The idea is that humans tend to focus on regions with important and interesting information such as the eyes [11]. According to Gareth Loy and Alexander Zelinsky a lot of these algorithms use symmetry to get the specific points of interest. Thus it would be common to select areas such as eyes, chins and noses.

Iconic Description (Definition by Extension) usually involves parts of images that are retrieved via blobs which is defined to be “an image area that isolate one visual manifestation of an Object of interest” [12]. The other method is a patch, which is defined to be “a thumbnail (or multiple) that isolates one visual manifestation of one salient part of one object of interest” [12]. The problem still arises which method to utilize in determining what to extract from the image, which normally requires substantial user interaction. This would in turn destroy the efficiency of using an automated software program to break the CAPTCHA.

2) Feature Extraction: Another area that is common in image processing is feature extraction where the areas of interest are identified and isolated in an image. In this topic, image classes would be defined either by an iconic or linguistic description or may be termed as definition by extension and definition by intension. Linguistic descriptions will be discussed in the Content Based Image Retrieval section [12].

3) Content Based Image Retrieval (CBIR): This one, as mentioned earlier, entails definition by intension,
(describing image classes linguistically using ontology) [12]. This is widely used in CBIR where a language is built over a ontological domain to assist in searching [12]. However, many domains can present weaknesses in situations that involve uncertainty or vagueness. These ontological domains tend to fail due to their inability to determine an outcome [13]. Computer systems require the need for a concrete domain, that is a domain with definitive answers that can be interpreted as values [13]. Most of the information that is given is done so in an abstract sense. When given the word rose a system may produce many results where rose could be a person or flower or perhaps more details would simply be required to make any decision. Here, the underlying problem which all computers face is the area of decision-making.

3.3. Usability

The CaptchAll system was developed with user-friendliness as its prime and standout feature. The concept of selecting a location in an image which pertains to a challenge question is incredibly simple for any user while being complex for an automated piece of software. Though it has not been explicitly tested, the authors conclude that time to solve the CaptchAll implementation would be substantially less than the time required to solve a distorted text-based CAPTCHA. Since the images presented to the user have no added distortion or noise, the proposed system certainly places itself above several others that take this approach and decrease usability [6]. Initial responses to the CaptchAll system have been favorable in that a mere two mouse clicks are required to successfully submit a solution.

4. Conclusion

In summary, an improvement on current CAPTCHA systems was proposed by combining image-based object recognition techniques with a knowledge-based component. The system asks users to identify certain objects within a scene and evaluates responses based on a certain level of tolerance to deem the user as human. An annotation system was also implemented to aid in the initial seeding of the image database with content. Evaluation of the system was found to have considerable difficulty for attackers with the best method of breaking being brute force with very low probability. On top of this, the system was determined to be exceptionally user-friendly by allowing users the possibility of solving it with as little as two clicks with none of the usual strain. While the system described is a step in the right direction based on current offerings, it is in no way complete and thus a variety of steps must be taken to further the possible benefits that it presents.

References