ESCAPT: Easy Strategies for Computers to Avoid the Public Turing Test

Andrew Dempsey
Mentor: Ming Chow

Fall 2014

Revised Spring 2016

Abstract

In this paper, we will examine the effectiveness of the Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA). The test is in use on almost all popular websites today to differentiate between human users and software, reducing the threat of bots exploiting the sites for malicious purposes. CAPTCHA generally requires users to interpret distorted images or audio segments that are easily deciphered by humans but are too obfuscated for algorithms to properly dissect. However, as computer vision and hearing technology advances, the efficacy of these techniques may be reduced. Here, we will study the mechanisms of programs used to deceive CAPTCHA and explore more advanced methods of the test that can provide robust defenses against modern automated systems.

Contents

1 Introduction

1.1 The Origin of CAPTCHA

In the 1980's, online chat rooms began to rise in popularity. With these chat rooms came word detection, written by site owners to raise alarms if members began talking about unscrupulous subjects.[2] To counteract these measures, many users began adopting a method of typing to obscure the content of their messages to software while remaining obvious to other humans. Users would substitute similar characters for others (or other sets of characters) to avoid using normal letters, e.g., "/\/\" for "M", "3" for "E" etc.[2] This method of typing would later come to be known as 'leetspeak.' In the 1990's, this pattern was further developed by AltaVista to defend their search engine against malicious bots by devising a system to deflect requests from software and exclusively respond to requests from human users.[2] Their defense consisted of an image of distorted text presented to the user, who was then prompted to transcribe the contents into a textbox. The goal was to devise an image generator that would create text too obfuscated for optical character recognition engines to recognize but still interpretable by humans.[2] This system was the birth of the Completely Automated Public Turing test to tell Humans and Humans Apart (CAPTCHA), which has now been in use on millions of websites since its invention.[2]

1.2 CAPTCHA Variants

1.2.1 Visual

The most common form of CAPTCHA presents users with an image containing distorted text. The user's task is simple: to transcribe the contents of the image into a textbox. The letters in the image are warped and rotated such that they are ostensibly unable to be processed by optical character recognition (OCR) algorithms but may still be read naturally by humans.[3] Additionally, the images will often contain multicolored backgrounds with irregular patterns intended to introduce visual noise, increasing the chance of error as an algorithm runs its course. However, visual CAPTCHAs have become widely notorious as they've gained popularity, since many (human) users have difficulty decoding the messages as well, especially those with visual disabilities.[3]

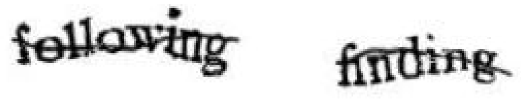


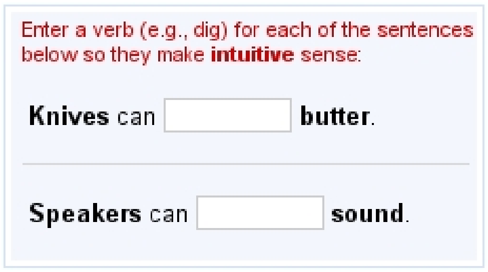
Figure 1: Example of a visual CAPTCHA image

1.2.2 Audio

In an audio CAPTCHA, the user is required to transcribe characters spoken in a brief audio clip. The voice speaking the characters is often either distorted or of a very low sound quality; the source of the voice may change from character-to-character as well (e.g., older man to younger woman). Again, audio CAPTCHA challenges are intentionally difficult for computers, but ostensibly easy for humans, to understand. However, they suffer from usability issues as well, as it is common for human users to have difficulty correctly hearing the spoken characters.[2] Audio CAPTCHAs additionally introduce assumptions about the user's environment (i.e., that they have access to speakers or headphones and are in a non-distracting location), but also provide an alternative option to visually impaired users while maintaining a similar standard of security.

1.2.3 Semantic

Semantic CAPTCHAs do not separate human users from software by testing perceptual competence, but rather the ability to connect abstract concepts. For example, some variants may ask the user to fill in the blank in a sentence such that it is meaningful: as in Figure 2, there are only a handful of intuitive words that could be placed in the sentence 'Knives can ________ butter' for it to make sense, and determining these words can be a fairly nontrivial task for software. Another possible variant of a semantic CAPTCHA requires the user to identify an object. This can take the form of an 'identify the odd one out' challenge, given a series of images (e.g., 'click the dog' when presented with six pictures, five of which are pictures of cats and one is of a dog), or asking the user to select objects of a specified color or size.[4]


Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Knives can ________ **butter**.

Speakers can ________ **sound**.

Figure 2: Example of a semantic CAPTCHA

2 Exploitation

CAPTCHAs generally perform sufficiently well enough to protect websites from malicious robots. However, there is a critical flaw in their nature: the effectiveness of CAPTCHAs is dependent on the rudimentary abilities of modern-day artificial intelligence.[5] As AI algorithms become more advanced, the lines between human users and software are blurring: many researchers have developed methods specifically to fool CAPTCHA systems that are accurate enough to potentially pose significant problems.[5] In this section, we will detail some of the methods researchers have used to exploit CAPTCHAs. In particular, we will focus on visual tests.

2.1 Optical Character Recognition

The supporting material of this paper includes an OpenCV script that takes a CAPTCHA image as input and outputs the text contents as a string. The script is intended to decode images generated by a simple variant of the SnapHost CAPTCHA, such as the following figure:

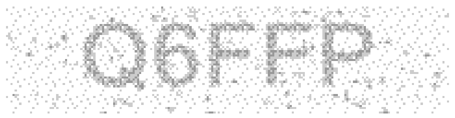


Figure 3: SnapHost Visual CAPTCHA as presented to the user

The first step is to reduce the level of noise in the image. The chaotic, dotted background can make it difficult for the program to detect the edges of individual letters, so we use a method known as Non-Local Means Denoising to eliminate distracting artifacts. In essence, NLMD iterates through each pixel of an image and overwrites its brightness value with a weighted average of all other pixels, with priority given to pixels of a similar color.[8] A specified parameter allows us to compute this average by means of rectangles; i.e., instead of individual pixels, we work with 'neighborhoods' of pixels. A high rectangle area will produce a smoother result, but may incur a loss of detail from the original image. The result of running NLMD on Figure 4 is as follows:

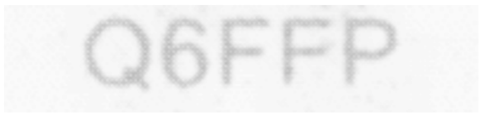


Figure 4: CAPTCHA image with Non-Local Means Denoising applied

From here, the next goal is to extract each individual character into its own segment. We first increase the contrast between the letters and the background to differentiate between the two areas. To do so, we use a method known as thresholding. Once again, we iterate through each pixel of the image to compute a mean brightness value.[9] Any pixels that differ significantly from the mean will be colored white, and all others will be colored black, resulting in a binarized image. The result of running the algorithm on Figure 5 is as follows:

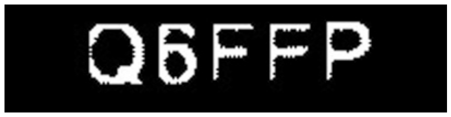


Figure 5: CAPTCHA image after a thresholding pass

The image is now ready for segmentation using OpenCV's findContours method, which uses an algorithm developed in [10]. The algorithm is capable of mapping edges precisely along curves, but for our purposes, we only need to construct a bounding box enclosing each letter. Running the algorithm over the binarized image, we achieve the result:

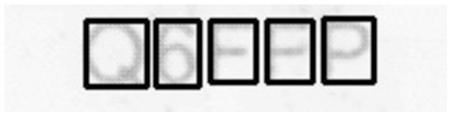


Figure 6: CAPTCHA image with individually extracted characters

Now that we have clean, well-separated representations of each individual character, we can attempt to classify their values. We load a training set as a reference to map each image representation to its true, text value:

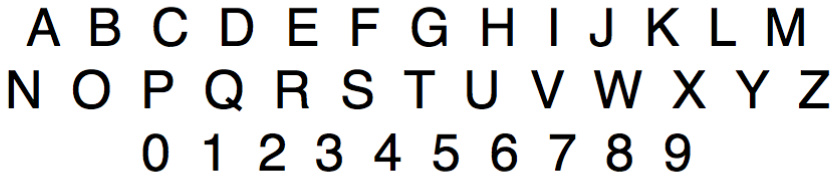


Figure 7: Training set for classifying characters

For each character in the challenge, we compute its similarity to each character in the training set. We subtract the brightness value at each pixel in the test character from the training character; this will make it so any intersecting pixels will be colored black, and non-intersecting pixels are left as-is. Afterward, we compute the number of black pixels in the subtracted image; the difference with the largest number of black pixels will have had the greatest intersection and, thus, the highest similarity, and so we choose that character as its label. Running on the input from Figure 4, the program successfully outputs "Q6FFP."

2.2 Learning

While the previous technique works decently well on this particular form of CAPTCHA, it does not fare as well on more complicated variants. More advanced techniques rely on machine learning algorithms to be able to identify characters presented in a much more obfuscated context. In [11], researchers created a training set of characters with multiple fonts, sizes, and rotation angles. Using a method known as a Support Vector Machine, the researchers developed a model used to classify future CAPTCHAs based on their training set. A support vector machine functions by constructing an $n-$dimensional hyperplane to separate classes by a maximal margin.

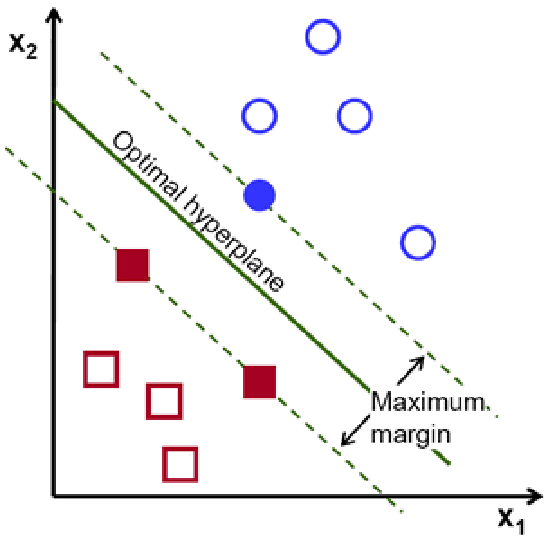


Figure 8: Visual Representation of a 2D SVM - from `http://docs.opencv.org/_images/optimal-hyperplane.png`

The points are placed in space by a measure of their image contents. The machine runs through training examples and adjusts its hyperplane to compensate for incorrect guesses, allowing it to classify future examples mroe accurately. Using these methods, even when programs generate CAPTCHA images unpredictably, the software can become sufficiently well-versed to a degree where it can successfully identify many previously unseen forms of characters[11].

2.3 Farming

While extensive work has been put into developing OCR and machine learning algorithms to decode CAPTCHA challenges, a much simpler method has proven to be quite successful: human vision. Numerous bots have been developed to scrape web pages for CAPTCHA images and automatically send them to a human user. The human user enters the answer and sends the result back to the bot, allowing it to continue with its automated tasks[12]. Because CAPTCHAs are inherently designed to be built for humans, this is very difficult to prevent[12]. As a result, a great degree of care must be taken to verify the identity of users with methods that go beyond simply decoding an image, as will be discussed in Section 4.

3 To the Community

While CAPTCHA has garnered a notorious reputation due its aggravating nature to normal users, it is an essential tool for any large scale internet service. Without any verification to ensure a user is human, websites can very easily be abused by robots for malicious purposes, affecting not only the owners of the site but its users as well. For example, many spam emails are able to be sent out due to insecure CAPTCHA systems - robots are able to automatically create numerous accounts on email services and use them to send mass emails to many users at a time, often with links to websites intended to steal information. With several options available for CAPTCHA services, it can be difficult to determine which methods offer the highest level of security without sacrificing the user's experience. The next section will introduce common flaws in CAPTCHA systems and what can be done to fix them.

4 Action Items

4.1 Challenge Generation

For user experience, it is often desirable to present challenges in the form of real words. Rather than having to scrutinize each individual letter to complete the puzzle, users can interpret the whole word much more quickly, even if some letters are independently difficult to discern. However, as with passwords, using common words in challenges can make services susceptible to dictionary attacks. Yahoo's Gimpy CAPTCHA utilized a fairly small dictionary of words when presenting challenges to the user[13]. As a result, researchers were able to build training sets based on the words that the puzzle presented (which were repeated in challenges often), and thus were able to train robots to solve the CAPTCHAs with a high amount of accuracy[13]. Thus, using dictionaries to present challenges can be quite dangerous and may require further obfuscation of the letters than normal, thus defeating the initial purpose of the convenience.

4.2 Network Security

As discussed in section 2.3, one of the biggest problems in CAPTCHA defense is robots sending images to human users to solve. While this is difficult to defend against completely, there are certain precautions that can significantly help in inhibiting attacks.

4.2.1 Same-Origin Policy

The Same-Origin Policy allows scripts to access the Document Object Model and resources of other files that all initiate from the same server. This is useful, for example, when asynchronously loading data on a webpage - JavaScript code can access data files that may be stored on the same server as the web page[14]. However, many bots are able to abuse this and send CAPTCHA challenges to human users by scraping web pages for the images[12]. Thus, CAPTCHA images should originate from an external domain in a separate request from the other resources of the webpage, and should disable Cross Origin Resource Sharing as well. Modern web browsers will not permit scripts to copy images loaded in this manner directly from the machine[14]. Even if the user copies an image pixel by pixel into a JavaScript canvas, for example, browsers can recognize the attempt to copy a 'tainted' image and block the operation[14]. Thus, in order to obtain the image to send to a human user, robots would need to make a separate request to the CAPTCHA API, but would then obtain a different challenge image and thus not gain any useful information.

4.2.2 HTTPS

It is important for CAPTCHA images to be sent over a secure protocol. If images are sent without encryption, they can be easily intercepted by a robot and sent to a farming service to be solved[15].

4.3 User Behavior

In December 2014, Google's reCAPTCHA, a very popular CAPTCHA service, announced that in lieu of puzzles, users would now encounter a checkbox to confirm they 'are not a robot.'[16]

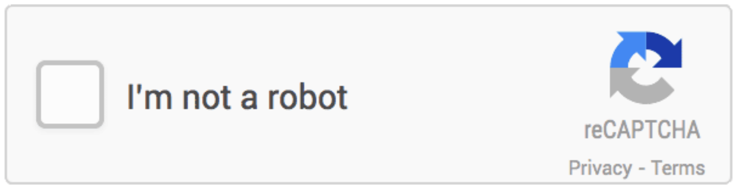


Figure 9: Modern format of Google reCAPTCHA

While the test initially seems too simple to be effective, reCAPTCHA extensively monitors the manner in which a user interacts with a webpage before pressing the checkbox[16]. The test can examine factors such as the speed at which letters were typed in previous textboxes, or the amount of errors made by the user and when[17]. Using this data, the service can determine whether or not it is appropriate to issue a challenge. If the user seems as though they are behaving as a human would, they can continue without any further evaluation. Otherwise, they are issued a challenge in the form of a visual, audio, or semantic CAPTCHA, and must solve it to continue[16]. The challenge can be skewed in difficulty, depending on how confident the service is that the site is being visited by a robot.

Google's motivation for focusing on alternative factors, rather than puzzles, stems not only from a usability standpoint, but from a security standpoint. In April 2014, Google Security published a report detailing that 99.8% of visual CAPTCHAs were able to be solved perfectly by artificial intelligence[5]. As a result, the switch to alternative forms of security was necessary not only to keep users happy, but to ensure the continued functionality of the service.

5 Conclusion

In this paper, we studied how varying implementations of CAPTCHA can affect the robustness of a website's defense against attackers. CAPTCHA is a very useful tool for verifying the identity of users and ensuring that robots are not abusing website resources for malicious purposes, but care must be taken to ensure the test will actually produce the desired results. We detailed methods by which current, primitive implementations can be easily abused, and discussed modern methods that developers can adopt to ensure the continued efficacy of the test.

References

[1] Artificial Intelligence — The Turing Test. (1999, January 1). Retrieved from http://psych.utoronto.ca/users/reingold/courses/ai/turing.html

[2] Yale, B. (2014, September 10). The CAPTCHA: A History, A Problem, Possible Solutions. Retrieved from http://www.informit.com/blogs/blog.aspx?uk=Why-Are-CAPTCHAs-So-Awful

[3] Strickland, J. (n.d.). How CAPTCHA Works. Retrieved from http://computer.howstuffworks.com/captcha.htm

[4] (2008, April 23). Retrieved from http://tech.slashdot.org/story/08/04/23/0044223/next-generation-captcha-exploits-the-semantic-gap

[5] Shet, V. (2014, April 16). Street View and reCAPTCHA technology just got smarter. Retrieved from http://googleonlinesecurity.blogspot.com/2014/04/street-view-and-recaptcha-technology.html

[6] Boyter, B. (2013, July 7). Decoding CAPTCHA's. Retrieved from http://www.boyter.org/decoding-captchas/

[7] K, A. (2012, March 8). Simple Digit Recognition OCR in OpenCV-Python. Retrieved from http://stackoverflow.com/questions/9413216/simple-digit-recognition-ocr-in-opencv-python

[8] Image Denoising. (n.d.). Retrieved from http://docs.opencv.org/trunk/doc/py_tutorials/py_photo/py_non_local_means /py_non_local_means.html

[9] K, A. (2013, May 19). Thresholding. Retrieved from http://opencvpython.blogspot.com/2013/05/thresholding.html

[10] Suzuki, S., & Be, K. (1985). Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1), 32-46. Retrieved from http://www.sciencedirect.com/science/article/pii/0734189X85900167

[11] Fiot, J., Paucher, R. (2009). The Captchaker Project.

[12] Danchev, D. (2008, August 29). Inside India's CAPTCHA solving economy. Retrieved from http://www.zdnet.com/article/inside-indias-captcha-solving-economy/

[13] Gupta, M. (2012). Threats, Countermeasures, and Advances in Applied Information Security (pp. 382-383).

[14] CORS enabled image. (n.d.). Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML/CORS_enabled_image

[15] Extending Burp Suite to solve reCAPTCHA. (2012, January 19). Retrieved from https://www.idontplaydarts.com/2012/01/extending-burp-suite-to-solve-recaptcha/

[16] Shet, V. (2014, December 3). Are you a robot? Introducing "No CAPTCHA reCAPTCHA" Retrieved from http://googleonlinesecurity.blogspot.com/2014/12/are-you-robot-introducing-no-captcha.html

[17] Shet, V. (2014, February 14). CAPTCHAs that capture your heart. Retrieved from http://googleonlinesecurity.blogspot.com/2014/02/captchas-that-capture-your-heart.html