AuPic CAPTCHA

Mohammed Musaddiq Sandeep Gupta Sigmund Chow

December 10, 2017

1 Introduction

The internet security community has constantly tried to come up with effective and efficient Captcha (Completely Automated Public Turing Test to Tell Computers and Humans Apart) solutions to[1] -

- Protect website registration
- Prevent comment spam in blogs
- Protect email addresses from scrapers
- Protect online polls
- Prevent dictionary attacks

While Captchas have become more sophisticated over the years, the programs developed to circumvent them (hereafter referred to as *bots*) have become equally adept at cracking them. A major challenge in building more secure captchas has been the human dropout (or cart abandonment) rate[2] i.e. ensuring they are still simple enough for humans to solve, without compromising on *bot*-detection.

We surveyed the various Captcha variants seen so far - Text, Math, Image, Animated, Audio Captchas, reCaptcha and studied the way each of these have been attacked/bypassed successfully. An obvious observation was - advancement in various domains in Computer Science - especially Artificial Intelligence, Machine Learning, Natural Language Processing and Computer Vision have enabled *bots* to behave more human-like (or atleast have similar capabilities) which makes it even more difficult to design secure Captchas.

Therefore, being conscious of the fact that designing an unbreakable Captcha would be near impossible, we attempt to improve upon existing methodologies in terms of making a *bot*'s work harder. An observation while understanding the way *bots* typically attacked existing Captchas was that they performed tailored attacks pertaining to a particular type of Captcha i.e. using computer vision techniques against image Captchas[3], using phonetic mapping against audio Captchas[4] and so on. So, we propose a hybrid Captcha approach - AuPic Captcha which is based on how humans can process information from different mediums (visual and auditory in this context) at a time, correlate and arrive at a response. This approach is in the spirit of a Turing Test - making it more human like and making the *bot* actually think rather than be able to overcome the challenge by simply performing a programmed set of routines as seen in the case of independent image and audio Captcha.

2 Related Work

The Captcha approaches seen so far involve using a single medium (image/audio) independently. We have not come across approaches where the two mediums are linked as in our proposed method. Following are the typical attacks performed on existing approaches:

2.1 Image Captchas/reCaptcha

Attack: Given a question like "Select all images matching the sample image & textual hint" - extract the sample & candidate images and the hint describing the sample image content (e.g. "wine"). Then, use services like Google Reverse Image Search (Figure 1) to select matching candidate images

Figure 1: Google Reverse Image Search (GRIS) for obtaining the description of an image[3]

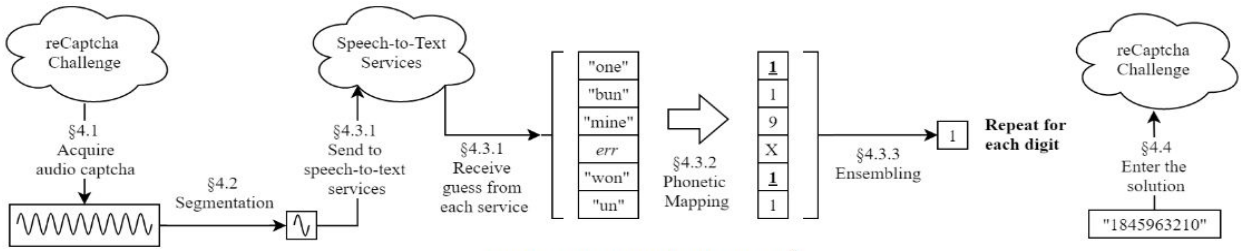

Figure 2: Attacking audio Captchas[4]

2.2 Audio Captchas/reCaptcha

Attack: Given an audio challenge like "Type the numbers: two five seven …. (say, upto 10 digits)" - segment the audio into per-digit sound bytes and analyze the sound bytes using Speech Recognition services like Google Cloud, IBM Bluemix to figure out the correct 10-digit sequence (Figure 2)

3 Adversary Model

In our adversary model, the adversary possesses modest computational resources of its own and leverages external services to help process and solve each type of Captcha challenge - such as Google Reverse Image Search for image captchas and Google Cloud for audio captchas as mentioned in Section 2.1 and Section 2.2 respectively. More specifically, the adversary does not possess superior Artificial Intelligence capabilities (such as those possessed by Apple's Siri or Microsoft's Cortana) to comprehend the semantics of a posed challenge on its own.

4 Methodology

4.1 Initial Approach

Initially, we explored a GIF-based Captcha approach where a GIF consisting of say, 10 random frames is played and a textual question is asked pertaining to the same such as -

- Count frames having 1 or more specified objects (like trees, cars) in them
- Count frames not having the specified objects (like trees, cars) in them
- Add the count of frames having the specified objects (like trees, cars) with a specified integer value

The response needs to be entered into a text field and submitted for verification.

We had assumed this approach to have a supposed advantage wherein the attacker needed to do more amount of processing for a GIF compared to existing image Captchas. However, we came across an animated-Captcha breaking approach[5] that when combined with the image-captcha breaking approach seen in Section 2.1 negated our assumption. Next, we tried to strengthen our approach by making the questions relatively harder but encountered another setback in terms of the possible answer space. Since we cannot increase the number of answer-relevant frames in a 10-frame GIF to anything above 5 (because it would make it harder for the human user to keep track), the answer space would always be limited to 5. So a *bot* could simply try different values within that range of 5 possible values as a response without bothering to solve the actual challenge and still succeed with a probability of 1/6 - a serious risk.

4.2 Reworked Approach

In order to address the drawbacks of our initial approach described in Section 4.1, we decided to -

- Shift to a 3x3 or 4x4 grid of images where the user needs to select one or more images as a response - thereby increasing the answer space considerably
- Use an audio captcha component in conjunction with the images for posing the question instead of displaying it in text form - thereby necessitating significantly more amount of processing

4.2.1 Process

(Figure 3) Initially, we have a labelled images dataset and a question templates repository. For constructing a challenge consisting of a 3x3 images grid, we pick 9 images by either choosing a random label and

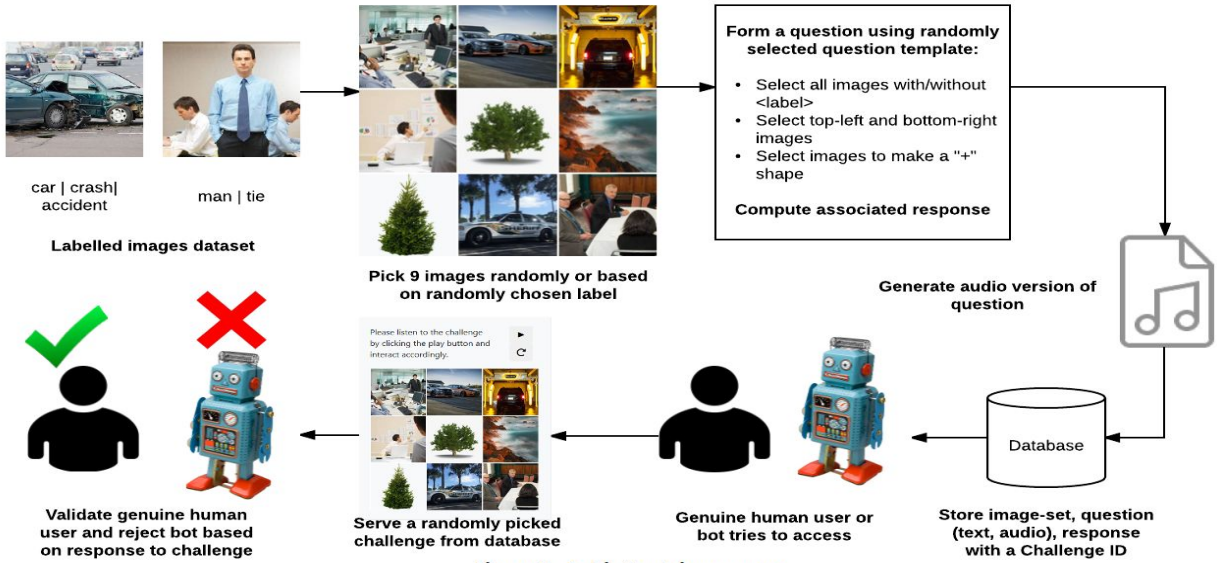

Figure 3: AuPic Captcha process

getting a mix of relevant + unrelated images (to be used as noise/distraction) or by choosing 9 images randomly from the dataset. Based on the images, we can pick a suitable question template like -

- Select all images with/without <label(s)>
- Select top-left and bottom-right images
- Select images from grid to make a "+" shape

and phrase it accordingly in the context of the current challenge. Compute the associated response and also generate an audio version of the question's textual part. Tag the various challenge components with a challenge Id and store in database. Repeat the process for generating more challenges. All this can be done offline, so that when the server gets an actual user/*bot* request in real-time, a random challenge Id from the existing challenges pool can be picked and served followed by validation.

5 Implementation

We implemented a proof of concept (Demo | Source) based on Section 4.2 using JavaScript. As per Figure 3 - a grid of 9 images is displayed along with an audio instruction which is used to make appropriate selections as response. An incorrect response will result in a new challenge being served.

The audio instructions were generated offline using *gTTS* - a python library which uses Google Text to Speech (TTS) API[6]. The images in the grid were manually selected from the internet. In a full-fledged implementation, we assume that websites will have access to a large images dataset to increase the challenge's randomness - thus preventing replay attacks when used with several random question templates.

A typical *bot* attack can use the audio-captcha breaker mentioned in Section 2.2 to translate the audio question to text with an average time of 22.24 secs if done serially (5.42 secs if done in parallel) and accuracy ~85%. The *bot* can use the image-captcha breaker mentioned in Section 2.3 for solving the image part of the challenge with an average time of 19.2 seconds and accuracy 70-80%. Additional time overhead and accuracy concerns for understanding the semantics of the audio and correctly utilizing it to compute the associated image(s) response would need to be addressed by the *bot*.

6 Conclusion

Surveying existing captchas and attack methodologies employed to break them, we have proposed a hybrid captcha that would require a time overhead and additional effort in terms of accuracy/correctness for *attackers* at the same time keeping the challenge intuitive enough for human users. Future direction involves making it accessible for visually impaired users. One way could involve introducing alternative question types like - hover the mouse in a certain way or click a total of (number 1) + (number 2) times.

7 References

1. "The Official CAPTCHA Site." http://www.captcha.net/

2. "CAPTCHAs' Effect on Conversion Rates - Moz.", https://moz.com/blog/captchas-affect-on-conversion-rates

3. "I'm not a human: Breaking the Google reCAPTCHA - Black Hat." https://www.blackhat.com/docs/asia-16/materials/asia-16-Sivakorn-Im-Not-a-Human-Breaking-the-Google-reCAPTCHA-wp.pdf

4. "unCaptcha: A Low-Resource Defeat of reCaptcha's Audio Challenge." https://uncaptcha.cs.umd.edu/papers/uncaptcha_woot17.pdf

5. https://www.researchgate.net/profile/Yang-Wai_Chow/publication/262394775_Breaking_an_animated_CAPTCHA_scheme/links/575769dd08ae5c654904292b.pdf

6. "Text to speech – Python." https://pythonprogramminglanguage.com/text-to-speech/