Image recognition problems in Google reCAPTCHA v2 are fixed using deep learning.

Dr. V Venkata Ramana[1,] Dr. V Lokeswara Reddy[2] Dr K Sreenivasa Rao[3], M Ramanjeneya Reddy[4]

[1,2,3, 4]Professor, Department of CSE, K.S.R.M College of Engineering(A), Kadapa

Abstract—

The most widely used CAPTCHA solution nowadays is Google's reCAPTCHA v2, which provides primarily an image-based CAPTCHA challenge. This research investigates the safety features of reCAPTCHA v2's image tests and suggests a deep learning-based approach for quickly and easily bypassing them. Both a bespoke object recognition deep learning model and Google's own Cloud Vision API are used to evaluate the suggested solution, which uses human mouse mimicry to get around obstacles. Some countermeasures to improve reCAPTCHA v2's security are also proposed in the article, along with some new avenues of attack.

INTRODUCTION

Completely Automated Public Turing Tests to tell Computers and Humans Apart (CAPTCHA) are a common occurrence when browsing the internet these days, and they are a common security measure employed by websites to defend against bots and other forms of autonomous traffic. They are tests that are designed to identify potential autonomous traffic from real human users. The general goal when designing a captcha is that it should be easy for humans and hard for computers. Originally captchas were text-based and had users identify alphanumeric characters, but these have been surpassed in popularity by image-based ones. With image-based

captchas, the user usually must perform some operation on an image or set of images, such as identifying an object. The biggest reason for the switch is the difference in complexity. For human users there is not a significant difference in difficulty between identifying images and identifying text, but for computers there is a significant uptick in complexity. This resulted in many captcha services to start migrating from using text to images, which calls into question the overall security of image-based captchas. The most popular service in use today is Google's captcha service reCAPTCHA v2 which mainly uses image-based challenges, and the service reportedly serves up millions of captcha challenges every day [1]. reCAPTCHA v2 has two main steps. Firstly, the user must click on a checkbox to initiate the service. At this time, Google performs a risk analysis on the user include behavior of user on current webpage, cookies and browsing history on Google services, etc. A higher score represents the likelihood that the user is a human and if the returned score meets a certain threshold, the user receives no captcha challenge and is automatically passed through, called noCAPTCHA. If the score didn't meet the threshold, the user is presented with an image-based captcha challenge, with an audio-based captcha challenge available as an alternative. Once the user successfully completes the challenge they are verified.

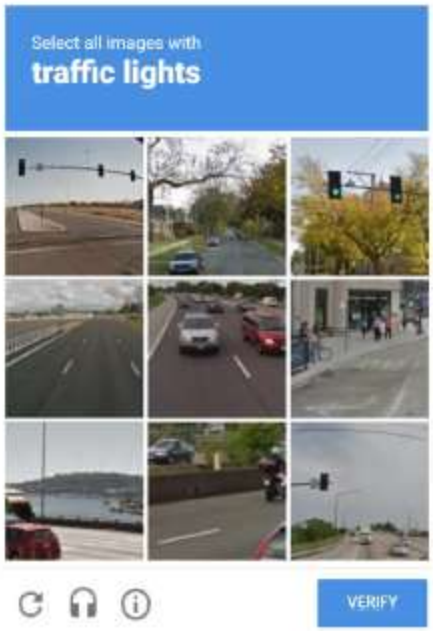

Fig. 1. Example of a captcha image challenge from reCAPTCHA v2

This paper will investigate the security of reCAPTCHA v2, focusing on its image-based challenges. It will identify security measures used on the given images as well as the interactions with the challenge elements. A method is proposed to automatically solve reCAPTCHA v2's image-based challenges using deep learning. Two deep learning methods will be implemented with this solution, a custom trained object-detector and an off-the-shelf object-detection API. This method is then evaluated by testing it on the live reCAPTCHA v2 service. Afterwards, the paper will identify possible additional measures that could be added to increase the security of the image

Fig. 2. The 3 different types of reCAPTCHA image challenges

Challenges and it will also suggest additional attack directions for bypassing reCAPTCHA v2.

BACKGROUND

For the proposed reCAPTCHA solver, the custom deep learning solution utilizes You-only-look-once version 3 (YOLOv3) object-detection, and the API used is Google's own Cloud Vision API. YOLOv3 is a convolutional neural network (CNN) based object-detection method that maintains accuracy while significantly increasing speed over other CNN based architectures like single-shot multibox detection or region based CNNs [2]. Google's Cloud Vision offers a high performing pre-trained machine learning model for object detection through an easy to use REST API [3]. For reCAPTCHA v2's image challenges, the user is given a text and sometimes picture clue and must identify all images containing said clue from a set of given candidates. It is important to note that the user doesn't have to necessarily achieve 100% accuracy, as it is possible to miss some images and still pass the captcha challenge. The image challenges can be further broken down into 3 distinct types which should be taken into consideration. *A. Type 1 – 3 x 3* The first and most common type will be called "3 x 3" or "3 by 3". In this type, the challenge gives the user a 3 by 3 grid of square and distinct images. Additionally, in this subtype after selecting a correct image it may be replaced by a new image that must also be considered. The user will be asked to continue Selecting all images with the clue until none remain. This type uses a set of about twelve possible clues.

B. Type 2 – 4 x 4

The second type will be called "4 x 4" or "4 by 4". In this arrangement, the user is given 1 single square image which is divided into a 4 by 4 grid of smaller sections. From testing it seems to share the same clue set as type 1. Unlike type 1 however, type 2 will not have new images be phased in after a successful selection.

C. Type 3 – 4 x 2

The last and least common type is "4 x 2" or "4 by 2". With this version, the user is given a 4 x 2 grid of rectangular and distinct images. This type generally has its own set of clues that isn't used in the other two types. Three potential clues were identified for type 3. Also, like type 1, new images might be phased in to replace correctly clicked images. Types 1 and 3 can be further broken down into two additional configurations, one configuration being a static set of images, and the other being the aforementioned new images replacing old images. Once the user has successfully completed a challenge and submitted it, they may be given a totally new additional challenge which they must also complete. This new challenge can be of different type from the previous challenge. Whether or not the user is given additional challenges seems to be based upon reCAPTCHA's risk detection, giving higher risk users more challenges, though most users are verified after only 1 or 2 challenges. This risk analysis score seems to also affect the images themselves, with more noise being added for users who are considered riskier

Fig. 3. Example of a reCAPTCHA image with and without noise

RELATED WORKS

While there are numerous works researching text-based catches, works focusing image-based catches are much more limited, and there are even fewer focused on creating an attack to solve real image catch services. For older revisions of reCAPTCHA, several papers exist such as [4] but the service has changed significantly since then. For the most recent work on reCAPTCHA v2, there is a 2018 paper that created an attack against its image challenges, also utilizing a form of deep learning [5]. Our proposed solution's testing results cannot be directly compared to this other work however, due the differing testing conditions and large updates made the reCAPTCHA v2 since then. This is because reCAPTCHA v2 is a live service and has changed quite a bit in even in the one-year gap, so the Performance of [5] will have changed if run today, and its source code isn't available for updated testing. In terms of results, [5] concluded that image-based captchas were quite vulnerable to potential attack and required additional security measures or alternatives.

PROPOSED SOLUTION

The proposed solution is to utilize human mimicking mouse movement in combination with deep learning object-detection to complete reCAPTCHA's image challenges. The mouse movements are used to try and initially receive a better risk analysis score before using the object-detection to detect the given clue in the image challenges.

A. Human Mouse Movements

The humanlike mouse movement is achieved by moving the mouse along a Bezier curve calculated with randomly selected points on the container webpage. ReCAPTCHA can see this movement data when doing its risk analysis, and the intent with this stage is to receive a better risk score. While this will not be enough to trigger the automatic noCAPTCHA verification without additional steps, it should be enough to receive a slightly better score than without any initial movement at all. This better score will result in the following image challenge having less Noise, which should make it easier for our object-detection method to classify images.

B. Image Challenge

Once the image challenge is received by the client after the initial risk analysis, the first step is to extract the text clue. This is stored in plaintext, so it is trivially extractable. Secondly, the solution extracts the given image set. The images are located and screen shorted by the solution on the webpage using the initial location of the reCAPTCHA widget as a starting point. Google dynamically generates the given set of images as one unique image file for each challenge request, so it needs to be cropped and spliced before being fed into the object-detector.

a) Type 1 and Type 3: For type 1 and type 3 challenges, the 9 and 8 distinct images respectively are cropped and stored individually. These images are then processed by the object detection method according to the clue, and any images that contain the clue label are selected in reCAPTCHA. After a click, the HTML code is scanned to detect if a new image is going to be phased in to replace the selected image. If the HTML element containing the image doesn't change after a click, then the set of images is static so the solution simply selects all labeled images and submits. If the HTML element does change, then that means new images will be added in that must also be Labeled through the object detector. In this case the solution will continue until no newly added images contain the clue label, and then submit afterwards.

b) Type 2: For type 2 challenges, the original image is spliced together from the given generated image, effectively removing the gridlines. This complete image is then sent through the object-detector to receive bounding boxes with labels. The solution then breaks the image back into the 4 by 4 sections, and any sections containing part of a bounding box for the clue label is selected. This type of challenge always features a static set of images, so once the indicated sections are clicked it will be submitted.

Fig. 4. An image pre- and post- YOLO object-detection

After submission of a challenge, the proposed solution will wait and see if reCAPTCHA verifies or instead sends another challenge. If it verifies, then the solution is done, otherwise the solution will repeat the solving process with the new challenge until it does receive verification.

C. Object-Detection

Two different custom YOLOv3 models will be created for testing, created using two different training sets. One data set was obtained from Image Net [6], supplemented by Google image searches for any clue categories not available on Image Net. The second training set will be composed of images taken directly from Google reCAPTCHA v2 challenges. For each clue category and image source, 300 images were used for the training set, and an additional 200 for a testing set. Transfer learning was used, with the initial pre-trained weights coming

From a YOLOv3 model trained on the Open Images data set by Google. This solution's YOLO-Image Net model achieved an average training accuracy of ~96.8% with 3-fold cross validation and took about 12 hours to train on the entire training set. The YOLO-Google model achieved ~96.3% and similarly took around 12 hours to train. The training was done on a machine with an AMD Ryzen 7 1700x CPU, an Nvidia GTX 1080 Ti GPU, and 16 GB of RAM. On their respective testing sets, the YOLO-Image Net model achieved an accuracy of ~93.8% and the YOLO-Google model achieved ~95.6%. For reference, the Cloud Vision API achieved an accuracy of ~96.7% and ~97.3% on those same respective testing sets.

EVALUATION

To evaluate the proposed solution, it will be run on Google's reCAPTCHA v2 demo site 200 times each for both YOLO setups and Cloud Vision. Additionally, it will also be tested with and without the initial human mimicking mouse movement to evaluate its effectiveness. A run will be considered a success if it receives verification from Google and a failure anytime it does not. A VPN was used between each run to switch IPs in order to avoid Google flagging the test site's actual IP as suspicious, which would result in throttled service requests and increased risk scores.

TABLE I. OVERALL SUCCESS RATE OF PROPOSED SOLUTION

Object-Detection Solution			Success Rate
YOLOv3	ImageNet	w/ movement	61%
		w/o movement	38%
	Google	w/ movement	68%
		w/o movement	41%
Cloud Vision		w/ movement	73%
		w/o movement	42%

As the results show in Table I, adding the initial mouse movements greatly increases the success rate, a 23% and 27% increase for YOLO and 31% increase for Cloud Vision, most likely due to the decreased chance to encounter noise in the given images. Noise is highly detrimental to the performance of object-detection methods, so minimizing any added noise was a key addition to the solution. As for the performance with the mouse movements, both YOLO models and Cloud Vision achieved greater than 60% success rate, with YOLO at 61% and 68%, and Cloud Vision at 73%

TABLE II. SUCCESS RATE FOR EACH TYPE OF CHALLENGE

Object-Detection Solution			Success Rate		
			Type 1	Type 2	Type 3
YOLOv3	ImageNet	w/ movement	47%	89%	24%
		w/o movement	16%	90%	26%
	Google	w/ movement	61%	90%	33%
		w/o movement	21%	91%	34%
Cloud Vision		w/ movement	67%	91%	31%
		w/o movement	20%	88%	30%

TABLE III. BREAKDOWN OF CHALLENGE TYPES ENCOUNTERED FOR EACH SOLUTION'S 200 RUNS

Object-Detection Solution			Number of Challenges			
			Type 1	Type 2	Type 3	Total
YOLO v3	*Image Net*	*w/ move-ment*	152 *54%*	92 *33%*	38 *13%*	282 *100%*
		w/o move-ment	280 *74%*	61 *16%*	40 *10%*	381 *100%*
	Google	*w/ move-ment*	150 52%	102 35%	38 13%	290 100%
		w/o move-ment	284 75%	64 17%	30 8%	378 100%
Cloud Vision		*w/ move-ment*	141 *51%*	97 *35%*	39 *14%*	277 *100%*
		w/o move-ment	298 *75%*	71 *18%*	28 *7%*	397 *100%*

For type 2 challenges, the high success rate could be attributed to the nature of the image compared to the other types. Type 2 is a singular larger image, as opposed to the many smaller images of the other types. Larger images are easier for object-detectors to work on, and less overall images means less complexity. For type 3, the low success rate is most likely attributed to the composition of the image. The images given for type 3 are generally more obscured and more tightly cropped, meaning less contextual information for models to use.

SUGGESTIONS FOR IMPROVING RECAPTCHA V2

From the results of the experiments, as well as personal use, a couple of possible improvements to reCAPTCHA's overall performance as a security tool have presented themselves.

A. Store Clue Differently

Currently, reCAPTCHA v2 stores its clues as plaintext, which is perhaps the least secure way for it to be stored. We recommend storing and presenting the clue in another fashion, such as an image of the text, or perhaps by displaying the clue itself using only a representative image. This change would not greatly impact human usability, but it would add an extra layer of complexity that automated systems would have to handle.

B. Remove Type 2 Challenges

It seems that the difficulty of type 2 challenges is lacking in comparison to type 1 and 3, with the proposed solution achieving about ~90% success rates. It might be better to remove them outright and focus on type 1 and type 3 challenges, as the solution struggled much more with them.

C. Image Transformations

As shown by the decrease in performance when noise was involved, such transformations upon an image cause issues for deep learning object-detectors. Perhaps additional image transformations could also be implemented to help throw off these machine learners, such as slight rotations or color shifts. Again, these types of modifications increase the difficulty for machines considerably more than it does for humans.

D. Context-Sensitive Challenges

Instead of having the user just perform identification on an image, maybe have the user do some task that also involves deciphering the context of the image. For example, instead of asking the user to determine whether or not a picture contains a fruit, instead ask the user to count the number of fruits in the picture. Another example is instead of asking the user to identify whether a photo has a person in it, ask the user to determine whether the person in the image is smiling or frowning. Context-sensitive challenges like these would require redesigning most current automated systems to adapt for each new variation, as they would need to do more than just image classification/detection. However, for humans these come naturally and should not be a large step up in complexity.

CONCLUSION

In conclusion, the suggested approach has a high success rate in evading the picture challenges used by Google reCAPTCHA v2. This method is reliable, since it has been tested using both a custom-trained object-detection deep learning model (68% bypass rate) and Google's own Cloud Vision API (73% bypass rate). The security of reCAPTCHA was also examined, along with possible ways to make it more secure and future research into reCAPTCHA bypass techniques.

REFERENCES

[1]reCAPTCHA,https://developers.google.com/recaptcha/

[2] Redmon, J., & Farhadi, A. (2018). "YOLOv3: An Incremental Improvement", https://pjreddie.com/media/files/papers/YOLOv3.pdf

[3] Google Cloud Vision, https://cloud.google.com/vision/

[4] S. Sivakorn, J. Polakis, and A. D. Keromytis, "I'm not a human: Breaking the google recaptcha," Black Hat,(i), pp. 1--12, 2016.

[5]BinbinZhao,HaiqinWeng,ShoulingJi,JianhaiChen,TingWang, Qinming He, Raheem Beyah. 2018. Towards Evaluating the Security of Real- World Deployed Image CAPTCHAs. In AISec '18: 11th ACM Workshop on Artificial Intelligence and Security Oct. 19, 2018, Toronto, ON, Canada. ACM, NY, NY, USA, 12 pages. https://doi.org/10.1145/3270101.3270104

[6] ImageNet, http://www.image-net.org/