

I Am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs

Suphannee Sivakorn, Iasonas Polakis and Angelos D. Keromytis

Department of Computer Science

Columbia University, New York, USA

{suphannee, polakis, angelos}@cs.columbia.edu

Abstract—Since their inception, captchas have been widely used for preventing fraudsters from performing illicit actions. Nevertheless, economic incentives have resulted in an arms race, where fraudsters develop automated solvers and, in turn, captcha services tweak their design to break the solvers. Recent work, however, presented a generic attack that can be applied to any text-based captcha scheme. Fittingly, Google recently unveiled the latest version of reCaptcha. The goal of their new system is twofold; to minimize the effort for legitimate users, while requiring tasks that are more challenging to computers than text recognition. ReCaptcha is driven by an “advanced risk analysis system” that evaluates requests and selects the difficulty of the captcha that will be returned. Users may be required to click in a checkbox, or solve a challenge by identifying images with similar content.

In this paper, we conduct a comprehensive study of reCaptcha, and explore how the risk analysis process is influenced by each aspect of the request. Through extensive experimentation, we identify flaws that allow adversaries to effortlessly influence the risk analysis, bypass restrictions, and deploy large-scale attacks. Subsequently, we design a novel low-cost attack that leverages deep learning technologies for the semantic annotation of images. Our system is extremely effective, automatically solving 70.78% of the image reCaptcha challenges, while requiring only 19 seconds per challenge. We also apply our attack to the Facebook image captcha and achieve an accuracy of 83.5%. Based on our experimental findings, we propose a series of safeguards and modifications for impacting the scalability and accuracy of our attacks. Overall, while our study focuses on reCaptcha, our findings have wide implications; as the semantic information conveyed via images is increasingly within the realm of automated reasoning, the future of captchas relies on the exploration of novel directions.

1. Introduction

The widespread use of automated bots for increasing the scale of nefarious online activities, has rendered the use of captchas¹ [1] a necessity. Captchas are a valuable defense mechanism in the fight against fraudsters and are used for preventing, among other, the mass creation of

accounts and posting of messages in popular services. According to reports, users solve over 200 million reCaptcha challenges every day [2]. However, it is widely accepted that users consider captchas to be a nuisance, and may require multiple attempts before passing a challenge. Even simple challenges deter a significant amount of users from visiting a website [3], [4]. Thus, it is imperative to render the process of solving a captcha challenge as effortless as possible for legitimate users, while remaining robust against automated solvers. Traditionally, automated solvers have been considered less lucrative than human solvers for underground markets [5], due to the quick response times exhibited by captcha services in tweaking their design. These design changes can render a solver ineffective, as each solver must be crafted for a specific type of challenge. However, the ever-advancing capabilities of computer vision continue to diminish the security that can be offered by text-based captchas. Recent work demonstrated the effectiveness of a generic solver that can be applied to new captcha schemes without requiring a custom-tailored approach [6]. As such, the future of text-based captchas seems uncertain, and automated solvers may soon become dominant in underground captcha-solving economies.

The “no captcha reCaptcha” deployed by Google, aims to tackle the aforementioned challenges. The motivation behind the new version is to verify users, when possible, without requiring them to actually solve a tedious challenge. An advanced risk analysis system analyzes various aspects of a user’s request for a captcha (e.g., browser characteristics, tracking cookie) and calculates a confidence score. The score reflects the confidence of the system that the request originates from an honest user and is not suspicious (e.g., bot or human-solver). For high confidence scores, the user is only required to click within a *checkbox*. For lower scores, the user may be presented with a new image-based challenge or a traditional text-based captcha. In the image-based scheme, users are required to distinguish between a set of images and select those with similar content.

In this paper, we conduct an extensive exploration of the design and implementation aspects of reCaptcha. We find that, apart from the risk analysis system, the reCaptcha widget also performs a series of browser checks for detecting the use of web automation frameworks or discrepancies in the browser’s behavior. The checks range from verifying the format of browser attributes to more complex techniques like

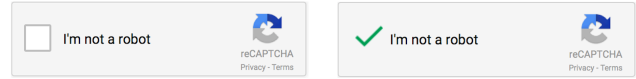
1. For readability, we follow the convention of the lowercased acronym.

canvas fingerprinting [7]. Nonetheless, we build a system that leverages a popular web automation framework and still passes the checks. Furthermore, following our black-box testing, we identify design flaws that allow an adversary to trivially “influence” the risk analysis process and receive the *checkbox* challenge. Specifically, we find that supplying a Google tracking cookie that is 9 days old is sufficient, even if it has not been associated with any browsing history (apart from the initial request that created it). Moreover, since tracking cookies have not been previously used by attackers, no safeguards exist for preventing their creation at a large scale; we create more than 63K cookies a day from a single host without triggering any defenses. Using these cookies, our system can maintain a solving rate of 52K-60K checkbox captchas per day, from a single IP address.

Next, we design a novel attack for solving image-based captchas. To our knowledge, this is the first captcha-breaking attack that extracts *semantic information* from images for solving the challenge; with the use of image annotation services and libraries, we are able to identify the content of images and select those depicting similar objects. We also take advantage of Google’s reverse image search functionality for enriching our information about the images. We further leverage machine learning for our image selection, and develop a classifier that processes the output of the image annotation systems and searches for subsets of common tags that occur across images with similar content. Our automated captcha-breaking system is highly effective, achieving an accuracy of 70.78% against the image-based reCaptcha. It is also efficient, as it solves challenges in 19 seconds. To demonstrate the general applicability of our attack, we also target Facebook’s new image captcha, where we achieve an accuracy of 83.5%.

Due to the prevalence of human solving services, the utility of any captcha-breaking system must also be evaluated in terms of cost-effectiveness. As such, we evaluate our system in an *offline mode*, where no online information or service is used. Under such restrictions, and running on commodity hardware, our attack solves 41.57% of the captchas while requiring only 20.9 seconds per challenge, with practically *no cost*. We employ a popular captcha-solving service, and find that our system is comparable in both accuracy and efficiency, rendering it an effective cost-free solution for fraudsters.

Overall, while our study focuses on reCaptcha, our findings are generalizable and with significant implications for future directions. Despite their current flaws, incorporating the safeguards we identify in reCaptcha’s risk analysis and widget can improve the security of future captcha implementations. Furthermore, evolving from the recognition of distorted alphanumeric characters to more advanced tasks, such as extracting semantic information from images, has been considered a promising direction for the design of robust and usable captchas [8]. Our attack, however, raises questions about the suitability of such tasks, given the recent advancements in computer vision and machine learning. We believe the future of captchas depends on the exploration of fundamentally different approaches to their design.



(a) Before user clicks checkbox. (b) User considered human.

Figure 1. The reCaptcha widget.

The major contributions of this paper are the following:

- We conduct a comprehensive study of the security aspects of the most widely used captcha service. Our extensive testing reveals flaws in Google’s advanced risk analysis system, which can be exploited by adversaries for deploying large-scale automated attacks. Our novel misuse of tracking cookies renders them a valuable commodity for adversaries.
- We design a novel attack that leverages deep learning systems for extracting the semantic information of images. We extensively evaluate our attack against reCaptcha, quantify the effect of different services and types of information on the accuracy, and demonstrate that it is both highly *accurate* and *efficient*. We also demonstrate that it is *generalizable*, as it is effective even against Facebook’s image captcha, and *practical*, as it achieves comparable results to captcha-solving services at virtually no cost.
- Based on the insights from our empirical analysis, and key aspects of our approach, we introduce new safeguards for preventing the manipulation of the risk analysis process at a large scale. We also present guidelines for mitigating attacks against the image reCaptcha. The disclosure of our findings to Google led to the modification of the risk analysis and image reCaptcha, resulting in a more robust captcha service.

2. Analyzing Recaptcha

The reCaptcha service [9] offered by Google, is the most widely used captcha service, and has been adopted by a plethora of popular websites for preventing automated bots from conducting nefarious activities. A recent announcement [10] reported the deployment of a new reCaptcha mechanism designed to be more user-friendly and secure. By leveraging information about users’ activities, acquired through persistent tracking cookies, Google can correlate requests to users that have previously interacted with any of its services. This is possible even in cases where the user is not currently logged into a Google account or the browser is in a private/incognito mode, as the (anonymous) tracking cookie can still reveal a user’s previous activities. This is a novel direction, as it allows Google to further strengthen the captcha service by coupling it to their vast pool of user information, allowing them to better detect suspicious requests. While safeguards have been previously proposed for mitigating automated large-scale captcha solving [3], reCaptcha goes beyond the stateless, per IP address, approach and employs a plethora of checks.

Widget. When visiting a webpage protected by reCaptcha, a widget is displayed (shown in Figure 1(a)). The widget’s JavaScript code is obfuscated, to prevent analysis from third parties. When the widget loads, it collects information about the user’s browser which will be sent back to the server. Furthermore, it performs a series of checks for verifying the user’s browser, and also checks for known browser automation kits. We provide more details on the checks in Section 4.

Workflow. Once the user clicks in the checkbox, a request is sent to Google containing (i) the Referrer, (ii) the website’s `sitekey` (obtained when registering for reCaptcha), (iii) the cookie for `google.com`, and (iv) the information generated by the widget’s browser checks (encrypted). If the user is logged into her Google account, the request will also contain an extra field with the user’s authentication cookie. The request is then analyzed by the *advanced risk analysis* system, which decides what type of captcha challenge will be presented to the user. The response is an HTML frame that contains the challenge, which is displayed by the widget in a popup.

Solution. Once the challenge has been presented to the user, it has to be answered within 55 seconds. Otherwise, the popup is closed and the user is required to click on the checkbox again to receive a new challenge. Once the user clicks, an HTML field called `recaptcha-token` is populated with a token. If the user is deemed legitimate and not required to solve a challenge, the token becomes valid on Google’s side. Otherwise, the token will remain invalid until the user solves the given challenge correctly. The token is submitted to the website when completing the desired action (e.g., POST an account creation form), and the user’s session expires after 2 minutes. The token is invalidated on Google’s side 2 minutes and 10 seconds after its creation. The website sends a verification request through the reCaptcha API which contains: (i) a shared secret, (ii) the response token and, optionally, (iii) the user’s IP address. The response is a JSON object with a boolean field indicating if the verification was a success. If the verification fails, error codes offer more information.

Based on the level of confidence assigned to the specific request, Google’s advanced risk analysis system will select which type of challenge to present to the user. The different versions present a varying level of difficulty and nuisance, as some are trivial to pass while others are problematic even for humans. If a specific user requests multiple challenges or provides several wrong answers in a short amount of time, the system will return increasingly harder challenges.

Challenge type. In our experiments we came across the following versions of reCaptcha:

“*No captcha reCaptcha*” [Figure 1]. This new user-friendly version is designed to completely remove the nuisance of solving captchas. Upon clicking the checkbox in the widget, if the advanced risk analysis system has high confidence that the request is “legitimate”, the checkbox changes to a tick, the challenge is considered solved, and no further action is required. For the remainder of the paper, we will refer to this version as the *checkbox* captcha.

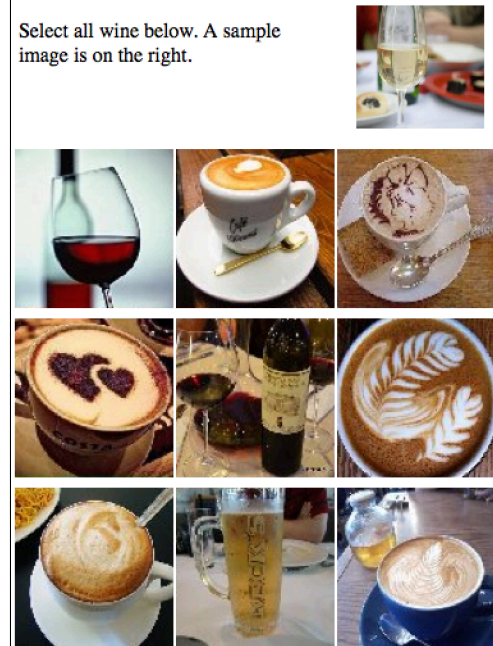


Figure 2. Similar images challenge by reCaptcha.

TABLE 1. EXAMPLES OF REMAINING VERSIONS OF RECAPTCHA.

(a)	<i>sedbj</i>	(b)	2126
(c)	special <i>sterid</i>	(d)	<i>WILLOR</i> <i>WILCOCKANT</i>
(e)	<i>thouch</i> <i>twof</i>		

Image reCaptcha [Figure 2]. This new version is built on the notion that identifying images with similar content is a difficult task for bots. The challenge contains a sample image and 9 candidate images, and the user is requested to select those that are similar to the sample.

Distorted one-word [Table 1-(a)]. This is the easiest version of a distorted text reCaptcha.

Street view numbers [Table 1-(b)]. The answers provided by users make Google Maps more precise and complete.

Scanned words [Table 1-(c)]. In an effort to improve the digitization of books, the challenge includes at least one word that was scanned from a book, but has not been recognized by the OCR software with high confidence.

Distorted two-word [Table 1-(d)]. This is returned when the request is considered suspicious.

Fallback captcha [Table 1-(e)]. If the User-Agent fails certain browser checks, the widget automatically fetches and presents a challenge of this type, before the checkbox is clicked. We explore when it is triggered in Section 4.

Type selection. While the new reCaptcha relies on the image challenges for security, text captchas have not been completely removed from the system. Specifically, upon

initial release of the new reCaptcha, the easy text captchas were returned intermittently with the new image captcha. However, over the period of the following 6 months, text captchas appeared to be gradually “phased out”, with the image captcha now being the default type returned. Nonetheless, the difficult text captchas (Table 1-(d), Table 1-(e)) are still in use; they are returned after multiple solutions of captchas or when certain browser checks fail. This suggests that the text captchas target suspicious human users (e.g., workers for captcha-solving services) and not bots, as these captchas are harder for humans to solve despite being solvable by bots [6], [11].

Threat model. In practice, fraudsters may follow two distinct approaches for solving challenges. First, they may employ an automated captcha breaking system, which will allow them to conduct nefarious actions unencumbered (e.g., create email accounts, post in forums). Second, they may employ humans to manually solve challenges, i.e., through an underground captcha-solving service [5]. In this paper, we develop methods for automatically solving the challenges. Our goal is to design methods for bypassing safeguards and influencing the advanced risk analysis into returning checkbox captchas, and develop an attack that can successfully solve semantic image captchas. Nevertheless, our findings could also be exploited by solving services, as we demonstrate the feasibility of accurate, large-scale, low-cost attacks.

3. System Overview

In this section we present an overview of our system designed to solve reCaptcha challenges. Due to the approach taken by Google, where part of the functionality is offloaded to the advanced risk analysis system (i.e., determining the difficulty of the challenge), our captcha-breaking system exhibits a novel aspect that has not been previously required by similar attacks; we build a component designed to “influence” the process that determines the level of difficulty for the challenge.

Our system is built on Selenium, an open-source browser automation framework. We opt for the Firefox-WebDriver, so we can leverage the rich functionality of the browser engine and handle all aspects of the webpages required for bypassing the browser checks of reCaptcha. Specifically, we build on top of Mozilla Firefox (v.36). The WebDriver offers functionality for locating specific HTML DOM elements in a page, provides features for executing JavaScript and controllers for handling keyboard and mouse events. We use the XVFB virtual framebuffer for configuring the display screen resolution. We also manage the saving and loading of browser cookies, as the persistent Google tracking cookie is an integral part of the advanced risk analysis process.

The system consists of two main components. The first is responsible for creating tracking cookies that can influence the risk analysis process. Our second component processes the challenges and, depending on the type of challenge returned, follows different techniques for solving them.

Cookie Manager. The Google tracking cookie plays a crucial role in determining the difficulty of the challenge that is presented to the user. Furthermore, each cookie can receive up to 8 checkbox captchas in a day. As part of our attack, we develop functionality for automatically creating Google cookies. The goal is to create cookies which are subsequently “trained” to appear as originating from legitimate users and not automated bots. In each case, we create a cookie in a clean virtual machine, where our browser automation system imitates a user browsing the web. We configure the system to perform actions specific to the website being visited, while mimicking a diurnal cycle and following random resting intervals between actions. For example, we conduct Google searches for certain terms and follow links from the results, open videos in Youtube, and perform searches in Google Maps. We also visit popular websites that contain social plugins associated to Google. As such, we are able to create Google tracking cookies and associate them with varying amounts of browsing activity.

Recaptcha Breaker. This component is designed to collect reCaptcha challenges. Using the cookies created by the previous module, it visits sites that employ reCaptcha for deterring automated bots from completing certain actions. If the image captcha is presented, it is passed to a different module. We have gathered a list of websites that rely on the reCaptcha service for preventing automated actions. During our experiments, we do not conduct any actions in these websites; we only target the reCaptcha challenges. Upon locating such websites, we manually inspected the website’s HTML page code and located the appropriate DOM element that holds the reCaptcha widget `IFrame`. We recorded the particular element and used it during our attack for identifying the frame containing the captcha. As the widget requires the user to click the checkbox to receive a captcha, our system locates the checkbox element through its identifier (`recaptcha-anchor`) and performs a mouse click action in it. If the advanced risk analysis considers the request legitimate and no challenge is returned, we only need to extract the `recaptcha-token`.

If the user is required to solve a challenge, a popup is created on the client page by the JavaScript, within an element with a class name of `goog-bubble-content`. Inside the popup there is an `IFrame` where all captcha challenge elements are located, along with the verification button for sending the response. To identify which type of captcha is shown to the user, our system attempts to locate `rc-imageselect` which is the element created for the image captcha, or `rc-defaultchallenge-response-field` which is the element created for holding the user response for text captchas. Based on which element appears in the code, we identify the type of challenge that was presented.

The text captcha is sent from the Google server and put inside the `rc-defaultchallenge-payload` element in JPG format. For the image captcha, the text description (which contains the *hint*) is located in `rc-imageselect-desc`. The sample and candidate images are put in `rc-imageselect-tile`. All images

are sent in a `base64` format and rendered in the user browser. Our system extracts the description and the `base64` images, which are saved in PNG format. Images are saved individually, so they can be supplied to the image annotation modules. The system also collects the `rc-imageselect-tile` and the verification button `recaptcha-verify-button`, for sending a mouse action to those HTML objects when our back-end breaker finishes processing the challenge and is ready to submit the solution.

3.1. Breaking the image captcha

The ability to discern objects and provide detailed descriptions of images is a cognitive process that comes naturally to humans. On the other hand, processing an image for identifying objects and assigning semantic information to it, is considered a complex computer vision problem [12]. Therefore, such tasks have been considered a promising alternative to text-based captchas, and several schemes have been proposed, in which users are asked to identify or label images [3], [13]. However, recent advancements in the area of computer vision have demonstrated impressive results [12], [14]–[16], and image annotation services that leverage such techniques have emerged. As these services are bound to become even more widely available, we explore if they can be used for solving image captchas.

To solve an image captcha, our system has to automatically identify which of the given images are semantically similar to the sample image. Upon receiving a challenge we extract the sample and candidate images, and the *hint* that describes the content of the sample image (e.g., “wine”). Next, all images are passed to an image annotation module.

GRIS. The Google Reverse Image Search, built on the work by Krizhevsky et al. [17], offers the ability to conduct a search-based on an image. If the search is successful it may return a “*best guess*” description of the image (which may differ for the same image across searches) along with a list of websites where the image is contained, and other available sizes of that image. While this is not part of Google’s public API, we identified the format of the search URL so our module can replicate the functionality. When conducting the reverse search for the 9 candidate images, we also collect the *page titles* of the webpages that contain the image, as an extra piece of information. If available, we also obtain a higher resolution version of each image, as it increases the accuracy of the image annotation modules.

During our initial experiments, we came across instances where the descriptions were not in English. As such, if a description is returned, we extract it and convert it to English through the automatic language detection feature of Google Translate. We also convert all tags to their singular form.

Image annotation. There are several free online services and libraries that offer relevant functionality, ranging from assigning tags (keywords) to providing free-form descriptions of images. Example outputs are shown in Table 2.

Clarifai is built on the deconvolutional networks by Zeiler et al. [18], and returns a set of 20 tags describing the

image along with a confidence score for each tag. The tags range from generic (“drink”) to very specific (“cabernet”).

*Alchemy*² is also built upon deep learning, and offers an API for image recognition. For each submitted image, the service returns a set of tags and a confidence score for each tag. In our experiments, images received up to 8 tags, which tend to be specific (e.g., “wine”).

TDL. Srivastava and Salakhutdinov [16] have released an app for demonstrating the image classification capabilities of their deep learning system. We have identified the API calls used by the app, and built a module that leverages the API for obtaining a description of the images. For each image, 8 tags are returned along with a confidence score.

NeuralTalk. Karpathy and Fei-Fei [12] developed NeuralTalk, a Recurrent Neural Network architecture for generating free-form descriptions of an image’s contents. We have developed a module for processing the images with the NeuralTalk library locally; we break the returned description down into individual words, and remove verbs, prepositions and conjunctions. The remaining words are considered the tags for the image.


Caffe. Jia et al. [19] have released Caffe, a deep learning framework, which we also leverage for processing images locally. Caffe returns a set of 10 labels; 5 with the highest confidence scores and 5 that are more specific as keywords but may have lower confidence scores.

Tag Classifier. The returned tags do not always exactly match the description (i.e., hint) given by reCaptcha for a challenge. To overcome this, we leverage machine learning to develop a classifier that can “guess” the content of an image based on a subset of the tags. Specifically, we opted for the Word2Vec word vectors proposed by Mikolov et al. [20] for finding the similarity between tags and hints. During the training of our classifier, we modeled and represented each word (tag and hint) as a real vector in vector space. Each tag assigned to an image is paired with the correct hint and all `<tag, hint>` pairs are given as input to the model. We tune the parameters and identify the optimal values for each annotation system. Once the classifier has been trained, it can be used to predict the similarity of the captcha’s hint and the tags by computing the cosine similarity between their corresponding word vectors, with the goal of identifying subsets of tags from each image that have been associated with the hint during the training phase. Thus, our classifier allows our system to select images with similar content even if the annotation system does not return tags that exactly match the given hint.

History Module. During our experiments we detect a non-negligible amount of repetition within the captchas, i.e., many images are actually re-used across challenges. As such, we manually create a labelled dataset with images and their tag from challenges we collect. Each image is annotated with the hint given in the challenge that describes the content (e.g., cat, soup). We also maintain a `hint_list` that contains the hints we have seen.

2. <http://www.alchemyapi.com>

TABLE 2. EXAMPLE OUTPUT FROM EACH IMAGE ANNOTATION MODULE, WITH TAGS SORTED IN DESCENDING ORDER OF CONFIDENCE.

	GRIS	Alchemy	Clarifai	TDL	NeuralTalk	Caffe
	wine and blood	wine, glass	glass, red wine, wine, merlot, liquid, bottle, still, glassware, alcohol, drink, wineglass, beverage, pouring, white wine, cabernet, taste, leaded glass, dining, party, vino	red wine, goblet, wine bottle, punching bag, beer glass, perfume, balloon	a glass of wine sitting on top of a table	red wine, wine, alcohol, drug of abuse, drug, red wine, punching bag, beaker, cocktail shaker, table lamp

Solution. Each module assigns the candidate images to one of 3 sets: *select*, *discard* or *undecided*. First, we collect information for all the images through GRIS. Next, if a hint is not provided, we search for the sample image in the labelled dataset to obtain one if possible. The history module searches for the candidate images in our labelled dataset and, if found, compares their tag to the hint and adds those that match to the *select* list. The remaining images are compared to the *hint_list* and added to the *discard* list if there is a match. An image annotation module then processes all the images and assigns them tags. If one of the tags matches the hint the image is added to the *select* set. If it matches one of the other tags in the *hint_list* it is added to the *discard* set. A similar process is conducted when we leverage the best guess and page title results for each image. Once all the modules have completed, the system processes the results and merges the sets from the modules. Each type of information is given a different “weight” (e.g., title pages have the lowest confidence) which allows us to overcome cases where modules assign the same image to a different set. If there is not an adequate number of images in the *select* set, the system picks the remaining images from the *undecided* set. That is done either through our tag classifier, or by selecting the images that have the most overlapping (i.e., common) tags with the sample image. In Section 4 we evaluate the effectiveness of these two approaches, and elaborate on why the optimal strategy is to select 3 images for the solution.

4. Attack Evaluation

In this section we evaluate our attacks against reCaptcha. First, we explore Google’s advanced risk analysis system, and identify how various characteristics of our system influence the confidence assigned by the system to our captcha requests. Next, we evaluate the accuracy of our attack against the image-based version of reCaptcha.

4.1. Influencing the risk analysis system

We follow a black-box testing approach to identify how different aspects of our system and testing environment influence the risk analysis process. Our goal is to issue requests for captchas that will be considered legitimate

TABLE 3. TRACKING COOKIE CREATION AND “TRAINING” BEHAVIOR.

Network	Web Surfing	Account	Threshold
Departmental	Frequent	No	9 th day
Departmental	Moderate	No	9 th day
ToR	Frequent	No	9 th day
ToR	Moderate	No	9 th day
Any	None	No	9th day

by the advanced risk analysis system and, thus, receive checkbox captchas that can be solved with a single click.

Browsing history. We aim to quantify the minimum amount of browsing history required for a specific cookie before it is presented with a checkbox challenge. When using a Google service, regardless of being logged in to a Google account or not, a plethora of relevant information is collected and associated to the cookies. As such, we deployed virtual users that exhibit different web surfing behavior. We explored multiple *network* connection setups, as Google may assign a higher level of trust to IP addresses originating from our university’s subnets. Thus, we also tunneled connections over ToR, which we configured to select exit nodes located in the USA. As shown in Table 3, regardless of our experimental setup, our system is presented with a checkbox captcha on the 9th day after the creation of the cookie. Specifically, we obtain a checkbox captcha after the beginning of the 9th day (00:01 PST) from the cookie’s creation. Our follow-up experiments revealed that the threshold remains the same even without conducting any web surfing with the cookie. Thus, *Google’s advanced risk analysis can practically be neutralized by simply appending a 9-day old cookie to the request.*

Account. We investigated if being logged in a Google account influences the risk analysis. First, we created fresh accounts but did not conduct a phone verification. We also created accounts and supplied them with an alternative email address from another provider. Our findings indicate that such accounts are considered suspicious and negatively influence the outcome. In both cases we were presented with a checkbox captcha after 60 days had passed. We repeated the experiment with a fresh account which we verified with a phone number from a US provider. Again, we were able to obtain a checkbox captcha after the 60th day, indicating that even phone-verified accounts start with a “bad reputation”. *Surprisingly, we conclude that it is actually better for an adversary to not use any account at all.*

TABLE 4. COMBINATIONS OF MISMATCHING INFORMATION BETWEEN WHAT OUR SYSTEM USES AND WHAT THE `USER-AGENT` CONTAINS.

Component	9-day Cookie	System runs	User-Agent reports	Captcha
<i>Browser</i>	✓	Firefox/36.0	{Mobile/8C148 Safari/6533.18.5, Chrome/42.0.2311.135 Safari/537.36}	image
<i>Browser version</i>	✓	Firefox/36.0	Firefox/{10.0, 35.0, 36.0, 3.0.12}	checkbox
<i>Browser version</i>	✗	Firefox/36.0	Firefox/{10.0, 35.0, 36.0}	image
<i>Browser version</i>	-	Firefox/36.0	Firefox/1.0.4	fallback
<i>Browser version</i>	✗	Chrome/42.0	Chrome/{15.0.861.0, 4.0.212.1}	image
<i>Browser version</i>	-	Chrome/42.0	Chrome/3.0.191.3	fallback
<i>Engine version</i>	-	Chrome/42.0; AppleWebKit/537.36	AppleWebKit/{528.10, 530.5, 531.3}	fallback
<i>Engine version</i>	✗	Chrome 42/0; AppleWebKit/537.36	AppleWebKit/{532 and up}	image
<i>Engine version</i>	✓ ✗	Firefox/36.0; Gecko/20100101	Gecko/20040914	image
<i>Browser/Engine</i>	-	Chrome 42/0; AppleWebKit/537.36	Chrome 42/0; Gecko/20100101	fallback
<i>Browser/Engine</i>	✓ ✗	Firefox/36.0; Gecko/20100101	Firefox/36.0; AppleWebKit/537.36	image
<i>Platform</i>	✓	Linux x86_64	{(Macintosh; Intel Mac OS X 10.8.); (Android; Mobile.); (Windows NT 6.3);}	checkbox
-	-	-	wrong format or incomplete information	fallback
-	✓	Linux x86_64; Firefox/36.0	Mozilla/5.0 (iPhone; U; CPU like Mac OS X; en) AppleWebKit/420 (KHTML, like Gecko) Version/3.0 Mobile/1A543a Safari/419.3	checkbox

Geo-location. We used ToR for exploring the impact of the user’s geo-location. We selected exit nodes across different countries and continents, including the top 3 countries associated with abused phone numbers used for Google account verification [21]. We find that that there is no restriction based on the country in which a cookie is created, as we are able to obtain the checkbox captcha regardless of the combination between the country where the cookie is created, and the current location of the user sending the request. This facilitates fraudsters since they can create cookies without any restrictions on the location of the IP addresses used.

Browser checks. The reCaptcha widget executes a series of checks for detecting suspicious browser attributes or behavior. While the widget’s JavaScript is obfuscated to prevent analysis, de-obfuscated code has been released³ providing indications about the type of checks conducted. Here we explore how aspects of our automated browser environment affects the outcome of the risk analysis. Due to space constraints we present a subset of our experiments.

Automation. According to the W3C specification⁴, Web-Driver is required to set a `webdriver` attribute to `True`, so websites can detect automation. While this is implemented by the current version of Firefox-WebDriver, and the widget checks for the attribute, we did not find it to have an effect. To further ascertain our finding, we changed our website’s JavaScript to explicitly set the attribute. Again the outcome remained the same, as we obtained the checkbox captcha.

Canvas fingerprinting. The way that content is rendered across machines and browsers varies, enabling device fingerprinting [7]. The reCaptcha widget leverages that for identifying the user’s browser. Specifically, the JavaScript code creates a Canvas element and draws a predefined composition. The `display` attribute is set to `None`, so the process remains invisible to the user. After the rendering is complete, the element is encoded in `base64` and sent back with the other data when the user clicks the checkbox.

The element can be compared to the outcome of known browser versions, for identifying the environment the widget is running in and detecting discrepancies with the reported `User-Agent`.

User-Agent. Table 4 presents a subset of the combinations we used in our experiments, to identify how the `User-Agent` influences the type of captcha that we receive. We have grouped certain variations that exhibit the same behavior. We repeat each combination multiple times at different moments, to verify the consistency of the outcome. We also conduct certain experiments with Chrome. Surprisingly, we found that if the `User-Agent` contains an outdated version of the browser or browser engine we are actually using, the widget automatically considers the environment suspicious and presents the user with a fallback captcha before the checkbox is clicked. The same happens if the browser and engine versions are up-to-date, but don’t correspond to the actual environment of the experiment (e.g., if we use Firefox but report Chrome). Fallback captchas are also returned when the `User-Agent` does not contain the complete information, or is mis-formatted. For other types of mismatching information, the widget usually returns the image captcha. Our findings also suggest that the widget does not detect the underlying operating system with the canvas fingerprinting, as we obtain checkbox captchas regardless of the platform stated in the `User-Agent`. The last row in Table 4 presents an interesting finding, as the browser engine we report (AppleWebKit) does not map to the engine we are actually using, yet we are still presented with the checkbox captcha. This behavior is unique to this outdated `User-Agent` from iOS1.0, and does not occur for newer versions. Finally, we also found that even if the `User-Agent` used during a cookie’s creation is different to the one used when requesting a captcha with that cookie, the outcome is not affected.

Screen resolution. We experimented with multiple combinations of screen resolutions, ranging from 1×1 to 4096×2160 pixels, and still obtained the checkbox captcha. Even when combining them with different

3. <https://github.com/neuroradiology/InsideReCaptcha>

4. <https://w3c.github.io/webdriver/webdriver-spec.html>

User-Agent options for both mobile and desktop devices, results remained the same.

Mouse. To identify whether the behavior of the mouse affects the outcome of the risk analysis, we experimented with various mouse behavior configurations. We explored aspects such as the timing of movements, erratic movement patterns, and issuing multiple clicks within the widget and checkbox. We also used the JavaScript `getElementById().click()` function to simulate a click within the checkbox without hovering over the widget. None of these had a negative effect on the risk analysis.

Cookie reputation. We explore if the server keeps information regarding previous behavior of a cookie, as a “reputation” score that affects the outcome of the risk analysis. We opt for two different types of suspicious behavior, using 9-day old cookies. In the first configuration, we use a User-Agent that automatically receives a fallback captcha. The second configuration uses a User-Agent that receives image or text captchas and always provides wrong solutions to the captchas. We run our experiments with varying request rates and for a varying number of days. In all cases, even after a week of consecutive wrong answers every one hour, once we change the User-Agent to a valid value the cookie receives a checkbox captcha. Our results suggest that, even though Google accounts have a reputation that affects the risk analysis outcome, that approach is not applied to cookies and they are not assigned a reputation.

Site restriction. The attack’s scale can be increased if we solve captchas on a website we control (`attacker.com`) but associate the tokens with a target website (`example.com`). This would facilitate captcha-solving services that harvest and sell tokens to others, as it will reduce the network activity and, thus, cost of the attacks. Furthermore, targeted websites may have stricter thresholds than the reCaptcha service, which would be preferable to avoid. Soon after the deployment of the new reCaptcha, a straightforward “clickjacking” attack was demonstrated [22]. To prevent the attack, reCaptcha was re-designed so that the token is tied to the website where the challenge was presented. Apart from checking the `Referrer`, the widget identifies the website through the `document.location.hostname`, which is `read-only` and cannot be intercepted for security reasons.

We present a workaround for bypassing this restriction. We setup a virtual host on our server and set the `ServerName` and other necessary fields to correspond with `example.com`. By using `a2ensite`, and modifying the `hosts` file, we can run our website on the `localhost` and trick reCaptcha into associating our request to `example.com`. To complete our attack we also need to send the target’s `sitekey`. This can be trivially obtained by visiting the website once and monitoring a reCaptcha request. Ultimately, when `example.com` verifies the token through the reCaptcha API, it will be successful.

Token harvesting. We also explored if creating a large number of cookies from a single IP address is prohibited. To avoid impacting other sites, we ran the attack against our own webserver. We maintain the same User-Agent

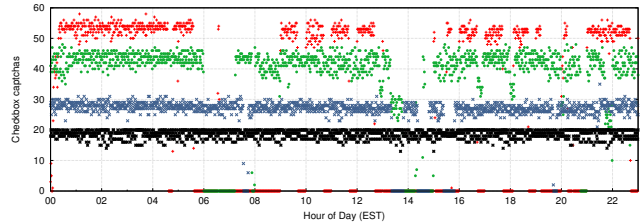


Figure 3. Checkbox captchas obtained per minute.

header for all requests, and deploy multiple instances of our system. Surprisingly, we are able to create over 63,000 cookies in a single day without triggering any mechanisms or getting blocked, and are only limited by the physical capabilities of the machine. This indicates that there is no mechanism to prohibit the creation of cookies from a single IP address. The only restriction we detected was triggered by a massive number of concurrent requests (i.e., for detecting DoS attacks). The lack of a safeguard can be justified by the fact that creating cookies at a large scale has not been required by attacks before. Indeed, we present a novel *misuse of tracking cookies*, which makes them a valuable commodity for fraudsters.

Next, we deployed our system to identify how many checkbox captchas we can solve in a single day. We experimented with different captcha-request rates. Figure 3 shows the results from representative experiments with different rates plotted in the attacker’s timezone (EST). Each experiment lasted 24 hours and is plotted with a different color. While the most aggressive rate (red points) gets blocked numerous times, if we maintain a lower rate of about 1,200 requests per hour (black points) we do not get blocked. For intermediate rates we get blocked at specific times that recur across experiments. Interestingly, these blocked periods coincide with a typical workday diurnal cycle (i.e., before work, lunchtime, after work). This can be attributed to the advanced risk analysis system either pro-actively adapting the threshold or “dropping” requests during peak hours, due to the increased traffic. Despite being blocked for short periods of time, at the optimal rate (green points) we receive approximately 2,500 checkbox captchas per hour, which drops to about 1,200 during peak hours. During weekdays, our results vary between 52,000 and 55,000. We observe less blocking during the weekend, and obtain over 59,000 checkbox captchas per day.

Overall evaluation. The current design and implementation of reCaptcha suffer from significant flaws and omissions that can be trivially exploited by adversaries. In an attempt to remove the burden for legitimate users, reCaptcha has enabled attacks that can effortlessly harvest tokens at a large scale and pass checkbox captchas, which do not require any computation. However, the plethora of checks that are performed, combined with those that are feasible (e.g., mouse checks) can be used to introduce more safeguards and improve the robustness of any captcha system.

TABLE 5. COMBINATIONS FOR PASSING THE IMAGE reCAPTCHA.

Image Selection	Constraint	Pass
n Correct + k Wrong	$k \leq 1$	✓
$(n - 1)$ Correct	$n > 2$	✓
$(n - 1)$ Correct + k Wrong	$k > 0$	✗

4.2. Breaking the image captcha

Solution flexibility. We explore whether reCaptcha has any flexibility when deciding if the given solution is correct. In any case, at least two images have to be selected before the response is sent. We manually solved image challenges using different combinations of the number of correct (n) and wrong (k) selections. In most cases (74%) we found the number of correct candidate images to be 2; the rest contain 3 and we also found two challenges with 4. As can be seen in Table 5, our experiments reveal that a user can pass the challenge even if a correct image is missed or a wrong selection is provided. Due to the small size of the images, in some cases their content may not be discernible even to humans. Based on these results, we set our captcha breaking system to select 3 images for the solution; this strategy offers us a “free” selection when $n = 2$ and may fall within the “relaxation” limits for the remaining challenges.

We also came across a few cases where we passed the challenge having supplied 2 correct and 2 wrong answers. This suggests that the image reCaptcha may contain both “control” and unknown images, similar to the text challenges [9]: apart from images with known content (i.e., control images), the challenge contains images for which the system has low confidence about their content. If a user selects the control images correctly, and also selects an unknown image, then the system can associate the unknown image with the hint. Thus, while in practice accepted combinations for passing a challenge may be even more flexible, Table 5 shows the passing combinations that we found to always hold true.

Image repetition. During our experiments we came across several cases of images being repeated across challenges. To quantify this behavior, we created a dataset of 700 downloaded challenges. First, we searched for images with identical MD5 values. Upon inspection, we came across a surprising finding. Out of the 700 captchas, we identified 6 pairs of completely identical challenges: for each pair, the images and their ordering were exactly the same in both challenges. In all 6 cases, the two challenges had been collected from a different website and always within two hours. Apart from the significant implications for the robustness of reCaptcha, it also suggests that challenges are not created “on-the-fly” but selected from a relatively small pool of challenges which is periodically updated.

We also found that images were being repeated across challenges. However, in most cases they had different MD5 values. Thus, we conducted a comparison using *perceptual hashes*, to identify identical images with different MD5 values. In all cases the images we detected seemed visually identical, despite being transformed for each new pool of

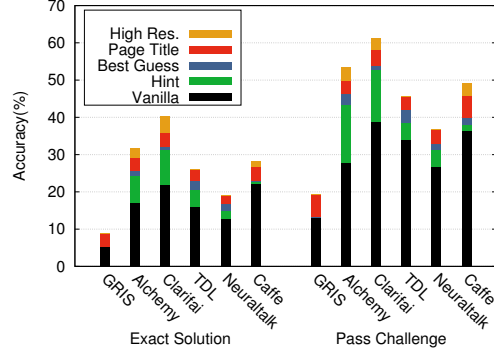


Figure 4. Accuracy of simulated attack for different combinations of modules and data against the image reCaptcha.

challenges; this may be done to prevent hash-based identification of the images. Since the hash value for identical images remains the same across websites and within the same pool of challenges, but changes in time, the transformation is independent of the website. We identified 1,368 redundant images that belonged to 358 sets of identical images. The largest set contained 92 images, i.e., the same image was shown as the sample in 92 different challenges. The most re-used candidate image was seen 12 times.

Attack simulation. To evaluate the effectiveness of each module, we simulated our attack on the dataset. To verify the accuracy of our attacks, we manually inspected the challenges and noted their solution. Figure 4 breaks down the accuracy for each module and type of information. Out of the 700 challenges, 667 contained a hint, and we acquired a best guess to use as a hint for 25 of the remaining challenges through a reverse search of the sample image. Here, we used a `hint_list` with the hints that we had come across in our experiments, but did not use the history module or tag classifier.

As we showed in Table 5, reCaptcha is flexible and a solution is considered correct even if it contains a wrong image along with the correct ones. Thus, to calculate the accuracy that our system would obtain against reCaptcha, we ran our simulated attack and accounted for that flexibility when deciding the outcome of each solution. As such, we configured the system to select 3 images for the solution so as to fall within the “relaxed” limits. The *Pass Challenge* bars represent the outcome of the attacks. We started with a baseline measurement for the “vanilla” version of each module which selects images based on the overlapping tags; for GRIS, this also entails using the hint provided in the challenge and the best guess returned by the reverse image search. GRIS passed 13.1% of the challenges. When using the page titles returned for the 9 candidate images as tags, the success rate increased to 19.2%. In general, the success rate for GRIS is limited by the number of candidate images for which we can obtain a best guess description. For the other modules, the baseline attack selects the 3 images that have the most common (overlapping) tags with the sample image. For Alchemy and Clarifai, the baseline success of

the attack was 27.9% and 38.9% respectively. When using the hint, best guess and page titles, the Alchemy module passed 49.9% of the challenges, while Clarifai passed 58%. Caffe is also very effective, solving 45.9% of the challenges. The hint has a significant effect in most cases, increasing the accuracy by 1.5-15.5% depending on the annotation system.

We explored how the attack’s accuracy is impacted by supplying the image annotation module with higher resolution versions of the images. We were able to automatically obtain a higher resolution version of 2,909 images from the 700 challenges. Out of those, 371 corresponded to the sample image. The high resolution images increased the attacks’ success, with Alchemy and Clarifai passing 53.4% and 61.2% of the challenges respectively. TDL is less accurate achieving 45%, while Caffe increases to 49.1%. While the higher resolution images could potentially improve the success rate of the GRIS module as well, conducting reverse image searches on all versions of each image, for acquiring a best guess description, would require a significantly higher number of queries for each challenge and is, thus, omitted. We also measured the number of challenges our system would pass if there was no flexibility. Since in most cases the solution consists of 2 images, we tuned the system to select 2 images for each challenge. The *Exact Solution* bars in Figure 4 present the results, and we can see that all the image annotation services were quite effective in identifying the correct images. Clarifai is the most effective as it selected the exact set of images in 40.2% of the challenges, while Alchemy reached 31.5% and Caffe 28.3%.

Tag classifier. To quantify the effectiveness of our tag classifier as part of our captcha-breaking system, we followed a 10-fold validation approach for training and testing our classifier on the dataset of 700 labelled image captchas. In our first experiment, we skipped the other image selection steps, and relied solely on the classifier for selecting the images. For each image, the classifier received as input the hint and the set of tags, and returned a “similarity” score; we selected the 3 images with the highest score. Our attack provided an exact match solution for 26.28% ($\sigma = 7.09$), and passed 44.71% ($\sigma = 6.39$) of the challenges. In the second experiment, we incorporated our classifier into our system, and used the classifier-based selection as a replacement of the overlapping-based selection of images from the *undecided* set. When using the classifier, our attack’s average accuracy for Clarifai reached 66.57% ($\sigma = 7.53$), resulting in an improvement of about 5.3%. The classifier is more effective than the overlap approach, as it identifies specific subsets of tags that are associated with each hint, instead of the more simplistic metric of the number of common tags. Furthermore, the use of the classifier does not impact the performance of the attack as the duration is increased by ~ 0.025 sec.

Live attack. To obtain an exact measurement of our attack’s accuracy, we run our automated captcha-breaker against reCaptcha. To minimize our impact, we do not repeat all previous combinations, but opt for the one that had the best results in the simulated experiments and, as such, we employ the Clarifai service.

Labelled dataset. We created a labelled dataset to exploit the image repetition. We manually labelled 3,000 images collected from challenges, and assigned each image a tag describing the content. We selected the appropriate tags from our *hint_list*. We used pHash for the comparison, as it is very efficient, and allows our system to compare all the images from a challenge to our dataset in 3.3 seconds.

We ran our captcha-breaking system against 2,235 captchas, and obtained a 70.78% accuracy. The higher accuracy compared to the simulated experiments is, at least partially, attributed to the image repetition; the history module located 1,515 sample images and 385 candidate images in our labelled dataset. We also came across 4 pairs of identical challenges. In one case, we had solved the challenge correctly the first time we received it, indicating that Google does not remove challenges from the pool even if they are answered correctly.

Figure 5 shows the average number of images added to the *select* and *discard* lists. We break the numbers down to challenges that our system passed and those it failed against. The *Undecided* column depicts the number of images we select from the *undecided* set using the overlap module or our skip-gram classifier, for reaching the 3 images we provide as the solution. As each module creates its own lists, the overall number of images in the *select* lists is higher than 3 due to images being in multiple lists. Our attack is more successful when Clarifai provides accurate tags that can be directly matched to the hint and the *hint_list*; passed challenges have an average of 1.5 selected images and 4.25 discarded, while failed ones have 1.09 and 3.8 respectively. The history, best guess, and title page are also effective in discarding images, which is beneficial as it reduces the chance of selecting wrong images from the *undecided* set. On average, we selected 1.49 *undecided* images for the solution when we succeeded. For the failed challenges, the average is 2, verifying that it is a more error-prone process.

Figure 6 shows the attack’s total time, broken down to each phase. The most time consuming phase is running GRIS, as it searches for all the images in Google and processes the results, including the extraction of links that point to higher resolution versions of the images. Currently, we follow a multi-threaded approach for processing the 10 images in parallel. This could be improved by further parallelizing the processing of each image. Nonetheless, the attack is very efficient, with an average duration of 19.2 seconds per challenge.

Hint repetition. We also found significant repetition of the type of content presented in the challenges. Figure 7 depicts all the hints and their respective frequency. As can be seen, there is a very limited variety of image categories used. 5 types account for the solution of over 54.7% of the challenges, and 10 for over 91.5%. An adversary can further increase the accuracy by tailoring the attack and training the image annotation system for these specific types of images.

Time restriction. The widget removes the challenge after 55 seconds and the user is required to click in the checkbox for receiving a new one. However, we found that the chal-

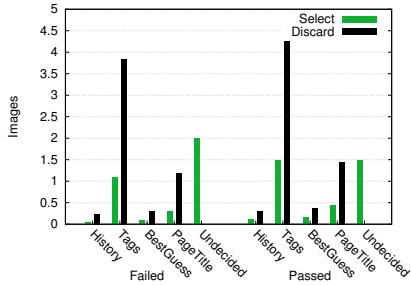


Figure 5. Average number of images selected or removed by each module.

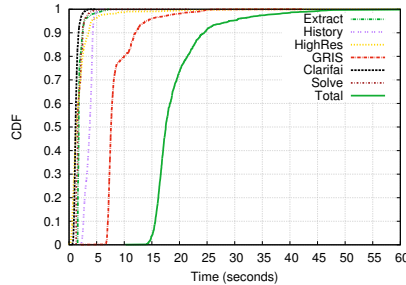


Figure 6. Cumulative distribution of time required for each step.

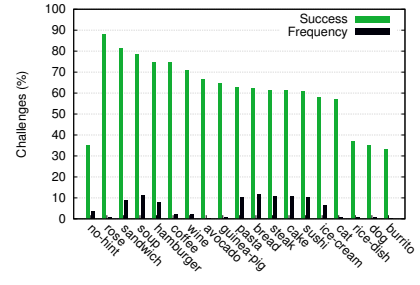


Figure 7. Frequency and success rate for each type of hint.

lenge is not invalidated on the server and we can provide an answer even after 20 minutes. The challenge is simply hidden by the widget, and changing the `visibility` and `position` attributes back to their original value makes the challenge reappear.

Offline mode. We also evaluated our attack in an *offline mode*, where we did not use any online annotation services or Google’s reverse image search; we relied solely on the local library, our labelled dataset, and our skip-gram classifier. We tested our attack with the two libraries, NeuralTalk and Caffe. When using Caffe and our classifier, our system solved 41.57% ($\sigma = 4.28$) of the image captchas, while the attack duration increased to 20.9 seconds per challenge. While NeuralTalk is similarly accurate at $\sim 40\%$, it introduces a large increase to the attack’s duration; specifically, the duration of the attack increases to 117.8 seconds, as NeuralTalk requires an average of 110.9 seconds to process the 10 images. However, leveraging the capabilities of a GPU for the computations will improve the performance and reduce the duration.

Thus, adversaries can deploy accurate and efficient attacks against the image reCaptcha without relying on external services, which may require payment for processing large collections of images or report suspicious actions.

4.3. Economic analysis

Since captcha-breaking is driven by monetary incentives, we evaluate our findings from an economic perspective.

Image captcha. To evaluate the viability of our captcha-breaking system as a solution for fraudsters, we compare our performance to that of Decaptcher, the (self-reported) oldest captcha-solving service. We selected Decaptcher for two reasons. First, it supports the image reCaptcha, charging \$2 per 1000 solved captchas (customers can request refunds for failed challenges). Second, previous work [5] found it to be the most accurate solving service (tied with another service), rendering it a suitable candidate for comparison.

We submitted the 700 image captchas to Decaptcher and measured the response time and accuracy (taking into account the solution flexibility). Interestingly, 147 were initially rejected due to the service being overloaded, and had to be re-submitted at a later time. Out of the 700 captchas, 88 received a time-out error (ERR_TIMEOUT) as

the solvers did not provide an answer in the time window allocated by the service and 258 (36.85%) were an exact match. When taking into account the flexibility, 321 (44.3%) of the captchas were solved. The average solving time for the challenges that received a solution was 22.5 seconds. While the accuracy may increase over time as the human solvers become more accustomed to the image reCaptcha, it is evident that our system is a cost-effective alternative. Nonetheless, *our completely offline captcha-breaking system is comparable to a professional solving service in both accuracy and attack duration, with the added benefit of not incurring any cost on the attacker.*

Checkbox captcha. Assuming a selling price of \$2 per 1,000 solved captchas, our token harvesting attack could accrue \$104 - \$110 daily, per host (i.e., IP address). By leveraging proxy services and running multiple attacks in parallel, this amount could be significantly higher for a single machine.

5. Applicability

We have demonstrated the effectiveness of our attack against the image reCaptcha. Nonetheless, the basis of our attack is widely applicable; by extracting the semantics of images, we can construct attacks against other image-based captchas. While other schemes may require different associations of objects, our findings demonstrate that extracting semantic information from images is no longer an obstacle for machines, and should not be the yardstick by which we evaluate the security of image captchas.

Currently, our approach can be readily applied to similar existing schemes, such as the recently released Facebook captcha (Figure 8). The image captcha is shown to users when they send messages to other users that contain suspicious URLs. Before the message is actually delivered to other users, the sender must pass an image captcha. This mechanism is designed to prevent the propagation of (automated) spam or phishing links by fake or compromised accounts. Facebook’s image captcha follows the same approach with reCaptcha, where users have to identify which images (out of 12 candidates) have content that match a given hint. There are, however, some differences. Facebook resizes the images dynamically in HTML, allowing access to the high resolution versions. Also, no sample image

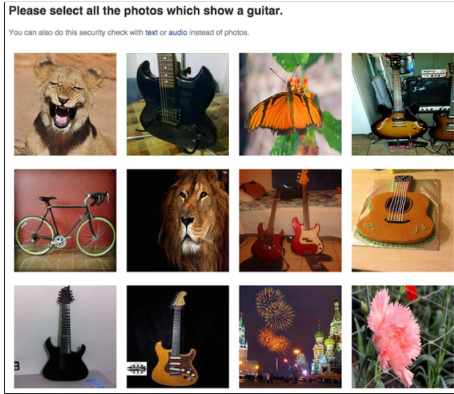


Figure 8. Image captcha by Facebook.

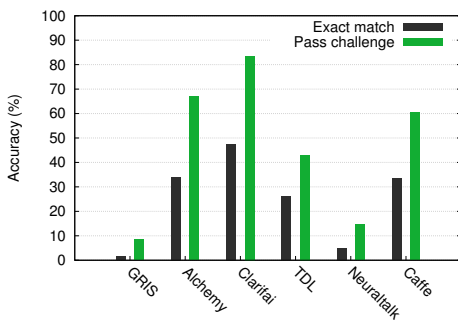


Figure 9. Attack accuracy against Facebook’s image captcha.

is shown. The system allows the same flexibility rules as reCaptcha. The number of correct images varies from 2 to 10, with 5-7 being the most common cases. Accordingly, we tweak our solution algorithm to only select the images contained in the `select` set, and not to opt for a specific number of images.

Figure 9 depicts the accuracy of our attack against 200 Facebook image captchas. Once again, Clarifai achieves the highest accuracy with 83.5% while Alchemy is also very effective with 67%. The higher accuracy, compared to reCaptcha, is due to two characteristics. First, the higher resolution images which lead to more accurate labels from the annotation systems. Second, the use of completely unrelated images when creating the challenge facilitates the discarding of the incorrect options; on the other hand, reCaptcha opts for images that belong to the same category (e.g., all are some type of food) which renders the distinction more difficult. Furthermore, these results demonstrate that while increasing the number of images significantly impacts random guessing attacks, it does not affect our attack, as other aspects of the captcha’s implementation are more influential.

6. Guidelines and Countermeasures

Here we discuss countermeasures for defending against our attacks, and their potential impact on the usability of the service. These measures are not exclusive to reCaptcha,

and can be adopted by other captcha providers. Since automated solvers for the image captcha cannot be prohibited, we present guidelines for reducing the accuracy and cost-effectiveness of the image captcha attack. Due to the potential impact, extensive user studies are required for evaluating each modification.

Token auctioning. The token verification API call has an optional field for comparing the IP address of the user that solved the captcha, to the one that submitted the token to the website. This field should be made mandatory to prevent services from selling tokens obtained from checkbox captchas. However, it cannot prevent outsourcing the solution for the other types of captchas, since the adversaries can extract the challenges, send them to the solving service and receive the solution. Nonetheless, it increases the cost of large scale attacks, as (automatically) solving the other captcha types is more costly in terms of computation.

Risk analysis. We propose the following safeguards for improving the advanced risk analysis system.

Account. Previous work [23] has proposed regulating the number of available challenges by tying them to a service account. If requests are valid only when they that originate from users logged into their Google account, the scale of attacks could be constrained. While this would result in the value of accounts increasing in the underground market, Google has deployed several safeguards that undermine the ability of adversaries to maintain phone verified accounts [21]. On the other hand, this introduces a usability issue, as users that are browsing with the privacy (incognito) mode enabled, will be blocked. As a workaround, for such cases reCaptcha can return the hardest type of challenge and maintain a separate, severely limited, bucket of tokens [3] per IP address. Once the tokens have been exhausted for a specific time period, no more challenges should be returned to requests that do not originate from a logged in user.


Cookie reputation. Assigning high confidence to cookies with no browsing history is a significant flaw. The “reputation” of a cookie should elevate with the amount of browsing conducted. This will increase the cost of cookie-creation. Also, the number of cookies that can be created within a time period from a specific host should be regulated. While cookie creation was not a problem in the past, their use in reCaptcha has rendered them a valuable commodity.

Browser checks. A stricter approach than the existing would be to not return any challenge if the checks detect an overtly suspicious environment (e.g., mismatch between detected browser and what is reported in the `User-Agent`).

Image captcha attacks. Due to recent advancements in computer vision and machine learning [12], [14]–[16], [18], [19], and the availability of free image annotation services and libraries, developing a captcha scheme that can defeat automated solvers remains an open problem. Here, we propose modifications and safeguards focused on reducing the accuracy of our automated attacks in lieu of such a development.

Solution. Creating challenges where the number of correct images is selected from a larger range, and uniformly distributed, will impact the accuracy of the attack. Without

TABLE 6. EXAMPLE OUTPUT FOR IMAGE WITH ARTIFICIAL NOISE.

	Clarifai	Caffe
	modern, glass, business, window, office, light, reflection, office building, communication, architecture, futuristic, future, panoramic, technology, nobody, city, structure, geometric, bright, building	prison, correctional institution, penal institution, institution, shoji, prison, shoji, pirate, pick, bearskin

knowledge of the number of images to select, the attack will require a threshold confidence score to be set for selecting an image, which may result in correct images not being selected. Furthermore, by not allowing any flexibility in the solution and requiring users to only select correct images, the attack’s accuracy can be further reduced.

Repetition. Once a challenge has been presented to a user, it should be removed from the pool of available challenges, even if the given solution was incorrect. Furthermore, the pool from which images are selected should be greatly increased in size. This will minimize the existing amount of repetition which can be exploited by adversaries.

Hint and content. As shown in Figure 7, when a captcha does not contain a hint, our attack is less successful. As such, the hint should be removed. Furthermore, the attack is less accurate for certain types of content. Captcha providers can conduct experiments for identifying more categories that are problematic for image annotation software and use those to construct challenges.

Content homogeneity. By populating challenges with “filler” images (i.e., those that are not part of the solution) that have the same type of content as the solution, may impact the accuracy of the attack. The output of the image annotation systems is not always precise enough to discern similar types of content (e.g., if all the images are some type of food). The heterogeneity of the filler images in Facebook’s captcha contributes significantly to the attack’s higher success rate.

Advanced semantic relations. Requiring users to identify images with more complicated semantic relations could increase the complexity required for the captcha-breaking system. Instead of similar objects, challenges could ask users to select semantically-related objects (e.g., a tennis ball, a racket, and a tennis court). While this cannot completely prevent attacks, it can be adopted as a temporary measure.

Introducing noise. We have conducted a preliminary experiment for evaluating the impact of artificial noise on the effectiveness of the image annotation services. Our system draws a random grid on each image, with a varying number of lines on each axis, and varying distances between each line. The grid’s color is chosen using the complimentary value of the image’s average color value, to make it more discernible to human users. We run our attack against 100 image reCaptcha challenges, and find that the attack’s accuracy drops to 16% for Clarifai and 13% for Caffe. An

example output can be found in Table 6. Interestingly, the image annotation systems are not impacted by the grid when the original image depicts an animal, as the grid is identified as a cage. The grid also reduces the probability of retrieving higher resolution versions of the images, as the reverse search is significantly affected.

As previous work has demonstrated effective methods for removing noise from images [24], [25], we consider this a temporary solution, that can increase the cost of the automated attack. As images will have to be “cleaned” before being processed by the image annotation system, these mechanisms may increase the computational cost of the attack. Further exploration is required for evaluating the trade-off between the amount of noise introduced (which may impact usability) and the time required by adversaries for removing it.

Adversarial images. Szegedy et al. [26] described a pixel-level distortion process that produces images that are almost identical (visually) to the original, yet are misclassified by all the neural networks they tested, regardless of their model parameters or training datasets. By altering a small number of pixels, the resulting images are misclassified (e.g., a school bus classified as an ostrich) or not recognized (e.g., a binary car classifier fails to recognize an image previously identified as a car). This process could be applied to image captchas, as it can prevent automated attacks from identifying the images, while the images remain easily identifiable for humans. However, evaluation is required for verifying the performance overhead of such an approach, as well as its robustness; it may be easy for adversaries to transform the images in a way that negates the distortion, before processing them with the image annotation system.

7. Ethics and disclosure

Demonstrating large scale attacks against reCaptcha has implications for the many websites that rely on it for securing resources and protecting their users. While the presented image-based attack cannot be prohibited, we propose safeguards and countermeasures for impacting the accuracy and cost-effectiveness of our attack, until a suitable replacement mechanism can be found. Furthermore, we have not affected the websites in any way during our experiments, as we do not perform any actions apart from acquiring the reCaptcha challenges. We have disclosed a report with our findings and recommendations to Google, in an effort to assist them in making reCaptcha more robust to automated attacks. Following our disclosure, reCaptcha altered the safeguards and the risk analysis process to mitigate our large-scale token harvesting attacks. They also removed the solution flexibility and sample image from the image captcha for reducing the attack’s accuracy. We have also informed Facebook, but have not been notified of any changes. Overall, we hope that sharing our findings will help initiate the much-needed discussion between researchers and industry regarding the future of captchas.

8. Limitations

Certain aspects of our system can be explored for further improving the accuracy of our attack.

Customize. Our current design does not account for certain characteristics of the image annotation modules. For example, Clarifai returns generic tags which can lead to false positives. In the future we plan to explore how the attack is impacted if the generic tags are removed, and identify specific types of keywords that can be omitted.

Confidence Scores. Several of the image annotation systems return confidence scores. However, the highest scores are frequently assigned to generic tags. Our current version of the attack does not assign higher weights to tags with higher confidence scores, as it may increase the false positives. By exploring which generic tags can be omitted, we can modify our system to leverage the tag-confidence scores.

9. Future directions for captchas

The results by Bursztein et al. [6] pose significant implications regarding the robustness of text-based captcha schemes and the security they offer. Similarly, our novel attacks pose an interesting dilemma regarding future directions for designing captchas. We believe that *the capabilities of computer vision and machine learning have finally reached the point where expectations of automatically distinguishing between humans and bots with existing schemes, without excluding a considerable number of users in the process, seem unrealistic.* Thus, we must reassess our concept of Reverse Turing tests, and approach their design from a fundamentally different perspective. While an in depth exploration of this open problem is out of the scope of our work, we present certain alternative schemes in the next section. An overview of potential alternative captcha designs was recently presented in [6].

10. Related Work

Text captchas. The majority of deployed captcha schemes are based on distorted letters or numbers, and a broad body of work has demonstrated attacks for automatically breaking such schemes. Yan and Ahmad [27] presented an attack against a Microsoft captcha. Their novel text segmentation approach focused on locating the separate characters which are, then, easier to identify. The authors had previously [28] demonstrated attacks against a variety of captcha schemes, using simple pattern recognition algorithms and exploiting design errors. Li et al. [29] explored the robustness of captcha schemes used by multiple e-banking services and found that in most cases they could achieve 100% recognition accuracy. They concluded that, in an attempt to maintain the usability of their systems, e-banking services opted for captcha designs that offered little security against automated attacks. Mori and Malik [30] presented two methods for breaking the EZ-Gimpy and Gimpy captchas that depict words over artificial noise, achieving a

success rate of 92% and 33% respectively. The attacks take advantage of the fact that the challenges depict actual words and not random strings of alphanumeric characters.

Bursztein [31] measured the success rate of users against captchas from various services. In their study they employed users through an underground solving service and workers from Amazon Turk. An important observation the authors made is that the difficulty of captchas is often very high, rendering their solution a troublesome process for users.

Various attacks have been demonstrated against previous text versions of reCaptcha [32]–[34]. Bursztein et al. [35] conducted an extensive study on the strengths and weaknesses of text captchas. During their evaluation they found 13 of the 15 schemes to be vulnerable to automated attacks. In [36] the authors proposed a more user-friendly scheme based on distorted numbers, with users passing 95.3% of the challenges. Recently, Bursztein et al. [6] presented a novel approach for breaking text captchas, that performs character segmentation and recognition in a single step. Most importantly, their approach was universally applicable to all tested schemes, and solved them with varying levels of accuracy (5.33-55.22%).

Alternate designs have also proposed the use of video CAPTCHA challenges that contain text. Xu et al. [37] demonstrated highly effective attacks against the NuCaptcha scheme that depicted dynamic text strings.

Image captchas. The first publication to rigorously explore the use of automated tests for security [1], also proposed the use of distorted images of animals for creating challenges. The Asirra captcha [3] required users to distinguish between images depicting cats and dogs. It was broken within a year [38] with a classifier trained on color and texture features.

Chew and Tygar [13] proposed three image captcha schemes where users are required to: (i) type a label that describes six images with similar content (e.g., astronaut), (ii) detect if each of 2 sets of images contains similar content, and (iii) identify one image out of 6 that has content different to the others. Our attack against the image captcha can be readily applied to the second and third type without complications as it is based on the same principle; identifying images with the same content. The first type would require an extension of our existing attack, for deciding which tag should be supplied as the answer.

The use of human faces for captchas has been proposed in multiple schemes. Goswami et al. proposed FaceD-captcha [39], a scheme where users are required to differentiate between actual images of human faces and animated versions of human faces. Rui and Liu [40] proposed ARTiFACIAL, a captcha scheme where users are required to identify faces and facial features within a heavily distorted image. Zhu et al. [8] demonstrated attacks against a series of image-based captcha schemes, including ARTiFACIAL. Based on the insight gathered by their attacks' characteristics they set guidelines for designing robust image-based captcha schemes. Their guidelines mandated, among other, that a captcha must rely on semantic information, require identification of multiple types of objects and prohibit at-

tacks based on a-priori knowledge such as the type of objects. While the image reCaptcha design does exhibit these characteristics, the current implementation breaks the third guideline as we found that the types of objects presented are from a limited selection of categories.

Social authentication. Yardi et al. [41] proposed photo-based authentication for social networks; to verify their identity, users are required to identify their friends in photos. Such a system was eventually deployed by Facebook for preventing adversaries from gaining access to user accounts after acquiring their credentials. This could be offered by Facebook as a captcha service for other websites. However, Polakis et al. [42] demonstrated an attack that leveraged publicly available data and face recognition algorithms for solving the challenges. In follow up work [23] the authors proposed a photo-based captcha scheme, based upon the same principle, which created challenges robust against face recognition and image comparison attacks. As the captcha challenges are crafted specifically for each user, such a scheme can prevent captcha-solving services [5] or smuggling attacks [43]. The drawback is the requirement for users to have an account with a specific service. ReCaptcha alleviates that requirement with cookies, which reveal information about a user’s activities without requiring a Google account.

Captcha-solving services. Motoyama et al. [5] explored the inner workings of captcha-solving markets from an economic perspective. They concluded that captchas should not be viewed as an isolated defense mechanism, but evaluated as a way to impact the attackers’ profitability at a large scale. In our experiments we have demonstrated the feasibility of low-cost large-scale attacks against reCaptcha that can be highly profitable for solving services. Shin et al. [44] analyzed the functionality of a popular forum spam automator designed to solve arithmetic tasks, which also contained answer-question pairs for trivia challenges.

As fraudsters can distribute their illicit activities across many hosts (e.g., by employing a botnet [45]), (IP address-based) token bucket approaches [3] cannot prevent large scale captcha-solving attacks. Leveraging reputation has been proposed as a mitigation technique [6]. Jakobsson [46] argued that existing approaches for throttling access to valuable resources through captchas are no longer a viable solution; challenges are becoming increasingly difficult for humans, while the effectiveness of automated attacks continues to improve. As an alternative, Jakobsson proposed a mechanism where users are required to create an account with a trusted third party that serves as a mediator. If the trusted party can verify the machine’s “identity” the website will grant access to the user. To effectively throttle requests from a machine, and prevent large scale attacks, accounts must be tied to (physical) resources that are, by nature, restricted. Such resources may be bank accounts, phone numbers or postal addresses. The current approach by reCaptcha is similar, in the sense that it treats users’ browsing history as a resource, and distinguishes users based on their web history (cookie). We have demonstrated, however, that this is not a restricted resource and the system can be manipulated. Furthermore, while recent work has

found that underground markets use phone-verified Google accounts [21], periodic verification of each account’s phone number could significantly mitigate such incidents. Thus, requiring a phone-verified account for presenting a captcha may be an effective direction to explore.

Dynamic cognitive game captchas have been proposed as an alternative, and mechanisms for detecting the outsourcing of the solution have been demonstrated [47]. While the most popular game captcha available has been broken [48], this approach could potentially lead to more robust captcha schemes.

11. Conclusion

This paper offers a comprehensive examination of the design and implementation characteristics of the reCaptcha service. Our findings demonstrate that the development of new generation captcha systems is a complex task, even for resource-rich entities such as Google and Facebook. In an effort to evolve towards more advanced functionality and address the issues of previous schemes, reCaptcha has introduced a series of mechanisms that have not been used in captcha systems before. Accordingly, we explored the design directions of this new system, as well as the implementation flaws. Due to the essential role of the risk analysis system, we evaluated how different aspects of our captcha-breaking system affect the outcome of our requests, and demonstrated a novel misuse of persistent tracking cookies. We identified a plethora of checks performed by the reCaptcha widget for identifying suspicious characteristics of the environment. We also found a lack of safeguards for preventing the creation and use of tracking cookies, which we exploited to demonstrate the feasibility of large-scale captcha-solving attacks. Next, we presented a novel no-cost attack for image captchas that builds on deep learning technologies for extracting semantic information from images. The effectiveness and efficiency of our attack further corroborate that new directions need to be explored for the design of captchas, as existing schemes rely on tasks that are within the capabilities of automated cognisance. Nonetheless, the advanced risk analysis and widget introduced by reCaptcha possess valuable functionality that can be incorporated into future captcha schemes for mitigating attacks.

Acknowledgements

We would like to thank the anonymous reviewers, as well as Fabian Monrose and Michalis Polychronakis for their comments on previous drafts of this paper. This work was supported by the NSF under grant CNS-13-18415. Author Suphanee Sivakorn is also partially supported by the Ministry of Science and Technology of the Royal Thai Government. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors, and do not necessarily reflect those of the US Government or the NSF.

References

- [1] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard ai problems for security," in *EUROCRYPT '03*.
- [2] M. Foley, "'Prove You're Human': Fetishizing material embodiment and immaterial labor in information networks," *Critical Studies in Media Communication*, vol. 31, no. 5, pp. 365–379, 2014.
- [3] E. Jeremy, J. R. Douceur, J. Howell, and J. Sault, "Asirra: a CAPTCHA that exploits interest-aligned manual image categorization," in *CCS '07*.
- [4] Distil Networks. CAPTCHAs Have Negative Impact on Web Traffic and Leads. <http://www.distilnetworks.com/distil-networks-study-captchas-negative-impact-on-web-traffic/>.
- [5] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Re: CAPTCHAs: understanding captcha-solving services in an economic context," in *USENIX Security '10*.
- [6] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based CAPTCHAs," in *USENIX WOOT '14*.
- [7] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in html5," in *W2SP '12*.
- [8] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai, "Attacks and design of image recognition CAPTCHAs," in *CCS '10*.
- [9] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-based character recognition via web security measures," *Science*, vol. 321, no. 5895, 2008.
- [10] Google Online Security Blog, "Are you a robot? Introducing 'No CAPTCHA reCAPTCHA'," <http://googleonlinesecurity.blogspot.com/2014/12/are-you-robot-introducing-no-captcha.html>.
- [11] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnaud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," in *CoRR '13*.
- [12] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *CoRR '14*.
- [13] M. Chew and J. D. Tygar, "Image recognition CAPTCHAs," in *ISC '04*.
- [14] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CoRR '14*.
- [15] M. M. Kalayeh, H. Idrees, and M. Shah, "NMF-KNN: Image Annotation Using Weighted Multi-view Non-negative Matrix Factorization," in *CVPR '14*.
- [16] N. Srivastava and R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," *Journal of Machine Learning Research*, vol. 15, pp. 2949–2980, 2014.
- [17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS '12*.
- [18] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *ICCV '11*.
- [19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding."
- [20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *CoRR '13*.
- [21] K. Thomas, D. Iatskiv, E. Bursztein, T. Pietraszek, C. Grier, and D. McCoy, "Dialing back abuse on phone verified accounts," in *CCS '14*.
- [22] E. Homakov. The No CAPTCHA problem. <http://homakov.blogspot.in/2014/12/the-no-captcha-problem.html>.
- [23] I. Polakis, P. Iliu, F. Maggi, M. Lancini, G. Kontaxis, S. Zanero, S. Ioannidis, and A. D. Keromytis, "Faces in the distorting mirror: Revisiting photo-based social authentication," in *CCS '14*.
- [24] R. H. Chan, C.-W. Ho, and M. Nikolova, "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization," *Trans. Img. Proc.*, vol. 14, no. 10, 2005.
- [25] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image," *IEEE TPAMI*, vol. 30, no. 2, 2008.
- [26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *CoRR '13*.
- [27] J. Yan, E. Ahmad, and A. Salah, "A low-cost attack on a microsoft CAPTCHA," in *CCS '08*.
- [28] —, "Breaking visual CAPTCHAs with naive pattern recognition algorithms," in *ACSAC '07*.
- [29] S. Li, S. A. H. Shah, M. A. U. Khan, S. A. Khayam, A.-R. Sadeghi, and R. Schmitz, "Breaking e-banking CAPTCHAs," in *ACSAC '10*.
- [30] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA," in *CVPR '03*.
- [31] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How good are humans at solving CAPTCHAs? A large scale evaluation," in *SP '10*.
- [32] C. Cruz-Perez, O. Starostenko, F. Uceda-Ponga, V. Alarcon-Aquino, and L. Reyes-Cabrera, "Breaking reCAPTCHAs with unpredictable collapse: Heuristic character segmentation and recognition," vol. 7329, 2012.
- [33] P. Baecher, N. Büscher, M. Fischlin, and B. Milde, "Breaking reCAPTCHA: a holistic approach via shape recognition," in *Future Challenges in Security and Privacy for Academia and Industry*, 2011, vol. 354.
- [34] O. Starostenko, C. Cruz-Perez, F. Uceda-Ponga, and V. Alarcon-Aquino, "Breaking text-based CAPTCHAs with variable word and character orientation," *Pattern Recognition*, vol. 48, 2015.
- [35] E. Bursztein, M. Martin, and J. C. Mitchell, "Text based CAPTCHA strengths and weaknesses," in *CCS '11*.
- [36] E. Bursztein, A. Moscicki, C. Fabry, S. Bethard, J. C. Mitchell, and D. Jurafsky, "Easy does it: More usable CAPTCHAs," in *CHI '14*.
- [37] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. van Oorschot, "Security and usability challenges of moving-object CAPTCHAs: Decoding codewords in motion," in *USENIX Security '12*.
- [38] P. Golle, "Machine learning attacks against the asirra CAPTCHA," in *CCS '08*.
- [39] G. Goswami, B. M. Powell, M. Vatsa, R. Singh, and A. Noore, "FaceDCAPTCHA: Face detection based color image CAPTCHA," *Future Generation Computer Systems*, vol. 31, 2014.
- [40] Y. Rui and Z. Liu, "Artificial: Automated reverse turing test using facial features," in *Multimedia '03*.
- [41] S. Yardi, N. Feamster, and A. Bruckman, "Photo-based authentication using social networks," in *WOSN '08*.
- [42] I. Polakis, M. Lancini, G. Kontaxis, F. Maggi, S. Ioannidis, A. D. Keromytis, and S. Zanero, "All your face are belong to us: breaking facebook's social authentication," in *ACSAC '12*.
- [43] M. Egele, L. Bilge, E. Kirida, and C. Kruegel, "CAPTCHA smuggling: Hijacking web browsing sessions to create captcha farms," in *SAC '10*.
- [44] Y. Shin, M. Gupta, and S. Myers, "The nuts and bolts of a forum spam automator," in *USENIX LEET '11*.
- [45] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna, "The underground economy of spam: A botmasters perspective of coordinating large-scale spam campaigns," in *USENIX LEET '11*.
- [46] M. Jakobsson, "Captcha-free throttling," in *AISec '09*.
- [47] M. Mohamed, S. Gao, N. Saxena, and C. Zhang, "Dynamic cognitive game CAPTCHA usability and detection of streaming-based farming," in *USEC '14*.
- [48] SPAM tech. Cracking the AreYouAHuman Captcha. <http://spamtech.co.uk/software/bots/cracking-the-areyouhuman-captcha/>.