Raising the Bar for the CAPTCHA-breaking Economy

Dave Caffrey & Julian Timpner
University of California, San Diego

1 Introduction

CAPTCHAs - *Completely Automated Public Turing Tests To Tell Computers And Humans Apart* [6] - are a major defense against exploitation of Web resources, such as free email services. An effective CAPTCHA is very hard for a machine to solve reliably, yet quickly and easily solved by a human. This mechanism is employed to prevent robot attacks on Web services. As a reaction to the CAPTCHA defense, a "robust solving ecosystem" (cf. [2]) has been developed by persons either seeking to exploit Web services, or seeking to profit from other exploiters. This CAPTCHA-breaking (as opposed to legitimately "solving") industry develops and sells software designed to solve CAPTCHAs automatically. A second branch of the CAPTCHA-breaking industry simply employs human workers to solve CAPTCHAs manually. Since CAPTCHAs are by definition easily solved by humans, paid CAPTCHA solving circumvents most anti-bot techniques. In either case, large scale CAPTCHA breaking puts Web resources into the hands of spammers seeking to profit from abuse. Consequently, each broken CAPTCHA has a monetary value associated with it. This drives the development of CAPTCHA-breaking attacks.

Previous research has focused on making CAPTCHAs even harder for computers to solve in order to counteract advances in CAPTCHA-breaking software. Newer research, such as [2], has focused on understanding the increasing trend of using low-cost human labor to break CAPTCHAs. As Savage et al. observe, CAPTCHA-breaking boils down to an economic decision: what is the value of the target CAPTCHA-protected resource, and the cheapest way to get at it? It is increasingly clear that paid human CAPTCHA solving is the cheapest way. This development may be interpreted as a victory of defense technology against automatic CAPTCHA solving approaches. Yet, this only shifts the problem and raises the question: how can paid human solvers be prevented from breaking CAPTCHA defenses at scale?

It is unlikely that conventional CAPTCHAs can stop paid solving attacks, since they cannot be made arbitrarily difficult or lengthy to solve without preventing regular users from gaining access to the service at the same time. Defenses such as IP blacklisting may be valuable, but they do not address the weaknesses of current CAPTCHA approaches to paid solving.

We examine methods of raising the cost of paid CAPTCHA solving. We propose two potential solutions to this problem, which not only make large-scale paid solving more costly, but can also be combined with traditional systems, such as reCaptcha [9], establishing a defense in depth against both manual and automated attacks. Limited testing of our prototype implementations suggests some potential strengths as defenses, but also several weaknesses.

Section 2 details the implementation and evaluation of our approaches, which include a refundable account tariff and CAPTCHAs which exploit a potential local knowledge gap between target users and far-flung paid solvers. Section 3 discusses drawbacks and possible attacks on our systems. In Section 4, we present our conclusions.

2 Approaches

PayCAPTCHA

Our first system, PayCAPTCHA, takes advantage of the very low price paid for broken CAPTCHAs, which is approximately $0.5-$1 per 1000 CAPTCHAs [2]. PayCAPTCHA introduces the concept of giving a small, short-term credit to the Web service provider, who then refunds the credit after a certain period of time. The exact amount of the credit and the time frame for refunding, as well as the amount of the refund may be dependent on the type and brand of the service. To test the user acceptability of such an approach, and to determine reasonable values for these parameters, we conducted a user survey.

Implementation Details

In order to implement the PayCAPTCHA fees and refunds, we must employ an established payment infrastructure. Otherwise, deployment would be unlikely for several reasons. First, to establish a new form of online payment requires a great deal resources for infrastructure, security, licencing, etc. Second, it is unlikely that users, who are rightfully concerned about their privacy and online security, would be willing to give away financial information to a non-trusted entity.

Thus, PayCAPTCHA employs PayPal, a major online platform for payments and money transfers. PayPal is a widely accepted and globally available payment solution, with enough resources and know-how to provide our system with a user-friendly and secure transaction infrastructure. The PayPal Express Checkout platform makes it easy for the users to make a payment and allows the service provider to accept PayPal while retaining control of the checkout flow. We streamline the checkout flow as much as possible in order to provide a seamless user experience.

Figure 1 depicts the overall control flow of PayCAPTCHA defending user account creation for a Web service: Once the user fills out and submits the sign-up form, he will be redirected to PayPal to initiate his approval for the payment. PayPal then redirects the user back to the service's site, where the transaction will be completed automatically and the user is signed up for an account. With each new account instance, an associated database entry records the time and amount of payment. This enables a more flexible deployment of PayCAPTCHA. For instance, the service provider might let the user choose between different credit amounts and refund policies.

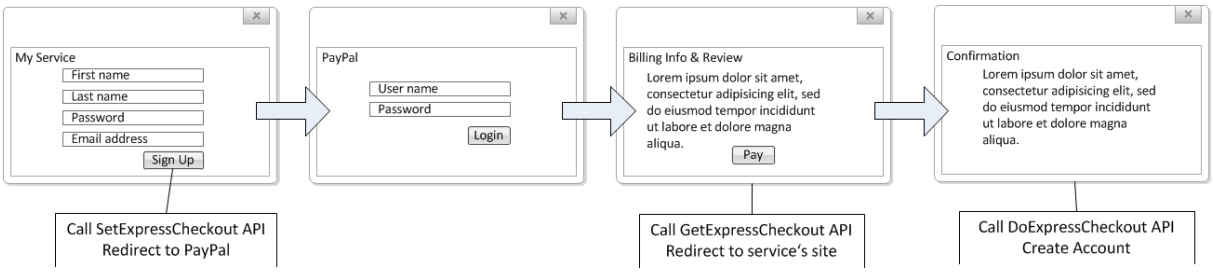


Figure 1: Express Checkout Workflow

An essential element of PayCAPTCHA is the automatic refund function. In our prototype, this is a script that is triggered regularly (typically daily) and checks the payment database entries for refunds due, according to the selected scheduling policy.
Both, the interaction with PayPal Express Checkout and the refund mechanism are implemented in PHP, using the PayPal NVP API.

Evaluation

To determine the feasibility of PayCAPTCHA, we deployed an online user survey. We interpret our survey results in order to determine reasonable fee structures for PayCAPTCHA, as well as to estimate the overall user acceptance of our system. As in Figure 2, approximately 80% of the respondents are generally willing to use PayCAPTCHA (though 30% only reluctantly).

1 -Not at all	6	20%
2	9	30%
3	5	17%
4	7	23%
5 -Very willing	3	10%

Figure 2: Willingness to pay a small fee, in case of a 100% refund a few days later

The feasible fee, one that more than 85% of users would accept, appears to be less than $1. 53% of our respondents would be willing to pay an amount less than $5 (cf. Figure 3).
Assuming that a service charged $1, and taking into account that "there are many service providers that can solve large numbers of CAPTCHAs via on-demand services with retail prices as low as $1 per thousand" (cf. [2]), the deployment of PayCAPTCHA would result in increased cost for the attacker by a factor of 1000. Of course, one of the main characteristics of our approach is that the payment is (at least partially) refunded. Still, PayCAPTCHA would require large scale attackers to pay a significant amount of money up-front, making an attack much less feasible.

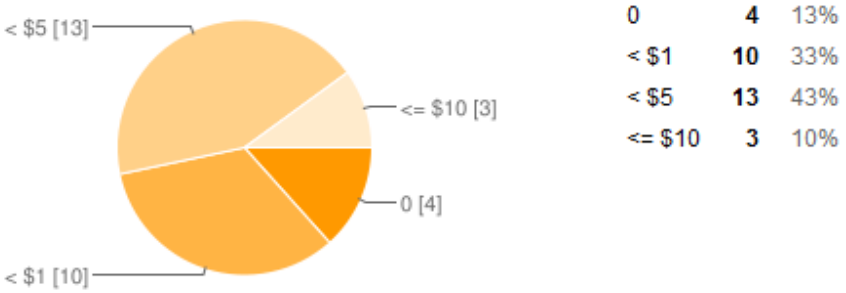


Figure 3: Maximum amount users would pay, if there is a full refund

PayPal charges a variable fee of 2.9% of each transaction amount plus a fixed fee of $0.30. When a refund is issued to a customer, PayPal keeps the $0.30 fee, so a full refund policy would result in and overhead cost of $0.30 per new account for the service provider. For large companies like Google or Microsoft this might be an acceptable cost, but for smaller providers this may be too expensive. Therefore, a smaller provider might want to keep $0.30 of the user credit, thus eliminating the overhead. Figure 4 depicts how users would react to such a reduced refund policy.

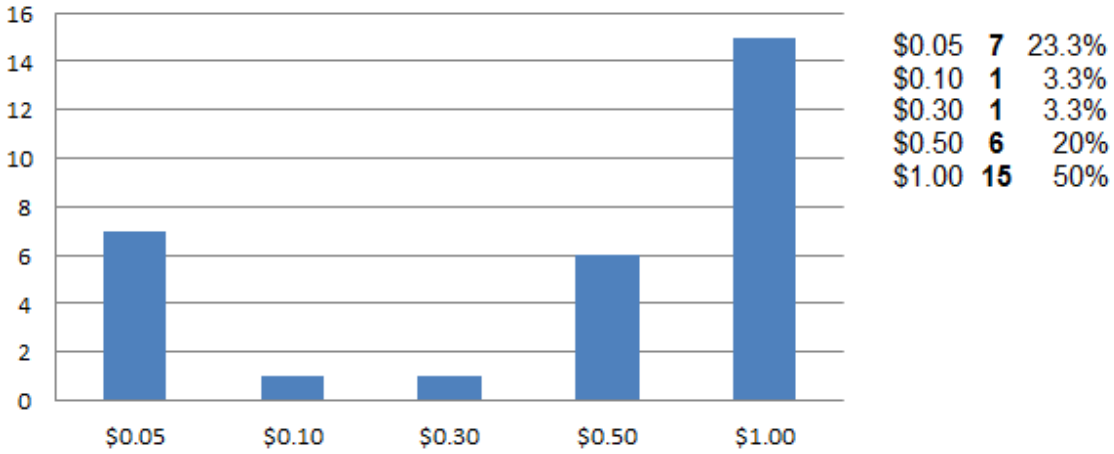
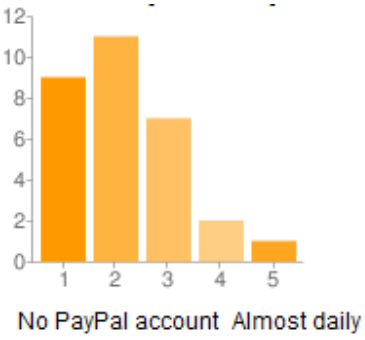


Figure 4: Maximum amount users would pay, if there is no refund

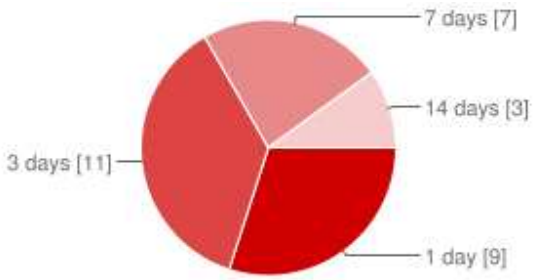
Almost 75% of all users would accept a service fee of $0.30 to $1 to finance the PayCAPTCHA infrastructure. Yet, a significant percentage would only accept an amount much smaller than the present cost. For this reason, one might argue that PayPal does not provide a cost-effective infrastructure for PayCAPTCHA.



1 -No PayPal account	**9**	30%
2	**11**	37%
3	**7**	23%
4	**2**	7%
5 -Almost daily	**1**	3%

Figure 5: Frequency of PayPal use

However, 70% of the surveyed users have a PayPal account, as can be seen in Figure 5, and would therefore be able to use PayCAPTCHA without additional inconvenience. This is a significant number of users, which is unlikely to be matched by any other payment service on the Web.



1 day	**9**	30%
3 days	**11**	37%
7 days	**7**	23%
14 days	**3**	10%

Figure 6: Maximum number of days users would wait for a refund

Depending on the number of days the user has to wait for a refund, there might be the option for the provider to compensate for overhead cost without passing it to the user. With 70% of the surveyed users willing to wait at least 3 days, the service provider would have opportunity to invest the credits they hold during the refund period. The return on investment might reduce infrastructure cost, such as PayPal fees.

Local CAPTCHA

Our second implementation aims to exploit the knowledge gap that may exist between the target audience of a localized service (such as a chat service or webmail for a college) and paid CAPTCHA solvers. Our Local CAPTCHA implementation presents the user with a map centered on the area of interest for the local service, and a challenge image of some landmark in that area (figure 7). The user must drag a marker on the map to the location of the subject of the challenge image. If the user submits a marker location that is within the margin of error of the correct location, then the test is passed.

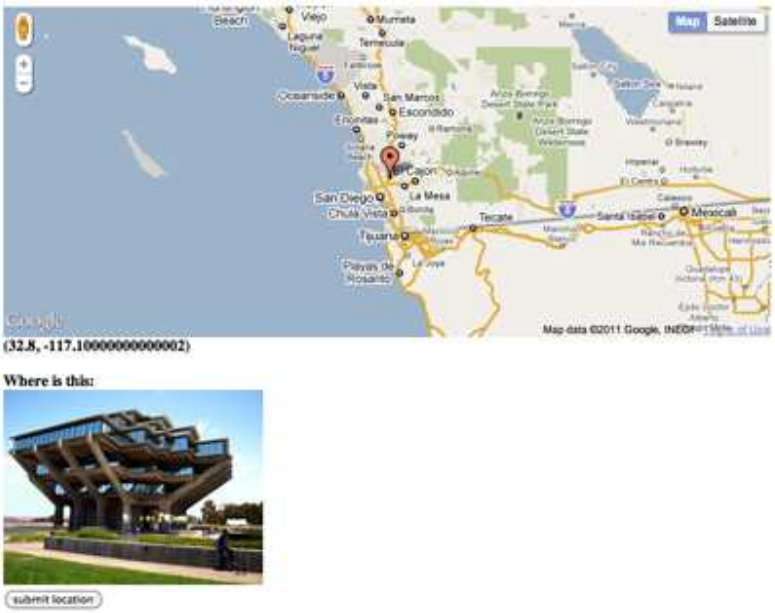


Figure 7: Local CAPTCHA Prototype: Map, Marker, and Challenge Image

Implementation Details

The Local CAPTCHA prototype serves the users with map from via Google Maps JavaScript API [8]. The challenge image is selected at random by a server script from a set of images within the location. The map is dragable and zoomable, and users can switch between the graphic map and satellite imagery. When the user submits a location, the server reports the location of the image. The Google Maps geometry library [8] is used to calculate the distance between the mark and the challenge location. If the mark is within an acceptable margin of error, then the user is directed to the success page.

Evaluation

We user-tested our implementation a map and pictures with two locations: The UCSD campus, which the test subjects were generally familiar with, and the city of Prague in the Czech Republic, which the subjects were not familiar with. Test subjects were able to correctly correlate pictures with map locations over the familiar location, submitting an average of 13 matches per 5 minutes (28.85s per CAPTCHA), with an average error rate of 20.8%. Subjects were permitted to utilize Internet search resources, but none elected to do so for the familiar location. Subjects had much more difficulty with the unfamiliar location. By employing Internet search resources, subjects submitted an average of 1 match in 5 minutes, with an error rate of 79.7%.

3 Drawbacks and Weaknesses

Several drawbacks are apparent in both of our prototype implementations. A Web service might deploy PayCAPTCHA and directly raise the cost of attacks, but at the same time our survey results suggest that at least some potential users would avoid the service as a result. Offering a full refund may reduce the number of legitimate users that would be discouraged by the PayCAPTCHA, but would increase overhead cost for the service provider. A service provider

must weigh the potential costs against the value of reducing the number of attacks. Our results suggest user's acceptance of PayCAPTCHA may depend heavily on the brand of the service, as shown in Figure 8. While Gmail seems to be of a high value to most users, significantly less users see that value in similar services of a different brand, such as Yahoo! Mail or Live.com.

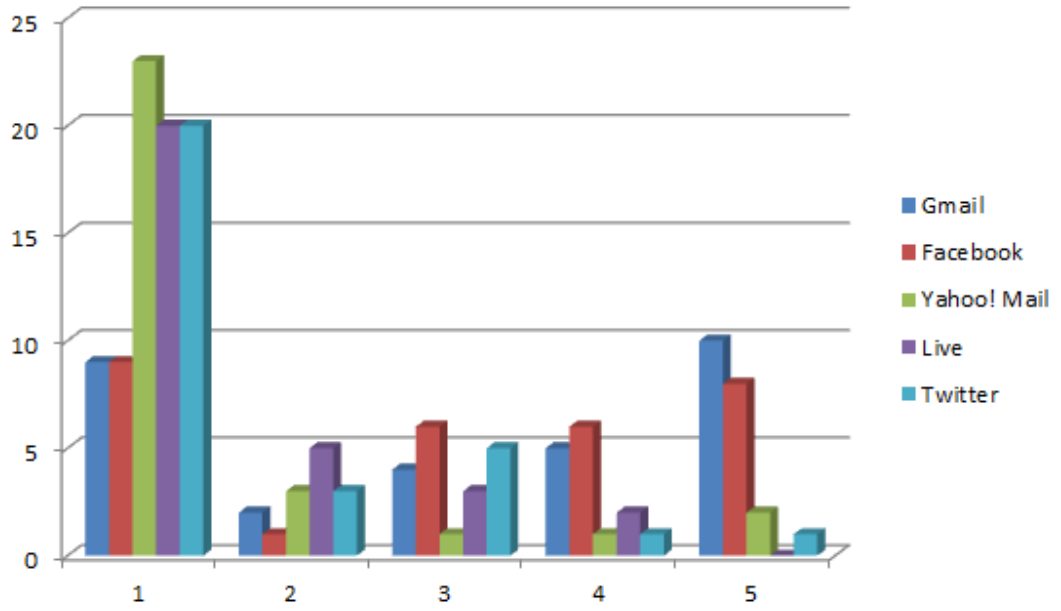


Figure 8: Users' willingness (1 - not at all, 5- absolutely) to sign up for a particular service, for a fee of $1 with a full refund in 7 days.

During our user studies, we also learned that PayPal France seems to have deployed a similar approach, which they use to verify addresses: users have to pay 1.50 Euros, which are credited back when the confirmation actually occurs. This demonstrates the general applicability of the PayCaptcha approach.

A drawback of the Local CAPTCHA system is in the resources required to deploy the system. Just as local knowledge of the subject area is needed to solve the challenge, local knowledge is required to create the challenges. Challenge image subjects must be recognized by the local target users, but not by a much wider audience. For example, the Eiffel tower would be a poor challenge image subject. At the same time, the subject must not be so obscure that even the target audience cannot recognize it. In addition, only images with a clear primary subject should be selected. These requirements make deploying and maintaining Local CAPTCHA more intensive than a traditional text-based CAPTCHA.

Another potential weakness of Local CAPTCHA is an attack by image recognition software. A large corpus of geo-tagged images is available online. Indeed, several tagged images with the same subjects as our UCSD test images are publicly available on Flickr.com. By focusing on a particular locality, we necessarily provide bounds for an attacker to limit an image search within. An attacker could request a set of images tagged within the locality via the Flickr API [7]. Then, the attacker might employ image matching software to compare the challenge image with the

set of local images. With a match, the attacker would associate the geo-tagged location with the image, and thus defeat the CAPTCHA in an automated fashion. Indeed, one of our test subjects uploaded some of the unfamiliar challenge images from Prague to the online image search tool Tineye.com [10], and produced a match which led to a correct location.

In order to prevent automated image-matching attacks, we propose to use image distortion to prevent image matching. In [4], the authors demonstrate image distortions which disrupt automated image matching, yet produce an image that is still be recognized by humans. The authors propose a CAPTCHA in which the subject must identify the subject of such a distorted image from a list of descriptive words. Image distortion which exploits this gap between machine and human recognition could be applied to the challenge images in our CAPTCHA system. By applying distortions, we were able to prevent the Tineye.com reverse image search from matching three of our test images which the system had previously recognized. Figure 9 presents an example. This example was a very small test, with only one matching system. However, the research conducted in [4] suggests that this approach might be expanded to defeat automatic image-matching attacks.

Figure 9: Original image at left. Distorted image at right, which breaks Tineye.com search.

4 Conclusion

Paid human solving has emerged as the most prevalent attack on Web resources which are defended by CAPTCHA. Current CAPTCHA approaches now provide only a small economic disincentive for attackers which employ bulk paid solving resources. PayCAPTCHA would directly increase the up-front cost of an attack by several orders of magnitude. Our survey results suggest that users would accept this defense, particularly for more popular resources. Local CAPTCHA offers locally-oriented services a defense which may drastically reduce the success of paid solving.

Bibliography

[1] Elie Bursztein, Steven Bethard, Celine Fabry, John C. Mitchell, and Dan Jurafsky. 2010. How Good Are Humans at Solving CAPTCHAs? A Large Scale Evaluation. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (SP '10). IEEE Computer Society, Washington, DC, USA, 399-413. DOI=10.1109/SP.2010.31 http://dx.doi.org/10.1109/SP.2010.31

[2] Marti Motoyama, Kirill Levchenko, Chris Kanich, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2010. Re: CAPTCHAs: understanding CAPTCHA-solving services in an economic context. In *Proceedings of the 19th USENIX conference on Security* (USENIX Security'10). USENIX Association, Berkeley, CA, USA, 28-28.

[3] Ritendra Datta, Jia Li, and James Z. Wang. 2009. Exploiting the human-machine gap in image recognition for designing CAPTCHAs. *Trans. Info. For. Sec.* 4, 3 (September 2009), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4956990&tag=1

[4] Peter Matthews and Cliff C. Zou. 2010. Scene tagging: image-based CAPTCHA using image composition and object relationships. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security* (ASIACCS '10). ACM, New York, NY, USA, http://doi.acm.org/10.1145/1755688.1755736

[5] Faster-than-SIFT Object Detection, Borja Peleato and Matt Jones. http://www.mattkjones.com/generic-trees.pdf

[6] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using hard AI problems for security. In *Eurocrypt*, 2003. http://www.captcha.net/captcha_crypt.pdf

[7] Flicker API at http://www.flickr.com/services/api/

[8] Goole Maps JavaScript API at http://code.google.com/apis/maps/documentation/javascript/

[9] reCaptcha at http://www.google.com/recaptcha

[10] TinEye reverse image search at http://www.tineye.com