A Machine Learning Attack against the Civil Rights CAPTCHA

Carlos Javier Hernández-Castro, David F. Barrero, and María D. R-Moreno

Abstract. Human Interactive Proofs (HIPs) are a basic security measure on the Internet to avoid several types of automatic attacks. Recently, a new HIP has been designed to increase security: the Civil Rights CAPTCHA. It employs the empathy capacity of humans to further strengthen the security of a well known OCR CAPTCHA, Securimage. In this paper, we analyse it from a security perspective, pointing out its design flaws. Then, we create a successful side-channel attack, leveraging some well-known machine learning algorithms.

1 Introduction

The abuse of free web-offered services was already a big issue in the 90's. First approaches to solve the problem proposed theoretical methods to prevent such attacks [10], under the idea of using problems (thought to be) hard for computers, but easy for humans, that is, Human Interactive Proofs (HIPs). Some years after, CMU Researchers improved this idea with a program to tell bots from humans apart, and listed the desirable properties such a program should have. Thus, the term CAPTCHA was coined. Their program used english words chosen at random and rendered them as images of printed text under a wide variety of shape deformations and image distortions. The user was asked to transcribe some minimum number of any of those words correctly.

During the 2000s decade they was a lot of research on new techniques [6, 9, 16] enabling the breaking of text-based word-image CAPTCHAs. All of these attacks

Carlos Javier Hernández-Castro
Universidad Complutense, Madrid, Spain
e-mail: chernandez@ucm.es

David F. Barrero · María D. R-Moreno
Computer Engineering Department, Universidad de Alcalá, Madrid, Spain
e-mail: {david,mdolores}@aut.uah.es

D. Camacho et al. (eds.), *Intelligent Distributed Computing VIII,*
Studies in Computational Intelligence 570, DOI: 10.1007/978-3-319-10422-5_26

were not really improvements of the state of the art in computer optical character recognition (OCR). Instead, they made a clever use of some very simple properties of the challenge images to undo part of the distortions or extract enough information from them [3]. Added to some design flaws, this allowed attackers to *read* them. Some companies created too difficult HIPs [5], whereas many researchers started looking into the broader AI problem of vision and image analysis.

Image-based CAPTCHAs need a large-enough database of labelled pictures. Ahn and Dabbish proposed a new way by creating a game, the "ESP game". The site Hot-Captcha.com was the first to propose using a large-scale, human-labeled database. Oli Warner proposed using photos of kittens to tell computers and humans apart [14]. Another proposal, the HumanAuth CAPTCHA, asks the user to distinguish pictures depicting either a nature-related image (e.g. a flower, grass, the sea), or a human-generated one (e.g a clock, a boat, or Big Ben). ASIRRA uses a similar approach based on cat/dog classification of images, but uses a large database "of more than 3 million photos from `Petfinder.com`". All these proposals have also been broken. Other, typically based on image analysis [13], like face classification [4], or even cartoons [15] had appeared recently, and await scrutiny. There are also some new proposals enhancing the typical OCR/text-based HIP [1, 8].

In this article we present a novel attack against the Civil Rights CAPTCHA, an original CAPTCHA that aims to join the strength of a typical word-recognition CAPTCHA, reinforcing it with an all-new empathy challenge. This combination purposely leads to a stronger, more secure CAPTCHA overall, and at the same time, makes users aware of Civil Rights news around the World.

Our attack is a side-channel attack, as it does not try to solve neither of the Artificial Intelligence (AI) problems used as a foundation for the CAPTCHA: it does not solve the empathy problem, nor the word recognition problem, as general AI problems. Instead, it solves both of them for this particular instance, or problem subset. We achieve so by identifying the security problems in the design of this HIP, and applying well-known Machine Learning algorithms to exploit them.

In the following sections, we first introduce the Civil Rights CAPTCHA (*CRC* from now on). We discuss the different design flaws found while closely examining it (section 3), and present a novel attack exploiting them. This attack uses some well-known machine learning algorithms to automatically solve it (section 4). Section 4 shows the results obtained using this approach. Finally, conclusions are outlined.

2 Civil Rights CAPTCHA

The *CRC* is based on the human ability to show empathy after being presented with a news excerpt, typically containing some news about Human Rights and/or Civil Rights around the World.

This CAPTCHA is based on Securimage, a word-distortion CAPTCHA (Fig. 2). How this Civil Rights CAPTCHA is supposed to work is simple: *CRC* picks up a Civil Right news from its DDBB, and uses Securimage to create three possible

answers, later presented to the user as distorted images containing words describing feelings (i.e. "agitated", "happy" and "angry"), who should write down the correct one based on the emotions originated from the news headline presented to her. This news bit is related to Human or Civil Rights, and supposed to create an emotion out of the empathy of a human reader. As a result, if we consider the CRC well designed, and Securimage security provides a security level *X*, this CAPTCHA design should provide an increased $3 \times X$, because picking up the correct answer should not be easier than random guessing. As we will see in brief, the security of this CAPTCHA is unfortunately below that *X* security level of Securimage.

CRC is provided as a service, directly accessible using an API allowing a programmer to connect to it, download a challenge composed of a news text, and three images containing one or two words distorted using *Securimage*, one of them the correct solution to the challenge. The same API allows to send to the *CRC* server the text the user inputs, to check if this is a correct answer (human) or not (program).

In order to analyze the *CRC*, we decided to analyze its client-server communication from the end-point, the same viewpoint a real attacker would have. We used a HTTP traffic analyzer . After a few interactions and tests, we were able to decipher the core of the client-server protocol needed in order to program an attack:

1. The first step is to request the main content for the CAPTCHA form, located at `http://captcha.civilrightsdefenders.org/captchaAPI/`. This, if presented as-is to the user, will be an *empty* HTML structure where the real content (news-bit, answer images) will be plugged in later using JavaScript. Another important function of this HTTP answer is to set the value of the *ci_session* cookie. This cookie is a meta-cookie containing several bits of information, like a *session_id*, the *IP* address of the client, information about its *user-agent*, etc.
2. Once this is loaded, the client JavaScript code makes another request with the same URL and *?sessid=1*. This sends back an answer containing the *PHPSESSID* cookie. This is the one the PHP library uses to keep track of client sessions.
3. In the next request, the parameters *callback, newtext* and *lang* are added to the URL, causing the server to send back the text of the challenge - the news bit. This comes back as a JSON encoded text.
4. The three images containing the answer are downloaded, each one requesting an unique (and random) 20-character id. The server keeps track of the ones to send using the previously provided cookies.
5. After the user has written the answer, this is sent to the server encoded in the URL of the next request.

3 Civil Rights CAPTCHA Design Flaws

After the analysis of the protocol, the first question raised naturally: what would happen if we keep asking the server for more word-image answers. We did, and confirmed that the server keeps providing us with new word-images, independently of adding or not the *newset* parameter.

This again raised a question: would it be possible that the server does not keep track of the word-images sent, and only checks that the answer is valid (positive, negative) according to the news bit?

We checked this hypothesis. In particular, we wrote down a few positive answers and a few negative ones from another questions. Then we proceeded to the next question, *"In October 2012 the Ukrainian parliament took the step to approve a law, which criminalises 'propaganda of homosexuality'. How does that make you feel?"*. The corresponding word-image answers were *very crappy, elastic* and *hopeful*. Being a negative news bit, we decided to answer with a negative answer present in other questions but not in this one, choosing *horrified*. Result: error! We tried the same attack a few more times, without suceess.

Our conclusion at this point was that either the answers are divided in finer categories or, more probable, the server keeps track of the word-image answers sent (probably the last three). To find out the correct hypothesis, we proceed with the attack (section 4), collecting logs of wrong and correct answers. Then, we tried again using these logs to find correct answers for each question. Again, we got a fail. The only possible conclusion is that the server keeps track of the word-image answers sent.

Once we finished playing around with the *CRC* and got familiar with how it operates, we wanted to analyze the challenges it could present to the users. We wanted to learn their number, distribution, and if any characteristic of the was not uniform. For this purpose, we wrote a program able to follow the protocol of the *CRC* (gathering and sending back the cookies, etc.), mimicking a regular user, download and interpret the information.

After downloading 1000 challenges, we learned that there are only 21 different challenge texts. This is in itself a flaw, as it is too low for a CAPTCHA.

We also checked how many times each challenge has been presented during this test. The questions have a seemingly uniform distribution of appearances, with a χ^2_{20} with a p-value of 0.336 (Fig. 1).

Each challenge comes with three different answers. How many total answers are there? During our experiments, we have been able to observe 130 different answers, 28 of them compositions of the words *quite, really, truly* and *very* and some of the remaining 102 basic categories. Again, this is a problem. A CAPTCHA with only

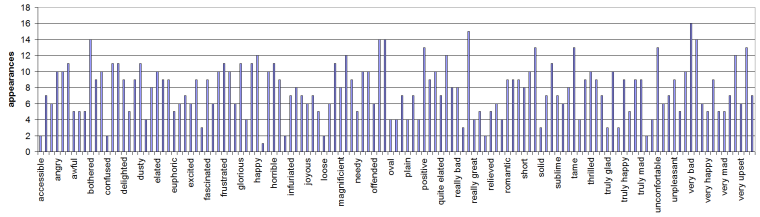

Fig. 1 Appearances of each question

130 possible answers is a CAPTCHA that can be broken 0.76% of the time just by answering any of them, already a bad result for any CAPTCHA.

The answer type distribution is not uniform:, there are 54% answers describing a negative emotion, and 40% describing a positive one (plus 6% describing no emotion, like the answers *accessible, oval, plain, temporary, typical...*). The distribution of their appearance (from the 1000 manually classified ones) seems to be uniform within the different categories, with $58,3\%$, $38,3\%$ and $3,4\%$ for each corresponding category. Surprisingly, it does not seem to be *uniform*, as the p-value of its χ^2_{129} is 0.00365. This can be further exploited for a blind, brute-force attack, although this is not our purpose here, as our attack will produce better results.

4 A Machine Learning Approach to Attack the Civil Rights CAPTCHA

In this section we will introduce our attack, that can be divided in *reading* the Securimage-protected answers, and later, classifying the challenge text. Given the design flaws of the *CRC*, classifying the challenge text is not strictly necessary, but does improve the results. We specify how we use Machine Learning for solving both problems to a level that breaks the *CRC*.

4.1 Classifying the Answers

The current iteration of Securimage might be a good OCR-based CAPTCHA, but the way it is employed in *CRC* makes it quite weak. The problem is Securimage was originally designed to work with a large alphabet, and either random words, or a huge dictionary. If we restrict it to just 130 words, its disguising capabilities might not be good enough for a strong classifier. This is what we decided to test.

Probably an in-depth analysis of the Securimage-produced images would be able to break it more accurately, in this case of only 130 possible answers. We did not want to produce such an attack, but to check if a very basic analysis of the same images, and the use of Machine Learning, would be able to break it.

The statistics we gathered from the images, to feed our AI-classifier, were typical ones: black pixel count, and pixel count per column. This, though, will be affected by the presence of the two black lines (see figure 2).

The two lines drawn are typically of the same thickness all along the length of each one. In this case, a derivative of the number of the vertical pixels in each column will be affected by their presence only at their start and their end (ideally, as intersection of lines and letters will affect too). We also added this statistic.

We decided to sample the image at every column, and then in groups of 3, or 5 columns; and to use a very simple approach - our intent is to do the least possible analysis and let the machine learning algorithm do it for us. Kind of the approach

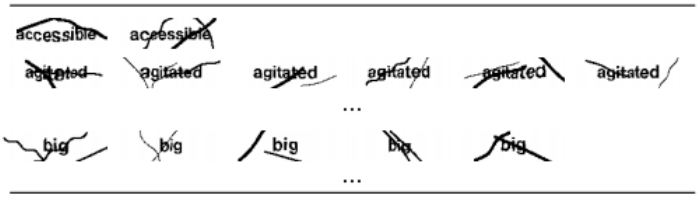

Fig. 2 CRC-Securimage answers

we would expect from a *lazy* hacker wanting to obtain results as fast as possible. So we calculated all the statistics, for each image, and let the different machine learning algorithms cope with the data.

To provide some knowledge for training, we downloaded 1000 answer word-images , and manually classified them into the 130 possible categories.

After feeding them to Weka [7], and using its different classifiers, always using 10-fold CV, we realized that best out-of-box classification was obtained with J48 trees [12]. An astonishing 26.7% of the time we were able to correctly *read* the answer.

4.2 Classifying the Challenge Text

Given the *CRC* design flaws, it is possible to do better. *CRC* presents only 21 different questions, 6 of them positive (29%), and 15 negative (71%). The problem with this is that a *lazy* all-negative classifier will have a 71% success ratio, so we can improve the success of our attack by discarding all read answers that are positive.

This result can be improved. Several projects are available to classify the emotion of a written text [2,11]. The problem with most of them is that they typically classify the emotion of the writer who wrote it by checking the adjectives and/or nouns used. This is not suitable to our case, because the news can be objectively described (no or little use of adjectives), but still be able to create an empathy emotion on the reader (according to the positive or negative impact of it on other people). For our case, one possible approach would be to use the Python *N*atural *L*anguage *T*ool-*K*it (*NTLK*) library [2]. This library provides several algorithms for treating Natural Language problems, some of them to classify text, including decision trees, maximum entropy, SVMs or Naïve Bayes, just to mention some. Among these possibilities, we chose the Naïve Bayes algorithm, which we feel will suit most an attacker looking for a low-cost breach.

To be able to train our model we needed a set of news-bits to be manually classified as either positive or negative. We found two main sources, the Human Rights Watch (HRW) association (`http://www.hrw.org/news`), and the Civil Rights Defenders (CRD) (`http://www.civilrightsdefenders.org/category/news/`).

These last ones happen to be the ones associated with the *CRC*. We downloaded 152 news from the Human Rights Watch association (most of them of negative content), and 622 from the Civil Rights Defenders.

During the set-up of our Bayes classifiers, we were careful to take out of the bags of words the name of any country (and corresponding adjectives), name of the civil & human rights organisations and related organisations, and of course, the NLTK stopwords for English, so they are not used for classification.

Training our model with the HRW association news, we were not able to attain better success in news-type classification. This can be because, from the 152 news excerpts, only 5 are of positive content. Training it with the CRD, we could obtain a 76% success ratio, by improving the classification by an additional correct detection.

As there are only 21 challenges, we can memorize their positive/negative classification. The reason why we do not do it here, and instead try this ML approach, is so we can successfully answer the question "if the number of challenges is raised properly, and actively maintained, could we still successfully attack this CAPTCHA?".

Still there is another, slower but more precise way of classification. Imagine we are presented with a new challenge, and we, as an algorithm, do not know whether the correct answer should be positive, or negative. We can still *read* each answer 27% of the time, and being a low number of them, classify them as either positive or negative. Thus we can keep answering randomly from the answers we read, and when we succeed, by looking at the type of answer that was successful (positive/negative), we can correctly classify the new challenge text.

In brief: adding this questions does not seem to have a huge beneficial impact of the security of this CAPTCHA. The fact that the empathy classification of these questions seems to be quite coarse (positive/negative) means it will not significatively add security to the CAPTCHA.

5 Experimental Results

In this section we explain in detail the implementation and results of our attack. This attack was slightly improved during its development, in what we introduced here as the *improved attack*. We compare the results of both. These two attacks join the results of the previous sections, showing that the use of Machine Learning to exploit the *CRC* design flaws is indeed able to break it.

5.1 Basic Attack

The attack was coded using the Python libraries *urllib, urllib2, cookielib, json* and *nltk*, and also calling Weka for classification of the answer images (*reading* them). While doing so, we saved a log with all the relevant information, for further exam.

Most importantly, we kept track of the answers returned by the *CRC* server as correct ones - passing the CAPTCHA as humans.

The basic attack is quite simple: after downloading the challenge text and images, it classifies the text (neg./pos.) according to our model, classifies the answers using the J48 tree, removes those answers with a sense (pos./neg.) not according to the question. Among the remaining answers, the J48 tree gives to us not only their classification but also a 0-1 index of security of that classification. So the attack chooses randomly one, giving more weight to more securely *read* ones.

5.2 *Improved Attack*

While designing and testing our basic attack, an improvement was tried out. The idea is simple: to remove answers that we know that are not correct for each question (by previous failures). Not only that, if among the answers is present one that we know is correct, then use it. We modified the logic of our attack to accommodate these new ideas, and checked how well it improved our results over time (that is, as our knowledge base of errors and successes was growing).

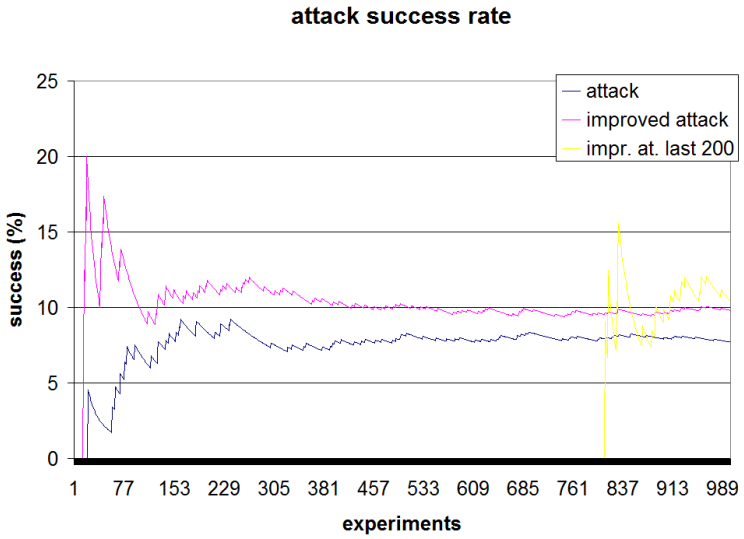

Fig. 3 Percentage of success rate of the attacks

5.3 Results

Due to technical problems, including the large amount of time taken by the *CRC* server to provide a full challenge (45 secs. during our tests, including the three corresponding images), and the low reliability of the same (since the server stopped responding for some minutes on a few occasions), it was easy to incur into timeouts, and difficult to finish a large series of experiments.

With these restrictions, our lengthiest experiments consists of series of merely 1000 challenges for the basic attack (took 17 hours and 54 minutes), and the same for the improved version of the attack (took 14 hours and 55 minutes).

In figure 3 you can see the ratio of success of both attacks as they evolve[1]. The first one is clearly stable at 7.7%, whereas the ratio of the improved version augments with the gathered *knowledge*, with a mean for the total experiment of 9.8%.

If we focus on the last 200 challenges presented to the improved version of our attack (when the *knowledge* has gained some size), figure 3 shows that the success ratio of breaking the *CRC* is 10.5%.

This result is better than a brute-force attack, that would break the *CRC* on average $\frac{1}{130}$ $(0,77\%)$, or $0,71 \times \frac{1}{70}$ $(1,01\%)$ if we restrict ourselves to negative answers. A brute-force attack would not be able to learn the correct answers to each question, as it is not *reading* which answers are present each time. If somehow an attacker creates a DDBB of correct answers to each question, and then uses it to answer a random correct answer, its success ratio would never be over $\frac{\sum_{i=1}^{i=21} \frac{1}{|solutions(i)|}}{21}$, where $solutions(i)$ is the set of all possible correct solutions to question i[2]. In an scenario of a well maintained DDBB of challenges, such an attacker would take extremely long to *learn* all the right answers to all the questions. Our attack will still be able to attain a minimum 7.7% success ratio. Once automatically learned some correct and wrong answers, a 10.5% success ratio or greater would be possible.

6 Conclusions

In this article, we analyze the Civil Rights CAPTCHA from a security standpoint, using Machine Learning algorithms to consistently break it at 10.5% success ratio.

We show how a CAPTCHA, Securimage, is rendered useless by using it out of the scope it was designed for. We also show that the idea of a CAPTCHA based on empathy about other subjects is not necessarily good, especially if this empathy test can only be administered as a choice between two main categories (or a small number of categories). Finally, we have shown that the combination of two CAPTCHAs is

[1] Calculated as $\frac{success}{total}$, so it is more prone to variation with a lower number of experiments.

[2] This would be in a scenario in which the attacker has learned *all* possible right answers to each question. Even in our 1000-length attacks we were not able to learn all the correct answers, with some questions having 5, 7 or 9 correct answers, but others still none.

not always more secure than one of them alone, as the way *Securimage* is used by the *CRC* lowers its security, and in turn allows us to break the *CRC*.

We plan to further improve on this attack by analyzing different statistics and ML algorithms against both the *CRC* and *Securimage*. We also plan on using the same security analysis techniques and ideas to check the security of other HIPs.

Acknowledgements. The first author wants to thank Zhenya for her Mashka i Dashka, i neë krasivaya ulybka.

References

1. Alsuhibany, S.A.: Optimising captcha generation. In: 2011 Sixth International Conference on Availability, Reliability and Security (ARES), pp. 740–745 (August 2011)
2. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly, Beijing (2009)
3. Bursztein, E., Martin, M., Mitchell, J.: Text-based captcha strengths and weaknesses. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, pp. 125–138. ACM, New York (2011)
4. D'Souza, D., Polina, P.C., Yampolskiy, R.V.: Avatar captcha: Telling computers and humans apart via face classification. In: 2012 IEEE International Conference on Electro/Information Technology (EIT), pp. 1–6 (May 2012)
5. Fidas, C.A., Voyiatzis, A.G., Avouris, N.M.: On the necessity of user-friendly captcha. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2011, pp. 2623–2626. ACM, New York (2011)
6. Golle, P.: Machine learning attacks against the asirra captcha. In: Proceedings of the 5th Symposium on Usable Privacy and Security, SOUPS 2009, Mountain View, California, USA, July 15-17. ACM International Conference Proceeding Series. ACM (2009)
7. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update (2009)
8. Kouritzin, M.A., Newton, F., Wu, B.: On random field completely automated public turing test to tell computers and humans apart generation. IEEE Transactions on Image Processing 22(4), 1656–1666 (2013)
9. Mohamed, M., Sachdeva, N., Georgescu, M., Gao, S., Saxena, N., Zhang, C., Kumaraguru, P., van Oorschot, P.C., Chen, W.B.: Three-way dissection of a game-captcha: Automated attacks, relay attacks, and usability. CoRR, abs/1310.1540 (2013)
10. Naor, M.: Verification of a human in the loop or identification via the turing test (1996)
11. Nielsen, F.Å.: A new anew: Evaluation of a word list for sentiment analysis in microblogs. CoRR, abs/1103.2903 (2011)
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
13. Vikram, S., Fan, Y., Gu, G.: Semage: A new image-based two-factor captcha. In: Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC 2011, pp. 237–246. ACM, New York (2011)
14. Warner, O.: Kittenauth (2009), `http://www.thepcspy.com/kittenauth`
15. Yamamoto, T., Suzuki, T., Nishigaki, M.: A proposal of four-panel cartoon captcha. In: 2011 IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 159–166 (March 2011)
16. Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K.: Attacks and design of image recognition captchas. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 187–200. ACM, New York (2010)