ARTICLE IN PRESS

COMPUTERS & SECURITY ■■ (2017) ■■–■■

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis

Carlos Javier Hernández-Castro [a,*], María D. R-Moreno [b], David F. Barrero [b], Stuart Gibson [c]

[a] Complutense University, Madrid, Spain
[b] Alcalá University, Madrid, Spain
[c] University of Kent, Canterbury, UK

ARTICLE INFO

Article history:
Received 8 June 2016
Received in revised form 28 February 2017
Accepted 16 May 2017
Available online

Keywords:
HIP
CAPTCHA
Machine learning
Gender classification
Side-channel attack

ABSTRACT

Human Interactive Proofs (HIPs [1] or CAPTCHAs [2]) have become a first-level security measure on the Internet to avoid automatic attacks or minimize their effects. All the most widespread, successful or interesting CAPTCHA designs put to scrutiny have been successfully broken. Many of these attacks have been side-channel attacks. New designs are proposed to tackle these security problems while improving their human interface. FunCAPTCHA is the first commercial implementation of a gender classification CAPTCHA, with reported improvements in conversion rates. This article finds weaknesses in the security of FunCAPTCHA and uses simple machine learning (ML) analysis to test them. It shows a side-channel attack that leverages these flaws and successfully solves FunCAPTCHA on 90% of occasions without using meaningful image analysis. This simple yet effective security analysis can be applied with minor modifications to other HIP proposals to check if they leak enough information as to allow simple side-channel attacks.

1. Introduction

Free on-line services have become prevalent since the broad deployment of the Internet in the late 90s. The abuse of such services, using automated methods, is the first step towards more sophisticated attacks that can result in substantial revenue for the attackers (as *twitterbots* that push products, people or false news on *Twitter*; automatic voters for different prize-awarding sites; abuse of social networks; abuse of free cloud computing services or on-line ticket selling services, and many others).

Naor (1996) was the first to propose a theoretical security framework based on the idea of using problems that could be solved easily by humans but were thought to be hard for computers. He offered some suggestions for such problems: gender classification, facial expression understanding (happy/sad), filling in words in sentences, etc. A few of his suggestions were subsequently used to create real CAPTCHAs, whilst other pioneers developed alternative designs, all based on theoretically hard-AI problems. Researchers at CMU[3] improved the idea, listing the desirable properties for CAPTCHAs, and presenting their own design (von Ahn et al., 2003).

* *Corresponding author.*
E-mail address: chernandez@ucm.es (C.J. Hernández-Castro).
[1] Human Interaction Proof, or also Human Interactive Proof.
[2] Completely Automated Public Turing test to tell Computers and Humans Apart.
[3] Carnegie Mellon University.
http://dx.doi.org/10.1016/j.cose.2017.05.005
0167-4048/

This was the trend during the 2000s decade that saw the publication of many new techniques enabling the breaking of text-based word-image CAPTCHAs. Some of these techniques relate to one particular CAPTCHA, or family of CAPTCHAs, while others target a broader range. Among the best known are.

In the next decade, there was a lot of research on new techniques, enabling the breaking of text-based CAPTCHAs based on the problem of optical character classification (OCR). These attacks were not based on improvements in the state of the art of OCR. Instead, they made clever use of some very simple properties of the images comprising the challenge set. These properties allowed the attackers to partially reverse the distortions applied to the characters and, in conjunction with other design flaws, gave sufficient information to break these CAPTCHAs, sometimes using very simple techniques as edge detection, flood-fill variants, thresholding, shape-matching, clustering, shrink and fill segmentation, principal component analysis, independent component analysis, horizontal and vertical histograms, skeletonization, erosion and dilation and pattern matching (Mohamed et al., 2013, Zhu et al., 2010, Yan and El Ahmad, 2008b, Wilkins, 2010, Wieser, 2007, Hindle et al., 2008, Yan and El Ahmad, 2008a, Harry "Dark SEO, 2008a, 2008b, Santamarta, 2008, Bursztein et al., 2011). The reaction from various companies was to increase the distortion levels, creating very tough HIPs even for humans (Fidas et al., 2011). Researchers started seeking new ideas for their CAPTCHA designs, looking into other subfields of artificial intelligence (AI), in particular into some of the different problems of computer vision: image classification, object classification and scene interpretation. The most frequently used idea was image classification. Warner proposed selecting photos of kittens to tell computers and humans apart (Warner, 2009). The HumanAuth CAPTCHA asked the user to distinguish between pictures depicting a nature-related scene (e.g. a flower, grass, the sea) and pictures of human-made objects (e.g. a clock, a boat, or Big Ben). Similarly, the creators of ASIRRA (Elson et al., 2007) based it on cat/dog image classification problem, using a large database of "more than 3 million photos". Currently, "No CAPTCHA" reCAPTCHA (by Google) asks the user to pick images related to different categories or to select parts of an image pertaining to a category.

These schemes have been broken, many times through side-channel attacks that did not solve the base problem (Santamarta, 2008, Tam et al., 2008, Hindle et al., 2008, Yan and El Ahmad, 2008a, 2008b, Harry "Dark SEO, 2008b, Hernández-Castro et al., 2010, Hernandez-Castro et al., 2009b, Hernández-Castro et al., 2015, Hernandez-Castro and Ribagorda, 2009a), and in other occasions using small improvements or state-of-the-art ML algorithms (Golle, 2009; Sivakorn et al., 2016). The recent advance in automatic image classification also poses a risk to image-classification CAPTCHAs (Ciregan et al., 2012; Goodfellow et al., 2013; Krizhevsky et al., 2012; LeCun et al., 1998), and although some research has been done on the limits of these algorithms (Goodfellow et al., 2014) and some propose to use them to build new CAPTCHAs (Osadchy et al., 2017), DL[4] can currently be a threat to them. Other proposals have appeared recently and await scrutiny. Some of these are based on different tasks in image classification (Vikram et al., 2011), like artificial vs human face classification (D'Souza et al., 2012). Recent proposals enhance the typical OCR/text-based HIP (Alsuhibany, 2011). Another current trend is to try to analyse different client metrics to detect possible access from bots, self-named as "behavioural", even though what they measure is typically a series of metrics dependent on the platform of the client for client fingerprinting – as *No CAPTCHA reCAPTCHA* by *Google*, NuCAPTCHA, BadBehaviour or Mollom CAPTCHA. These proposals avoid, in some cases, to show a typical CAPTCHA, and use them when there is insufficient data. Other proposals use the same behaviour analysis to vary the difficulty of the CAPTCHA presented to the user. All of these schemes have been so far proprietary. They have the added drawback that these behavioural and client metrics are all taken remotely on the client's machine. These CAPTCHA authors try to protect their measurement algorithms with obfuscation. This paradigm is known as *Security by Obscurity*, and has a long tradition of failure (Anderson, 2002; Hoepman and Jacobs, 2007; Swire, 2004). IT security history suggests that this is not a sound way to go.

During these years, those CAPTCHAs and CAPTCHA proposals that have gotten the interest of the researchers – might it have been because of their originality, their commercial success, or widespread use – and that have been scrutinized have been found vulnerable to attacks. These attacks have been either side-channel attacks or ML attacks that have used the intended path of attack. New CAPTCHAs keep appearing, some of which use original ideas, either applied to known CAPTCHA types, or based on completely new paradigms. Known attacks are typically not applicable to them. Thus, they await scrutiny from the IT security community. It is important to assess their security, and even more, to find ways to assess semi-automatically a basic security level for new CAPTCHA designs.

FunCAPTCHA is an original CAPTCHA implementation that claims better strength and usability than a typical word-classification CAPTCHA. FunCAPTCHA is not the first CAPTCHA design to be based on image orientation and gender classification (Gossweiler et al., 2009, Kim et al., 2010 and Kim et al., 2014 respectively). However, to the best of our knowledge, it is the first readily available gender classification CAPTCHA system in production. The general problem of face identification and interpretation can be considered currently unsolved by ML, as "bad" inputs, like partial pictures of faces from an angle, faces partially covered, faces with different garments (glasses, hats, etc.) or facial expressions are still considered a problem. It is possible to assume a similarly higher difficulty in classifying the gender of faces that are similarly "bad". Furthermore, FunCAPTCHA uses synthetic images, which allows its designers to choose creation parameters that are the most difficult for a ML classifier.

In this article we demonstrate a novel attack against FunCAPTCHA. Our attack does not follow the intended path of attack by a gender classification CAPTCHA like FunCAPTCHA. We neither perform sophisticated image analysis nor use novel ML techniques. Instead, we perform a basic security analysis of it, finding flaws that might enable a side-channel attack. We then proceed to assess their real impact on the Security of FunCAPTCHA using ML.

[4] Deep Learning, or very deep neural networks, that typically include convolutional layers (DCNNs) if they are dealing with images.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

Flaws that allow side-channel attacks are common in new CAPTCHA designs. Many CAPTCHA designers, following the paradigm introduced by Naor and Von Ahn (Naor, 1996; von Ahn et al., 2003), have presented proposals based on problems that they consider to be AI-hard. As most AI problems have not been formalized, we have yet to prove that these problems are AI-hard. Worse, a CAPTCHA by design presents a subset of what can be considered the full problem. This subset might not be representative of the whole problem. In particular, it might not be as hard for AI. It might significantly alter the proportion of weak, easy to solve challenges. It might discard the most complex examples. If any of these are the cases, it might fall to side-channel attacks that do not deal at all with the base problem. Thus, this type of analysis is important for new CAPTCHA designs. We propose here a simple way of doing it and show its results for FunCAPTCHA.

In the following sections, we introduce FunCAPTCHA and then proceed to discuss its possible design flaws (Section 3). In Section 5 we test how different well-known ML algorithms can be used to exploit these flaws. In Section 6 we check if our previous positive results can be turned into a novel attack. We proceed to implement it and analyse its performance (Section 7), showing that it passes FunCAPTCHA on 90% of occasions. We then discuss some possible countermeasures to our attack and similar attacks (Section 8). We finish by presenting our conclusions.

2. FunCAPTCHA

FunCAPTCHA tests are of two different main types, each one appearing roughly 50% of the time. The first type requires the user to rotate in 40° increments an image until it is in its correct vertical orientation. This idea is not new (Gossweiler et al., 2009) and has known drawbacks (Zhu et al., 2010) that make it of little interest.

The second type of challenges is a gender classification challenge that requires the user to select a picture of a female face among 8 pictures and drag and drop it to the centre of a 9×9 tile box. Because of its novelty, this is the test that interests us and that we will study in this article.

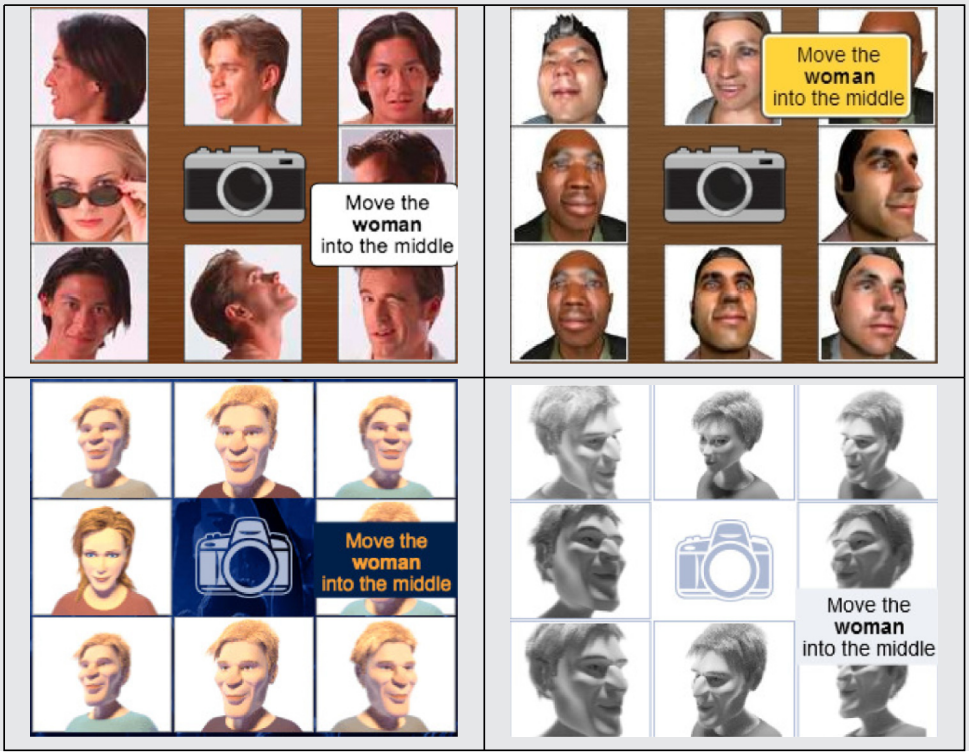
Each of these two types of CAPTCHA has subtypes depending on how many tests are performed sequentially to pass the CAPTCHA. In our tests, the whole CAPTCHA challenges have consisted of either 1, 3 or 5 tests.

FunCAPTCHA has implemented different versions of the gender classification test over time (Table 1). We are aware of at least four different versions: using real human models, rendering different 3D facial models in 2D in colour, using only one model per gender in colour, and rendering them in greyscale. Why the FunCAPTCHA designers did these changes is uncertain.

Apart from its security, another strong point according to FunCAPTCHA marketing is that it offers a significantly higher conversion rate than other CAPTCHAs, as "FunCaptcha has a 96% completion rate" and "is completed 28% more than twisty-lettered CAPTCHAs".

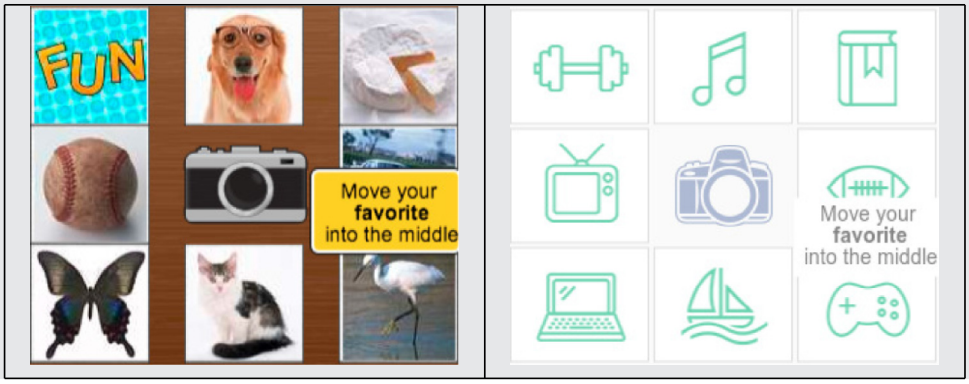
FunCAPTCHA presents a last fake test in which it asks the user to move the icon representing its favourite activity to the centre (possibilities being TV, sports, etc.). An example is shown in Table 2. The user's answer does not affect the outcome of the challenge. From an IT security standpoint, we do not understand the value of this question. After contacting FunCAPTCHA designers, they comment that this test is not render for

Table 1 – Different FunCAPTCHA gender classification iteratens.



Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

Table 2 – Different FunCAPTCHA iteratens of the *fake* test.



security reasons, but for UI reasons: "[it is] better to give the user something to do rather than make them look at a loading bar or spinner".

3. FunCAPTCHA design analysis

We analysed FunCAPTCHA from the viewpoint of an attacker that wants to bypass it as a means to gain access to some online service. Thus, we did not register with the API of FunCAPTCHA, nor installed a client in our machines. We analysed its protocol directly from the browser, using HTTP analysis tools.

3.1. *FunCAPTCHA initial analysis*

The most basic challenge of FunCAPTCHA consists of a single gender classification test. It is correct to argue that a challenge with a 12.5% brute-force success rate ($\frac{1}{8}$) is weak (0.6% is enough to consider a CAPTCHA broken (Zhu et al., 2010), even though we do not know in what cases FunCAPTCHA decides that this simple challenge is enough. The chances of passing the 3- and 5-test challenge by brute-force would be 1.5% and 0.003% respectively. Only the 5-test challenge has a strength against brute-force attacks good enough for a production CAPTCHA. We do not know whether the shorter tests are just for demonstration purposes, so even though we will analyze all cases, we will focus on the results of the 5-test challenge. Given that the typical time to complete it is around 15 s, we do not think that a version of FunCAPTCHA with more tests per challenge would have the same conversion rates.

It seems that FunCAPTCHA relies on some tracking, maybe based on IP tracking, to *harden* the test after a user sends a wrong answer. FunCAPTCHA authors later confirmed this extreme. They claim to use "other ways to trace a user's identity besides IP", and that because of this, the 1-test challenge does not pose a risk, as a bot (with low accuracy) would be identified.

During our analysis, we found that FunCAPTCHA is using several obfuscation techniques to try to prevent its analysis. Some of these techniques are JavaScript code obfuscation at two levels, cyphering its communications using the AES[5] cypher in Counter-mode for the transmission of some values (in addition to using HTTPS), obfuscating the order in which it deploys the face images on the client's browser and using 2-level of cross-domain IFrame nesting to prevent easy JavaScript debugging. Each of these measures was rendered at least partially useless after the following findings:

1. It was possible to partially revert JavaScript code obfuscation, as a different JavaScript code was found thanks to caches and using a smaller version name.
2. FunCAPTCHA uses the AES library from Chris Veness (which can be found at http://www.movable-type.co.uk/scripts/aes.html). Even though this AES library is protected by a MIT license and requests a link to the original page and the original copyright notice, we were not able to find those in FunCAPTCHA's site. Thanks to this finding, it was possible to easily decipher its communications. In particular, it was possible to see that the value of the form attribute *guess* was being used to send back the answers to FunCAPTCHA after each drag and drop. Its value was cyphered using AES in Counter mode, initialized with a value partially time dependent and partially pseudo-random. This value was transmitted with the cyphered answer to allow for its decyphering at the FunCAPTCHA server. The key used for cyphering was the *session_token*, passed from the FunCAPTCHA server to the client during the initial set-up of the test.
3. The other three obfuscation measures were all bypassed by using a real browser to analyse and later bypass the CAPTCHA. More details about this in Section 6.

The reasoning for some of these obfuscation measures is not clear. As an example, encoding the answers using AES with a key already delivered from the server does not increase its security. Other measures are more of a nuisance to some analysis that a real impediment to any attacker. After contacting FunCAPTCHA designers, they claim that "obfuscation is asymmetrical effort" that is "effective at slowing down the progress of attackers". We think this is at least controversial, especially

[5] Advanced Encryption Standard, or Rijndael, an algorithm and specification for the encryption of electronic published by US NIST.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

given browser automation tools and open-source browsers. Our attack, that can be easily modified to bypass the obfuscation, is an example of this.

3.2. *FunCAPTCHA image repository*

Soon we knew enough of the client–server communications protocol as to download several hundreds of images from FunCAPTCHA. After downloading 500 images, we calculated the MD5[6] and SHA1[7] Cryptographic hash functions for all the images downloaded, using them as a simple fingerprint of the file contents. We found 0 coincidences. The fact that there is no real image repetition was somehow surprising, as many of the faces look quite similar to the eye.

This leads us to affirm that FunCAPTCHA does render the 3D model each time using different parameters: different angle, illumination and distance (field of view), so that no two images are identical at the bit level. This initially looks as a sound implementation decision for FunCAPTCHA.

3.3. *FunCAPTCHA protocol analysis*

Even though FunCAPTCHA uses several obfuscation mechanisms, it was possible to relate its client–server communications to the different events happening at the browser. The communications cyphered with AES were easily deciphered. We were able to easily analyse the FunCAPTCHA communications protocol.

This allowed us to know whether a particular challenge was of the orientation type or of the gender classification type (i.e. if the value of the parameter *challengeURL* was 001 or 002 respectively), to know how to request the different components of a challenge to the server (POST petition at https://funCAPTCHA.co/fc/gfct/ including the variables *token* and *sid* among other sent before), how to send our answer to the server (POST to https://funCAPTCHA.co/fc/ca/ sending the variable *guess* encoded with AES), and how to know when such answer is complete (all tests were answered) and correct (*"response": "answered", "solved": true, …*).

It is important to note that, when the answer is incorrect, the server returns:

```
{"response":"answered","solved":false,"incorrect_guess":4,"score":3}'
```

Where *incorrect_guess* is the ordinal of the answer that was wrong. This is not a sound idea, as it allows an attacker to know which tests within a full challenge have been correct, and thus, to correctly label a subset of the images of the challenge and gain knowledge for other attacks. This makes it easier for an attacker create a labelled training set.

Using a proxy, we were able to programmatically intercept the communications between the client (browser) and the server. This allowed us to determine what type of challenge we were facing – rotation or gender classification – and also how many tests it contained. When we were dealing with a gender classification task, we were also able to download the challenge images. Finally, it allowed us to easily know whether the answers sent to FunCAPTCHA were correct or not according to their servers.

4. FunCAPTCHA design flaws

At this point of our research, we could list some decisions of the FunCAPTCHA design that might be key to its security:

1. It uses only one male and one female 3D model.
2. The model does not show facial expressions nor other distortions, as the addition of glasses, different haircuts, etc.
3. Even though the served 2D images do not repeat at the bit level, some of them *look* similar or very similar to images shown before.
4. The background is always plain white.
5. The images do not have the same distance from the model. For example, there are images that include the shoulders, others show the neck partially, and others show mostly only the face. The field of view changes from image to image.

A common mistake in novel CAPTCHA designs is that the problem presented by the CAPTCHA is not as strong as the AI problem in which it is based (Hindle et al., 2008, Yan and El Ahmad, 2008a, 2008b, Harry "Dark SEO, 2008b, Hernández-Castro et al., 2010, Hernandez-Castro et al., 2009b, 2011, 2014). Just by analysing the raw tests and challenges of a CAPTCHA, it is difficult to know whether they have been carefully selected or crafted to be hard against ML or not. We wanted to know if this might be the case with FunCAPTCHA, and we wanted to check it in a way that might be extensible to other CAPTCHAs.

Even given the limited number of models and rendering parameters, it is true that visually, there is no apparent way to algorithmically classify male from female pictures. The number of white pixels is more affected by the distance than any other factor. The histogram of the use of the different grey shades used does not seem significantly different in both cases. The head of the male model is larger than the head of the female model, but given that the distance and field of view changes randomly, a histogram of white pixels or shades of grey will probably not be relevant for classification. This seems to be the result of some thought put into the CAPTCHA design and analysis, and possibly this pushed the evolution of FunCAPTCHA through time.

To check whether this is the case, we employed a simple classifier with the aim of distinguishing male faces from female faces. We wanted to check whether the similarities of the FunCAPTCHA images would allow a classifier to efficiently distinguish male vs. female images if fed with very simple statistics from the images.

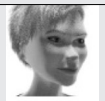
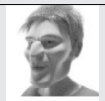
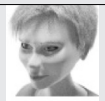
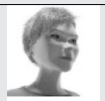
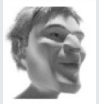
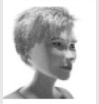
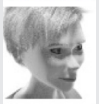
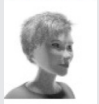
To test this hypothesis, we downloaded and manually classified 4320 images from FunCAPTCHA. Note that this was not strictly necessary, it would have been possible to solve the 1-test challenges with $\frac{1}{8} = 12\%$, and use these solved challenges as a training set. Yet during our initial experiments with FunCAPTCHA, it was only serving the 3-test and 5-test challenges, which would have made it quite slow for us to get the training set size we wanted. Of those 4320 images, only 535

[6] Message Digest 5, a widely used hash function producing a 128-bit hash value.

[7] Secure Hash Algorithm 1, a 160-bit cryptographic hash function designed by the US NSA and published by the US NIST.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

Table 3 – Some FunCAPTCHA faces, both wrongly and correctly classified, and their associated metrics.

	Wrongly classified		Correctly classified	
problem (test)				
white pix. (%)				
color histogram (5 bars)				
color histogram (15 bars)				
color histogram (25 bars)				
JPEG sizes				
class (training)				
white pix. (%)				
color histogram (5 bars)				
color histogram (15 bars)				
color histogram (25 bars)				
JPEG sizes				
diff. img. (×5)				

were images of females (not exactly 1 in 8 due to some timeouts during the downloads). We used these images to extract some very basic statistical information from them: the percentage of white pixels; the histograms of the use of different grey intensities, in groups of 5, 10, 15 and 25 intervals; and the size of the image compressed with JPEG[8] using different quality factors (from *quality* = 0 to 100).

Initially, we decided to use the *k*-nearest neighbours algorithm. *k*NN has little parametrization: the number of neighbours searched, how the weights are calculated regarding the distances and the algorithm to use for the search. *k*NN is a good representation of the idea of classifying by finding similarities between examples. It can also show the previous known examples that are found to be *similar* to the one being classified. This allows checking if the metrics and distances are relevant for the classification we are trying to achieve.

We *trained* *k*NN using all the manually classified images. To test it, we downloaded an additional 148 challenges, each one composed of 5 tests (except of a few download errors). We proceeded with a semi-exhaustive search trying different values for *k* and the rest of the parameters. We ordered the results by their Cohen's κ statistic values. We chose this statistic as it measures a classifier against the expected accuracy, which is more relevant for such an unbalanced training data than just the accuracy. The best result was typically obtained selecting only the closest neighbour, reaching an accuracy of 97% and a κ statistic of 0.84 when tested on new images.

We run our experiment again to show the closest image to each unknown image. The result of this experiment can be partially seen in Table 3. In this table, the first six rows represent a test image and the value of its different metrics, and the next six rows are the training image representing the corresponding *class* (i.e. the closest image to the test one) and the same metrics for that training image. The metrics, in order of appearance from higher to lower, are: the number of white pixels (% from maximum), histogram of grey-scales used divided in 5 bins, 15 and 25 bins, and the sizes of the image compressed with JPEG and different quality settings. Table 3 shows two wrongly and two correctly classified images, and the closest one to each query.

5. Exploit of FunCAPTCHA design flaws using machine learning

*k*NN is possibly the simplest ML algorithm. We wondered whether other ML algorithms, using the same metrics, would

[8] Joint Photographic Experts Group, which created a standard and method for lossy compression of images.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

Table 4 – Best and worst classifiers for off-line gender classification with FunCAPTCHA.

Algorithm	Weka class name	Correct (%)	κ statistic
A multilayer NN that uses backpropagation to classify instances. All the nodes in the network are sigmoid.	MultilayerPerceptron	99.19	0.96
K* is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it. It uses an entropy-based distance function (Cleary and Trigg, 1995).	KStar	98.94	0.95
Nearest-neighbour classifier: uses normalized Euclidean distance to find the training instance closest to the given test instance.	IB1	98.91	0.95
K-nearest neighbours classifier: selects the appropriate value of *K* based on cross-validation (Aha and Kibler, 1991).	IBk	98.91	0.95
Logistic Model Trees: classification trees with logistic regression functions at the leaves (Landwehr et al., 2005).	LMT	97.73	0.89
Multinomial logistic regression model with a ridge estimator (le Cessie and van Houwelingen, 1992).	Logistic	97.59	0.89
Linear logistic regression, with LogitBoost for fitting the logistic models (Landwehr et al., 2005).	SimpleLogistic	97.43	0.88
Functional trees: classification trees that could have logistic regression functions at the inner nodes and/or leaves (Gama, 2004).	FT	97.36	0.88
Stochastic variant of the Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) (Shalev-Shwartz et al., 2007).	SPegasos	97.43	0.88
John Platt's sequential minimal optimization algorithm for training a support vector classifier (Keerthi et al., 2001).	SMO	96.83	0.84
Forest of random trees (Breiman, 2001).	RandomForest	96.74	0.83
Class is binarized and one regression model is built for each class value (Frank et al., 1998).	ClassificationViaRegression	96.41	0.83
...			
Voted perceptron algorithm by Freund and Schapire.	VotedPerceptron	88.63	0.13
Normalized Gaussian radial basis function network: uses the k-means clustering algorithm to provide the basis functions and learns a logistic regression on top of that.	RBFNetwork	88.17	0.13
Locally weighted learning: an instance-based algorithm to assign instance weights. Uses naive Bayes for classification.	LWL	87.75	0.03
Meta-classifier that uses a clusterer for classification, like simple k-Means.	ClassificationViaClustering	55.56	0.01
Discriminative Multinomial Naive Bayes classifier.	DMNBtext	87.71	0.01
Bayesian Logistic Regression for both Gaussian and Laplace Priors.	BayesianLogisticRegression	.	0
Uses some base classifiers that are "graded".	Grading	87.64	0
MultiBoosting can be viewed as combining AdaBoost with wagging.	MultiBoostAB	87.64	0
Selects the best ZeroR classifier (can select others, but this is the default).	MultiScheme	87.64	0
Implements a single conjunctive ("and") rule learner.	ConjunctiveRule	87.64	0
Class for building and using a 0-R classifier. Predicts the mode.	ZeroR	87.64	0
Decision stump, classifies based on entropy.	DecisionStump	87.64	0

cope better with the problem proposed by FunCAPTCHA. To try other algorithms, we checked the use of different ML frameworks that allow the use of several ML classifiers and have some integration with *Python*. In particular, we looked at Orange and Weka (Hall et al., 2014). We decided to use Weka because of the many more classifiers that Weka has out-of-the-box (79 vs. 11 in Orange).

We compared all compatible Weka classifiers with their default parameters, testing them using 5-CV. The selection of the best-performing algorithms was done using the κ metric, as the classifiers have to be effective with a heavily unbalanced training and test set. It is worth to note this unbalanced training set caused some problems to several classification algorithms that went along with the *they are all males* classification. Other classifiers were much better at coping with it. The results of these tests are available in Table 4. This table shows the best and worst 12 performers of the whole set. It turned out that the multilayer perceptron, IB1/k, KStar, and tree-based algorithms are the ones that perform best.

6. Machine learning attack to the FunCAPTCHA

The effectiveness of the ML classifiers for bypassing the different challenges presented by FunCAPTCHA was determined and hence the strength of the design was assessed.

For that purpose, we created an attack that comprises the following steps:

1. Start a local proxy for the HTTP and HTTPS protocols. We use the *proxpy* Open-Source proxy (available at https://code.google.com/p/proxpy/).
2. Open a web browser (Mozilla FireFox) and direct it to the web-page at https://www.funCAPTCHA.com/contact-us/. We control this browser instance thanks to the Selenium library. This web-page contains the FunCAPTCHA CAPTCHA at its bottom. We decided not to use the web-page at https://www.funCAPTCHA.com/demo/ because of its frequent changes during our analysis, including a period of over a

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

month during which it did not provide any challenge demonstration.

3. We wait for the proxy to capture the value of the *challenge_url* variable that indicates if we are facing an image orientation challenge or a gender classification one.
   (a) If we are served an image orientation challenge (*challenge_url* = 001), we restart the process, unless we have done it 2 times already, in which case we wait a random time in between 25 and 115 s.
   (b) If we are served a gender classification challenge (*challenge_url* = 002), we read how many images it is composed of by looking at the contents of the array variable *image_urls_str* in the page content.
4. We wait till all images are downloaded by the browser.
5. A classifier is run over the sets of 8 images (1, 3 or 5 sets or tests). We use the Weka ML framework (Hall et al., 2014) and the previously trained models in Section 5. We check that for each set, one and only one image is classified as a woman.
   (a) If the classifier fails to do so, not classifying one and exactly one as a woman each 8 images, then the challenge is declared failed. This case is counted both as a classification failure and an attack failure. A log is saved, and the process starts again. Note that it is possible to take more intelligent options here in order to increase the success rate of the attack.
   (b) If the classifier classifies one and only one image of each set as a woman, we proceed to the next step in order to send the answers to the server.
6. Send the answers to each classification test of the challenge:
   (a) We locate the solution face on the screen using the SWIFT algorithm as implemented in the OpenCV library.
   (b) We drag and drop the face to the centre of the challenge using the *pyautogui* library (at GitHub (https://github.com/asweigart/pyautogui).
   (c) We wait for the answer from the FunCAPTCHA server. It could be:

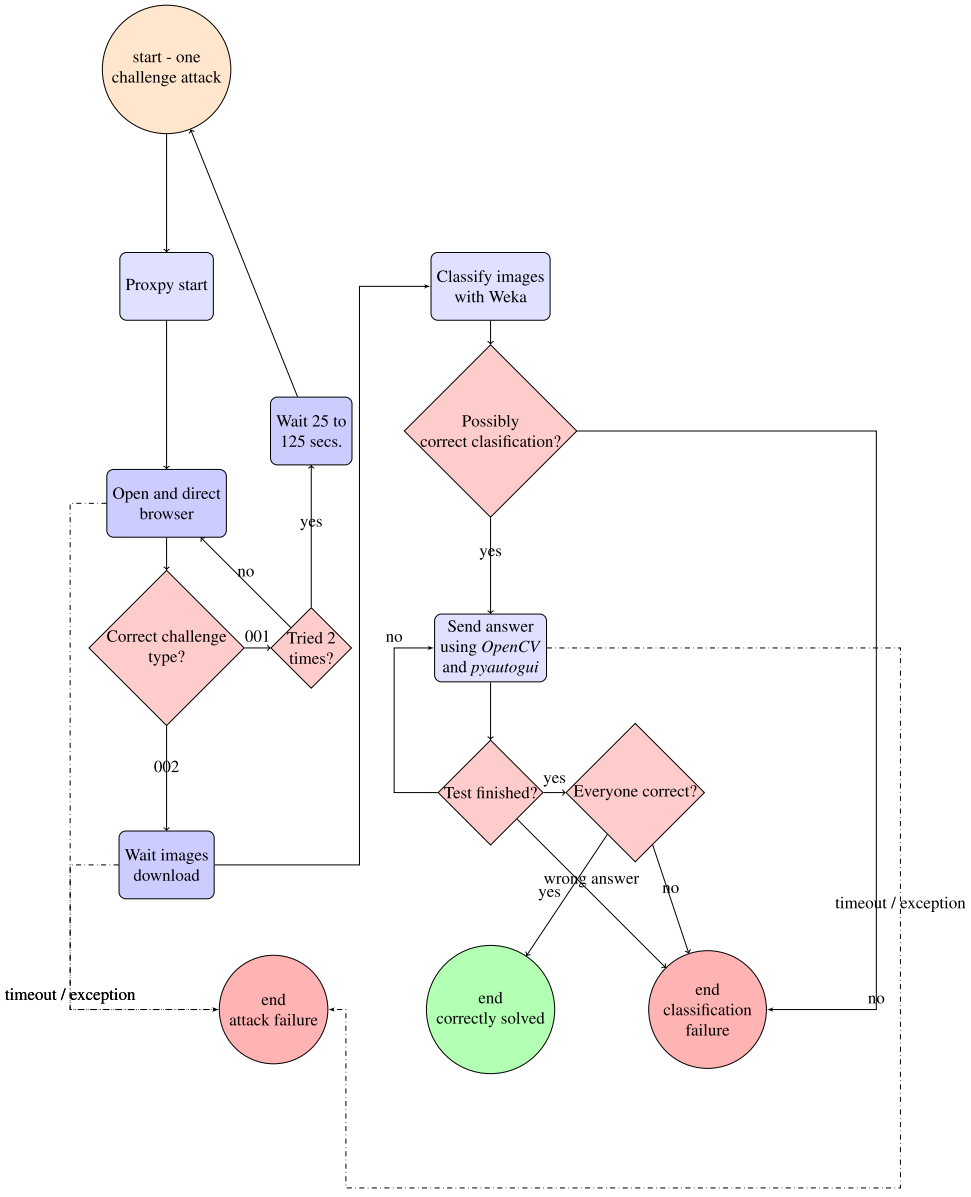


Fig. 1 – Flow chart of the attack.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

- "not solved": we proceed to send the next answer.
- "solved:false": we log the challenge as failed (both for the attack and the classifier).
- "solved:true": we log the challenge as correct.

Fig. 1 shows a summarized flow chart of this attack. All steps of the attack had a set time-out that, when reached, would declare that challenge as failed and restart it.

7. Attack results

We ran our attack for the classifiers that performed best on our off-line classification test, and also with the original *k*NN implementation. The behaviour of FunCAPTCHA adapted to the success rate of our attack. In those cases when the attack kept solving correctly the gender classification challenges composed of only 1 test, FunCAPTCHA almost never served us the more difficult 3-test or 5-test challenges. For this reason, we ran each experiment in two versions for each one of ML algorithms used:

- the regular one already described (*basic*), in which we try to solve all gender classification tests presented to us by FunCAPTCHA, and
- the *difficult* one, in which we randomly answered all 1-test challenges thus failing most of them, in order to receive more 3- and 5-test challenges.

With the many restrictions on speed as not to overload the servers, our lengthiest experiments consisted of various series of around 255 full challenges for each one of our experiments. The 255 challenges were seldom reached, as we frequently run into time-outs, errors downloading information, or with the on-screen iteration.

Table 5 presents the success rate of the attacks to FunCAPTCHA by different classification algorithms. This table is relevant to know the success rate of our attack in the current FunCAPTCHA implementation. In this table, the first column is the Weka classifier name, the second column shows the classifier accuracy during the attack, and the third column shows the success rate of the attack itself.

The classifier accuracy during the attack is measured per full CAPTCHA challenge, independently of if they are 1-, 3- or 5-test challenges.

FunCAPTCHA is typically going to serve to us more 3- or 5-test challenges the more 1-test challenges we fail. Then a slightly worse classification rate in 1-test challenges triggers a feedback mechanism that can have a major effect on the classification accuracy rate, as it is here measured by full CAPTCHA challenge.

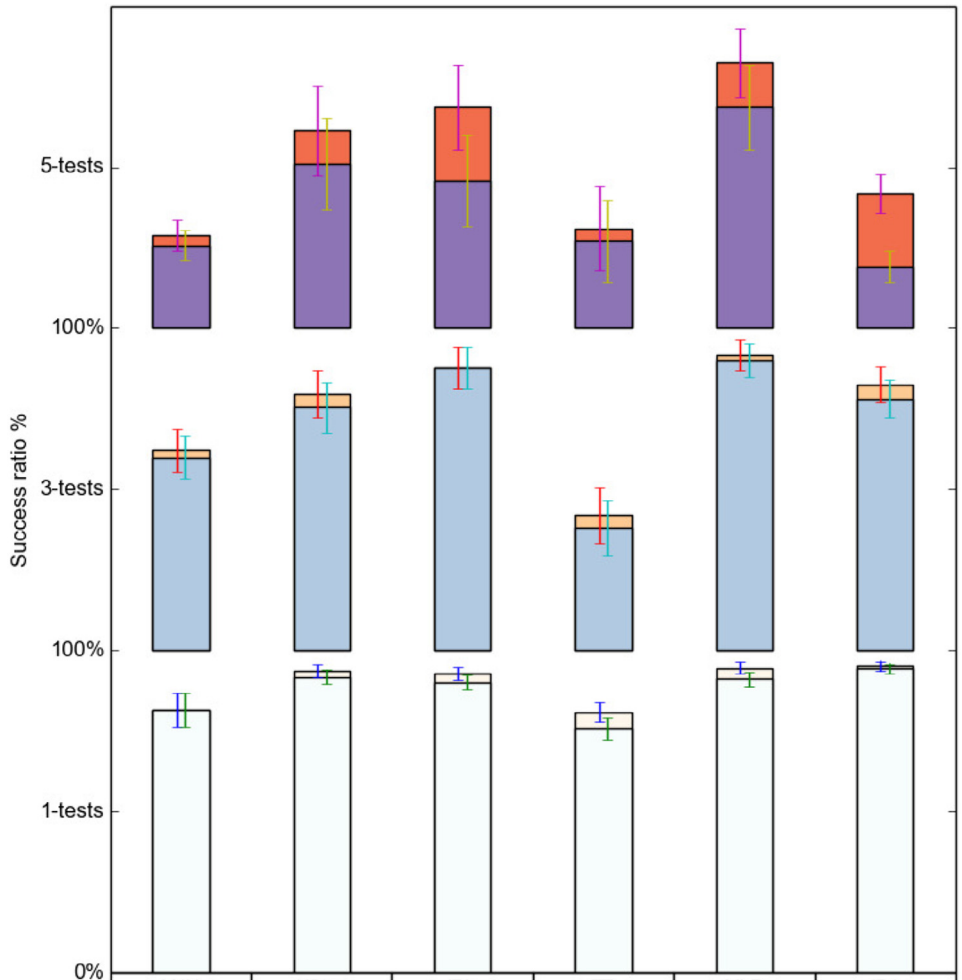
The second half of Table 5 answers the question "what would be the success rate of our attack if FunCAPTCHA used only the harder 3-test or 5-test challenges?". It presents the success rate of the attacks to FunCAPTCHA by different classification algorithms. We can see that *MultilayerPerceptron* and *IBk* are among the top overall performers. We can also see that the difference in success rate between classifier and attack is higher than in the first half of the table, as each challenge now involves more communications with the server and thus is more prone to errors.

Fig. 2 shows a combined result of both attacks, summing the results obtained during both the *basic* and *difficult* settings in order to obtain more 3- and 5-test challenges. The bars indicate the success in a scale from 0% to 100% for each subtype. Each bar is divided in two: the classifier success identifying the correct one and only one woman in each of the *n* groups of 8 images for the whole challenge, and the attack success for the whole *n*-test challenge. Along with each bar, we show the confidence interval, estimated for a binomial distribution using the Wald method. The multi-layer perceptron can solve 94.53% of the 1-test challenges, 91.23% of the 3-test challenges and 82.05% of the full 5-test challenges (68.09% attack success). This

Table 5 – Attack results by classifier, for the basic and difficult attack. This table shows the success rate of our attack against the current FunCAPTCHA implementation (basic attack) and answers the question of how successful would our attack be if FunCAPTCHA were to use only their harder challenges (difficult attack).

Basic attack					
Classifier	% Classifier	% Attack	Number of *n*-test challenges		
			1	3	5
IB1	94.02 ± 0.02	90.42 ± 0.03	448	58	0
KStar	93.15 ± 0.03	89.19 ± 0.04	252	1	0
IBk	92.61 ± 0.03	88.15 ± 0.04	264	98	0
MultilayerPerceptron	94.68 ± 0.03	85.27 ± 0.04	266	6	1
Logistic	77.3 ± 0.05	76.05 ± 0.05	248	51	9
FT	80.59 ± 0.05	72.9 ± 0.05	251	2	0
kNN	55.65 ± 0.06	54.07 ± 0.06	70	69	107
Difficult attack					
MultilayerPerceptron	88.35 ± 0.03	82.69 ± 0.04		110	46
IBk	83.07 ± 0.04	72.97 ± 0.05		98	48
KStar	75.83 ± 0.05	62.43 ± 0.05		116	47
IB1	53.63 ± 0.04	29.35 ± 0.04		72	255
FT	38.70 ± 0.05	28.98 ± 0.05		125	48
kNN	36.80 ± 0.05	23.71 ± 0.04		72	119
Logistic	24.18 ± 0.03	18.20 ± 0.03		236	132

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005



	kNN	KStar	IBk	FT	MLP	IB1
1-tests	81.4 (81.4)	93.5 (91.7)	92.6 (90.2)	80.9 (75.7)	94.5 (91.0)	95.0 (94.2)
3-tests	62.1 (60.0)	79.5 (75.4)	87.8 (87.8)	41.9 (38.0)	91.7 (90.2)	82.6 (78.2)
5-tests	28.9 (25.8)	61.5 (51.1)	68.8 (45.8)	31.0 (27.1)	82.5 (68.8)	41.9 (19.2)

Fig. 2 – Success rate by classifier and challenge type, for both the *basic* and *difficult* attack. Each column corresponds to one classifier. There are three bars per classifier, one per type of challenge (1, 3 and 5-tests). These bars are subdivided each in classification accuracy (for 8-images tests) and attack success rate (lower, as it includes any additional problem during the attack). They show the corresponding confidence interval at 95%. The table below shows the same information numerically. The numbers are the classification success for the whole (1/3/5) 8-images tests, and the numbers between parenthesis are the attack success rate for the same challenges.

means that even if FunCAPTCHA designers decided now to use only their most secure 5-test challenges, this attack would break their CAPTCHA 68.09% of the time.

8. Proposed improvements

In this section we will discuss some possible improvements to the FunCAPTCHA both in general and against this particular attack.

8.1. *Answer space*

In general, FunCAPTCHA should never serve 1-test or even 3-test challenges, only challenges composed of 5-tests. 5-tests challenges are the only viable option to make it resilient to brute force attack, also preventing the attacker from easily obtaining automatically labelled images using FunCAPTCHA as an oracle.

As our attack can break the 1-test challenge 91% of the time, if we want to reach a success rate lower than 0.4% (Zhu et al., 2010), we would need to repeat this test $log_{0.91}(0.004) = 58.54$ times.

If FunCAPTCHA authors add more possible answers, as our best classifier is able to correctly differentiate the gender 99.19% of the time, that is, correctly solve a 8-image test $99.19^8 = 93.7\%$ of the time (actually it is 94.5%, but here we are extrapolating using our off-line results), we would need to have $log_{0.9919}(0.004) = 678$ faces among which to pick one female. That seems a little bit too much from a usability point of view.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

8.2. *ML analysis*

There are some ways to try to prevent the ML attacks that we have presented here. An obvious one would be to use a much larger number of models. This would allow for a bigger chance of *collision* of the metrics used. The models themselves can be studied using the commented ML algorithms to discard those that are too easily classified automatically.

Model rendering parameters could also be wider. It might be possible that there exists a sweet spot in the rendering parameters (angle, light, etc.) in which ML classification does not perform well while human classification still performs well due to a number of reasons (clues about hair, etc.).

It is also possible to include measures to distort or homogenize the result of basic statistics from the images (i.e. histogram of grey scales). The aim would be to render the most common and/or trivial statistics completely useless for ML classification.

8.3. *Resilience*

Nothing prevents the authors of FunCAPTCHA from having new models in their reserve to make A/B tests, either in general or against a particular client This strategy would allow not only to automatically detect attacks but to reply to them in real time.

If a large-enough number of models is present, this could mean that in reality the CAPTCHA would be able to detect and adapt to attack scenarios.

By everything mentioned above, it is unclear to us at this point whether these measures would render FunCAPTCHA secure against this kind of attack. After a new redesign, a full new security analysis should be done. Even if the redesign can cope with this attack and variants of it, it is certainly unclear whether this subset of the gender classification problem would be secure against the recent advances in image classification, more precisely Deep Convolutional Neural Networks (Ciregan et al., 2012; Krizhevsky et al., 2012; LeCun et al., 1998).

9. Conclusions

In this paper, we analyse the security of a production CAPTCHA called FunCAPTCHA. It is the first commercial proposal to our knowledge to implement the idea of gender classification as the basic way to tell computers and humans apart. We analyse its current implementation as of July to October 2015. The authors of FunCAPTCHA claim it to be broadly used, never broken, and with a high security level.

We analyse its security and find possible flaws in its design. These weaknesses give a hint that the problem posed by the CAPTCHA designers might be a small subset of the general problem, not carefully chosen, and not representative.

To analyse whether this is the case, we use a general method based on extracting simple and generic metrics and analyse then using ML algorithms to try to find correlations between the challenges and their correct answers.

We see that the particular problem proposed by FunCAPTCHA can be solved. We confirm this through an attack that can bypass FunCAPTCHA 90% of times. Even if the authors of FunCAPTCHA would only use their most difficult 5-test challenges, this attack would be able to pass it at least 68% of the time. Our attack does not solve the general gender classification problem but exploits design weaknesses.

This is an unexpected result given the apparent complexity of the base AI problem (at least when using the most demanding examples) and the simple attack methods used. We thus find the security of their CAPTCHA to be unexpectedly low. We present some possible ways to partially solve these design flaws.

The security analysis we present is very generic and can be applied to other CAPTCHA designs with minor variations. It constitutes a low-cost way to check for flaws that would allow side-channel attacks against them. We plan on using similar security analysis techniques and ideas to test the security of other HIPs.

Acknowledgements

The work is supported by the Universidad de Alcalá project 2016/00351/001, and MINECO project Epheme-CH TIN2014-56494-C4-4-P.

The first author wants to thank, in no particular order, Julio H.O., Declan J. H.C., Julio C. H.C., Женя C., Carmen C.V. and Лена C. for their contributions and support.

REFERENCES

Aha D, Kibler D. Instance-based learning algorithms. Mach Learn 1991;6:37–66.

Alsuhibany SA. Optimising CAPTCHA generation. In Availability, Reliability and Security (ARES), 2011 Sixth International Conference on, pages 740–745; 2011. doi:10.1109/ARES.2011.114.

Anderson R. Security in open versus closed systems – the dance of Boltzmann, Coase and Moore. at Open Source Software Economics, 2002.

Breiman L. Random forests. Mach Learn 2001;45(1):5–32.

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 '11, pages 125–138, New York, NY, USA; 2011. ACM. ISBN 978-1-4503-0948-6. http://doi.acm.org/10.1145/2046707.2046724.

Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3642–3649. IEEE, 2012.

Cleary JG, Trigg LE. K*: an instance-based learner using an entropic distance measure. In 12th International Conference on Machine Learning, pages 108–114, 1995.

D'Souza D, Polina PC, Yampolskiy RV. Avatar CAPTCHA: telling computers and humans apart via face classification. In Electro/Information Technology (EIT), 2012 IEEE International Conference on, pages 1–6, 2012. doi:10.1109/EIT.2012.6220734.

Elson J, Douceur JR, Howell J, Saul J. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, pages 366–374, New York, NY, USA; 2007. ISBN 9781595937032. http://dx.doi.org/10.1145/1315245.1315291.

Fidas CA, Voyiatzis AG, Avouris NM. On the necessity of user-friendly CAPTCHA. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, pages 2623–2626, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. http://doi.acm.org/10.1145/1978942.1979325.

Frank E, Wang Y, Inglis S, Holmes G, Witten IH. Using model trees for classification. Mach Learn 1998;32(1):63–76.

Gama J. Functional trees. Mach Learn 2004;55(3):219–50.

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, 2009, ACM International Conference Proceeding Series. ACM, 2009. ISBN 978-1-60558-736-3.

Goodfellow IJ, Bulatov Y, Ibarz J, Arnoud S, Shet VD. Multi-digit number recognition from street view imagery using deep convolutional neural networks. CoRR, abs/1312.6082; 2013. Available from: http://arxiv.org/abs/1312.6082.

Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572; 2014.

Gossweiler R, Kamvar M, Baluja S. What's up CAPTCHA?: a CAPTCHA based on image orientation. In Proceedings of the 18th International Conference on World Wide Web, WWW '09, pages 841–850, New York, NY, USA; 2009. ACM. ISBN 978-1-60558-487-4. http://doi.acm.org/10.1145/1526709.1526822.

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The Weka data mining software: an update. SIGKDD Explor. Newsl, 2014.

Harry "Dark SEO". Letter derrotation; 2008a. Available from: https://web.archive.org/web/20080611182905/http://www.darkseoprogramming.com/2008/04/05/letter-derotation/.

Harry "Dark SEO". Phpbb3 CAPTCHA is super easy. 2008b. Available from: https://web.archive.org/web/20080701082455/http://www.darkseoprogramming.com/2008/05/12/phpbb3-captcha-is-super-easy/.

Hernández-Castro CJ, Ribagorda Garnacho A, Saez Y. Side-channel attack on the HumanAuth CAPTCHA. In Proceedings of the International Conference on Security and Cryptography (Secrypt), 2010.

Hernández-Castro CJ, R-Moreno MD, Barrero DF. Using jpeg to measure image continuity and break Capy and other puzzle CAPTCHAs. IEEE Internet Comput 2015;19(6):46–53. doi:10.1109/MIC.2015.127. ISSN 1089-7801.

Hernandez-Castro CJ, Ribagorda A. Pitfalls in CAPTCHA design and implementation: the math CAPTCHA, a case study. Computers & Secur 2009a;doi:10.1016/j.cose.2009.06.006. ISSN 01674048.

Hernandez-Castro CJ, Ribagorda A, Saez Y. Side-channel attack on labeling CAPTCHAs; 2009b. Available from: http://arxiv.org/abs/0908.1185.

Hernandez-Castro CJ, Ribagorda A, Hernandez-Castro JC. On the strength of egglue and other logic CAPTCHAs. In Proceedings of the International Conference on Security and Cryptography, pages 157–167; 2011.

Hernandez-Castro CJ, Barrero DF, R-Moreno MD. A machine learning attack against the civil rights CAPTCHA. In Proceedings of the 8th International Symposium on Intelligent Distributed Computing – IDC2014, 2014.

Hindle A, Godfrey MW, Holt RC. Reverse engineering CAPTCHAs, 2008.

Hoepman J-H, Jacobs B. Increased security through open source. Commun ACM 2007;50(1):79–83. doi:10.1145/1188913.1188921. ISSN 0001-0782.

Keerthi SS, Shevade SK, Bhattacharyya C, Murthy KRK. Improvements to Platt's SMO algorithm for SVM classifier design. Neural Comput 2001;13(3):637–49.

Kim J, Kim S, Yang J, Ryu J-H, Wohn K. FaceCAPTCHA: a CAPTCHA that identifies the gender of face images unrecognized by existing gender classifiers. Multimed Tools Appl 2014;72(2):1215–37. doi:10.1007/s11042-013-1422-z. ISSN 1380-7501.

Kim J-W, Chung W-K, Cho H-G. A new image-based CAPTCHA using the orientation of the polygonally cropped sub-images. Visual Comput 2010;26(6–8):1135–43. doi:10.1007/s00371-010-0469-3. ISSN 0178-2789.

Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc.; 2012. Available from: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

le Cessie S, van Houwelingen JC. Ridge estimators in logistic regression. Appl Stat 1992;41(1):191–201.

Landwehr N, Hall M, Frank E. Logistic model trees. Mach Learn 2005;95(1–2):161–205.

LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. P IEEE 1998;86(11):2278–324. doi:10.1109/5.726791. ISSN 0018-9219.

Mohamed M, Sachdeva N, Georgescu M, Gao S, Saxena N, Zhang C, et al. Three-way dissection of a game-CAPTCHA: automated attacks, relay attacks, and usability. CoRR, abs/1310.1540, 2013.

Naor M. Verification of a human in the loop or identification via the Turing test; 1996. Available from: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps.

Osadchy M, Hernandez-Castro J, Gibson S, Dunkelman O, Pérez-Cabo D. "No bot expects the DeepCAPTCHA! Introducing Immutable Adversarial Examples with Applications to CAPTCHA". IEEE Transactions on Information Forensics and Security, in print. 2017

Santamarta R. Breaking Gmail's audio CAPTCHA; 2008. Available from: http://blog.wintercore.com/?p=11.

Shalev-Shwartz S, Singer Y, Srebro N. Pegasos: primal estimated sub-gradient solver for SVM. In 24th International Conference on Machine Learning, pages 807–814, 2007.

Sivakorn S, Polakis I, Keromytis AD. I am robot: (deep) learning to break semantic image CAPTCHAs. In Proceedings of the 1st IEEE European Symposium on Security and Privacy, EuroSP '16, 2016.

Swire P. A model for when disclosure helps security: what is different about computer and network security? J Telecomm High Tech Law 2004;2.

Tam J, Simsa J, Hyde S, von Ahn L. Breaking audio CAPTCHAs. Advances in Neural Information Processing Systems (NIPS), 2008.

von Ahn L, Blum M, Hopper NJ, Langford J. CAPTCHA: Using hard AI problems for security. In Proceedings of Eurocrypt, volume 2656, pages 294–311; 2003. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.3335.

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 '11, pages 237–246, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0672-0. http://doi.acm.org/10.1145/2076732.2076766.

Warner O. KittenAuth; 2009. Available from: http://www.thepcspy.com/kittenauth.

Wieser W. CAPTCHA recognition via averaging; 2007. Available from: http://www.triplespark.net/misc/captcha/.

Wilkins J. Strong CAPTCHA guidelines; 2010. Available from: http://bitland.net/captcha.pdf.

Yan J, El Ahmad AS. Breaking visual CAPTCHAs with naïve pattern recognition algorithms, 2008a.

Yan J, El Ahmad AS. A low-cost attack on a Microsoft CAPTCHA, 2008b.

Zhu BB, Yan J, Li Q, Yang C, Liu J, Xu N, et al. Attacks and design of image recognition CAPTCHAs. In Proceedings of the 17th ACM conference on computer and communications security, CCS '10, pages 187–200, New York, NY, USA; 2010. ACM. ISBN 978-1-4503-0245-6. http://doi.acm.org/10.1145/1866307.1866329.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005

David F. Barrero is an Associate Professor at the Escuela Politécnica Superior of the Alcalá University in Madrid, Spain. His research interests involve Evolutionary Computation and Data Analysis applied to the empirical study of algorithms.

María Dolores Rodríguez Moreno is an Associate Professor at the Escuela Politécnica Superior of the Alcalá University in Madrid, Spain. She collaborates with the Planning and Scheduling Team group in the STC-CNR of Rome. She contributes to the study of heuristics for goal recognition and probabilistic planning with researchers at NASA Ames Research Center. In collaboration with the Jet Propulsion Laboratory (JPL) she studies cognitive control architectures and path-planning algorithms than can be use to plan the routes for the Curiosity and MER-Opportunity rovers.

Stuart Gibson is a lecturer in the School of Physical Sciences, University of Kent, UK. HE is the co-inventor of the EFIT-V facial composite system which is currently used by the majority of UK police constabularies and in numerous other countries.

Carlos Javier Hernández Castro is a PhD candidate at the Alcalá University in Madrid, Spain. His research interests include IT Security, machine learning, and their confluence: CAPTCHAs/HIPs. He is also interested in ML and Security applied to Games and planning.

Please cite this article in press as: Carlos Javier Hernández-Castro, María D. R-Moreno, David F. Barrero, Stuart Gibson, Using machine learning to identify common flaws in CAPTCHA design: FunCAPTCHA case analysis, computers & security (2017), doi: 10.1016/j.cose.2017.05.005