How to Break CaptchaStar

Thomas Gougeon and Patrick Lacharme
Normandie Universite, ENSICAEN-UNICAEN-CNRS, GREYC UMR 6072, France

Keywords: CAPTCHA.

Abstract: Most of web sites adopt a Captcha system to distinguish malicious software from humans. This paper proposes an attack on a recent interactive image-based Captcha scheme, called CaptchaStar. The CaptchaStar was designed to be more secure and user friendly than existing solutions, however, as we show in this paper, it fails to meet these goals Nevertheless, the presented attack is very efficient, with a success rate for the attacker of 96% on the on-line version proposed by the authors. Moreover, the modification of CaptchaStar parameters (as noise addition) does not prevent our attack.

1 INTRODUCTION

CAPTCHAs [1] (Completely Automated Public Turing test to tell Computers and Humans Aparts) are systems used on the Internet to differentiate between human and software (von Ahn et al., 2003) (the term Turing Test comes from (Turing, 1950)). These tests should be easily implementable, easy to solve for a human and hard for a computer program. Captchas are well known to the general public because most of Internet giants, as Google, Microsoft or Amazon, use them for the creation of a free account, as protective measure against malicious bot systems. In all cases, users are asked to provide some response to a challenge proposed by the web site, in order to access the service. Captchas have considerably changed over the years, because most of these schemes were proved not secure, in the sense that it was possible for a software to solve the proposed challenge (a lot of techniques have also been developed to break these Captchas). Nevertheless, even if most of Captcha schemes are broken, these systems are still considered to be relevant, useful and continue to be used in practice by a lot of service providers (Thomas et al., 2013).

The design of Captcha systems should take into account several parameters, particularly the success rate and the usability. In this paper the success rate is considered by default from the attacker's perspective, as the proportion of Captchas that an automatic method is able to answer correctly. For example, a rate of 0.01% is suggested as acceptable in (Chellapilla et al., 2005; Yan and Ahmad, 2007b), but an higher rate between 0.6% and 1% is considered as more realistic in (Zhu et al., 2010; Bursztein et al., 2011b). This rate is obviously linked to the success rate of a genuine human, proposed at 90% in (Chellapilla et al., 2005; Yan and Ahmad, 2007b). Nevertheless, in most case this last rate is often hard to reach, particularly in the case of audio Captcha (Bursztein et al., 2010) (but this aspect is not related to this paper). A second parameter is the time to solve the Captcha: the attack should not be slower than a human response. In (Rui and Liu, 2004; Bursztein et al., 2011b), it is suggested that human should be able to provide a response within 30*s*.

CaptchaStar is a new image-based Captcha (shape discovery) proposed at ACNS 2016 by M. Conti, C. Guarisco and R. Spolaor in (Conti et al., 2016a). Each image of this Captcha is composed of *stars*, that are small groups of white pixels. At the beginning a window with random stars is generated, then these stars are moved in interaction with the mouse of the user. A shape (formed of most of stars) appears on the screen if the mouse is close to the correct place. In this case the user is asked to click and the position of the mouse is sent to the server. A demo is available on the web site of CaptchaStar: (Conti et al., 2016b). This Captcha is presented by the authors as user-friendly, secure and suitable for mobile applications.

CaptchaStar is clearly a novel and elegant type of Captcha, but the security of the scheme is not as robust as presented by authors. This paper presents an extremely efficient attack on the proposed implementation of CaptchaStar, with a success rate of 96%. The proposed attack is only based on the concentration of pixels during the formation of the image. Simulating different positions of the mouse on the screen, it automatically determines an approximation of the

[1] For readability reasons, this acronym will be now written in lowercase.

Gougeon, T. and Lacharme, P.
How to Break CaptchaStar.
DOI: 10.5220/0006577600410051
In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*, pages 41-51
ISBN: 978-989-758-282-0

solution. Improvements, limits, and corrections of CaptchaStar (with modified parameters) are discussed at the end of the paper.

This paper is organised as follows, Section 2 presents the state of the art on previous attacks realised on various types of Captcha. Section 3 presents the new system CaptchaStar and Section 4 describes the attack on this system. Finally, Section 5 proposes a discussion on possible improvements in this type of Captcha (with modified parameters).

2 STATE OF THE ART ON CAPTCHAS

There exist several types of Captcha, from the classical text-based system where an user is asked to read a distorted word (originally embedded in a 2D image) to other various systems, based for example on image recognition, combined with some questions. If these distorted text-images are sophisticated, they are not easily recognisable by a machine, but an human should be able to read them, without much effort.

2.1 Text-based Captchas

Text-based Captchas are simple to understand for a large public and are used in most of applications since fifteen years (Baird et al., 2003). They are easily implementable, with various designs, as for example the old Yahoo's Captcha Gimpy and Ez-Gimpy described in (Mori and Malik, 2003). One of the most used text-based captcha system, is reCaptcha (von Ahn et al., 2008), where users are asked to read scanned words from books (helping to digitise old printed material), combined with a second simply distorted word. This system was later acquired by Google (in 2009) (von Ahn, 2009) (with several attacks as in (Goodfellow et al., 2013)) and was recently replaced by new mechanisms (including noCaptcha), as described and recently attacked in (Sivakorn et al., 2016). More sophisticated text-based Captchas have been proposed, as for example hollow Captchas, 3D text-based Captchas or animated text-based Captchas (as NuCaptcha (Xu et al., 2012)).

The first attack on text-based Captchas was proposed by Mori and Malik in 2003 (Mori and Malik, 2003). This attack works with matching using shape context of characters and a database of image of known objects. The success rate was 92% on EZ-Gimpy (a single word) and 33% on Gimpy (three words in an image). The following year, Chellapilla and Simard use machine learning techniques (and segmentation) to break six other text-based Captchas (Chellapilla and Simard, 2004). It includes Captchas used by Ticketmaster, Yahoo V2 and Google/Gmail, with success rates of 45.7% for Yahoo V2 and 4.9% for Google.

Yan and El Ahmad broke a large set of text-based Captchas using very simple techniques (as the pixel count by letters), or more evaluated techniques in several papers (Yan and Ahmad, 2007a; Yan and Ahmad, 2007b; Yan and Ahmad, 2009; Ahmad et al., 2011). For example, success rates on (old) Microsoft Captcha, Megaupload or reCaptcha are between 33% and 90%. This last work improved the success rate of previous attacks on Recaptcha, where the success rate was between 10% and 31% (Wilkins, 2009). More recent contributions on text-based Captchas attacks include the attack of Bursztein et al. (based on SVM and KNN classifiers) on several Captcha systems with variable success rates (from 0% to 93%) (Bursztein et al., 2014a), the attack of Gao et al. on hollow Captchas with success rates between 36% and 89% (Gao et al., 2013). Finally, an attack on text-based Captchas, using Gabor filters, has been recently proposed in (Gao et al., 2016), with a success rate between 5% and 77% in less of 15 second on a standard computer on various Captchas as the new version of reCaptcha (described in (Bursztein et al., 2014b) and currently widely used by Facebook, Google, Youtube, Linkedin and Twitter).

An alternative to simple text-based Captchas are 3D text-based Captchas, but these systems are also vulnerable to attacks (Nguyen et al., 2012; Nguyen et al., 2014). Finally, the robustness of animated text-based Captchas, as proposed in (Kluever and Zanibbi, 2009; Cui et al., 2010) is investigated in (Xu et al., 2012). All the mentioned attacks are not described because the Captcha system attacked in this paper is not a text-based Captcha and consequently, there are no direct links between them and the following of this paper. A good survey on these Captchas can be found in (Basso and Bergadano, 2010; Bursztein et al., 2011b).

2.2 Image-based Captchas and Variants

The failure of secure text-based Captchas encouraged the design of other types of Captcha, as proposed by (Chew and Tygar, 2004) in 2004. In this case, the user is asked to link a word with an image (with variations in different schemes). Nevertheless, most of them were not more secure than previous schemes. One of most known image-based scheme is Asirra (Animal Species Image Recognition for Restricting Access), described in (Elson et al., 2007) and also defined as an HIP (Human Interactive Proof). In this system, users

are asked to identify dogs and cats in a set of random images. Nevertheless, the system was not secure and Asirra has been closed by Microsoft in 2014. The image database should be (very) large and the number of possible responses, related to the image(s) should be sufficiently high, in order to avoid a random response.

Most of the first generation of image-based Captchas were simply vulnerable to random guessing attacks, because there was only a limited number of possible response to the challenge. In 2008, Golle proposed an attack on the image-based Captcha Asirra, using a machine learning technique (more precisely a SVM classification) with a success rate of 10% (Golle, 2008) (an other study on cat head detection provides success rates of 90% (Zhang et al., 2008)). Others image-based Captchas were proposed as Imagination (Datta et al., 2005) and ARTiFACIAL (Rui and Liu, 2004), but they were attacked in (Zhu et al., 2010) with other image-based Captchas. Authors propose a new scheme, called CORTCHA, robust to their attacks, but the generation of the challenge takes 122 seconds on a standard PC. Two new (and independent) schemes, both called DeepCaptcha, are proposed in (Nejati et al., 2014; Osadchy et al., 2016) using deep learning (there are no direct link between these schemes). To the best of our knowledge, there are no attacks against the two last schemes. Finally a recent attack on the image-based reCaptcha and the image-based Facebook Captcha gives success rates of 70 and 83% respectively.

Audio Captchas have been proposed as (Shirali-Shahreza and Shirali-Shahreza, 2008), because they are useful for people visually impaired. However, they are difficult for a non-English speaker. Attacks on audio Captcha can be found in (Tam et al., 2008; Bursztein and Bethard, 2009; Bursztein et al., 2011a), where the conclusion is a total failure of these systems. Game Captchas are analysed in depth in (Mohamed et al., 2014b). Finally another scheme, called Captcha as graphical passwords (CaGP), is proposed in (Zhu et al., 2014). All these systems are also far off the analysed Captcha of this paper.

These alternatives are useful, because classical text-based Captcha seems not entirely adapted to smartphone applications (for usability reasons as the screen size or environmental conditions). For example the NuCaptcha scheme provides better success rates for smarpthone users in the evaluation of (Reynaga and Chiasson, 2013).

Another type of attack, called relay attacks (or indirect attacks), requires an external human to solve the Captcha. Relay attacks are possible on many type of Captcha and are completely different to the previous automated attacks. These attacks are not negligible because some service providers propose to solve a large number of Captcha for very low price (Motoyama et al., 2010). The objective of interactive Captcha is the mitigation (and/or the detection) of these attacks, as presented in the case of a game Captcha in (Mohamed et al., 2014a). This scheme is based on a drag-and-drop system between two sets of objects. Finally, a text-based Captcha, called icaptcha (Truong et al., 2011), was also designed to mitigate relay attacks with a timing analysis between interactions, based on a sequence of mouse clicks.

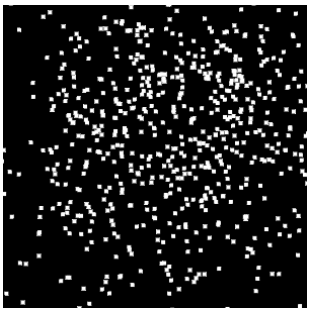
3 CAPTCHASTAR

This section presents CaptchaStar (Conti et al., 2016a), a recent image-based Captcha that relies on user interaction. A demo of the Captcha is available online (Conti et al., 2016b). CaptchaStar authors argued that their solution is better than other Captchas from the literature in terms of resiliency against automated attacks and usability.

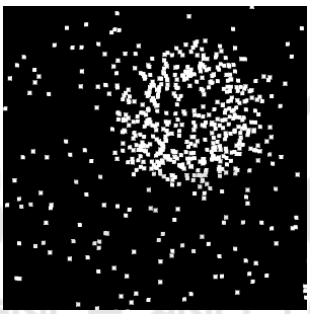
The challenge presented by CaptchaStar is a pixel grid 300×300 with a black background containing randomly positioned stars. A star corresponds to a white square 5×5. Moving the mouse cursor changes the position of the stars. In order to solve the challenge, the user has to move the cursor until he is able to recognise a shape. Only one position of the cursor leads to reconstruct perfectly the shape, this position is kept secret from the user and it is the solution of the challenge. When the cursor is far away from the solution, the stars seem to be displayed randomly, but approaching the solution, the stars aggregate. Once the user is confident that its cursor is correctly positioned, he clicks to submit its answer. Then, the server compares the submitted cursor position to the solution, and if they are close enough the user is considered as a human. Figure 1 presents the process of solving a CaptchaStar challenge. Figure 1(a) is not known by the user, it is used by the server to generate the challenge.

The shape is created with a randomly picked image from a pool using a sampling algorithm. CaptchaStar authors use a pool of $5,000$ 2-color icons. The sampling algorithm decomposes the image into stars. The generated stars are then randomly positioned to be displayed on the grid. In addition to the stars generated by the sampling algorithm, some noisy stars are randomly added to prevent automated attacks. The usability and the security of the challenge can be tuned adjusting some parameters values. These parameters include the noise (ψ) defining the percentage of added stars, the sensitivity(δ) represent-

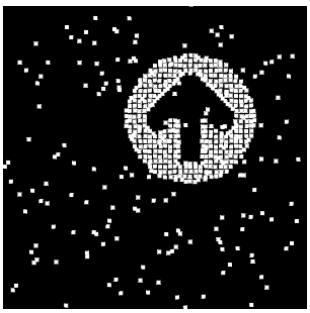
(a) A random starting picture.



(b) A sample unsolved challenge.



(c) An almost solved challenge.



(d) A correctly solved challenge.

Figure 1: The process of solving a CaptchaStar challenge.

ing the relationship between the cursor movement and the movement of each star, the number of possible solutions (*NSol*) representing the number of shapes hidden in the Captcha, the rotation indicates whether the picture is rotated by a random degree or not, and the usability tolerance α representing the maximum distance allowed between the answer and the solution to be considered as a human.

After picking randomly a picture, the sampling algorithm generates a set of n stars from the figure, $\mathbf{S} = \{s^i, 1 \leq i \leq n\}$. Each star s^i is defined by a pair of coordinates (s^i_x, s^i_y). The sampling also generates the original coordinates $P^i = (P^i_x, P^i_y)$ for each star s^i. In addition to the stars belonging to the shape, there are $\psi \times n$ noisy stars that are generated with random coordinates. The solution of the challenge is defined by a pair of coordinates $sol = (sol_x, sol_y)$ where each coordinate is picked randomly in the range $[5, 295]$, thus the solution is never too close of the edges of the grid. Moreover, the coordinates solution are generated independently to the figure. Two challenges generated with the same figure, will have a different solution.

The position (i.e. coordinates) of s^i changes with the cursor position ($cur = (cur_x, cur_y)$). It is computed using P^i, cur, and some coefficients picked randomly for each star. The exact computation of the trajectory of stars is not detailed here because it is not directly used in our attack (it depends also of the sensitivity parameter). When the user clicks on the grid, the current cursor position, $ans = (ans_x, ans_y)$, is submitted to the system. Then, the server computes the euclidean distance Δ between the answer and the solution. The distance Δ is then compared to a threshold α representing the usability tolerance. When Δ is lower (resp. greater) than α the challenge is solved (resp. failed). The value $\alpha = 5$ is proposed in the original paper.

3.1 Resiliency to Automated Attacks

CaptchaStar authors claimed that their solution showed promising results against all the studied attacks, comparable or even better than the state of the art.

CaptchaStar defeats classic attacks as indirect attacks, database exhaustion or leak of database, random choice, and pure relay attacks. Only the stream relay attacks that is the more powerful one, is a real threat to this Captcha, but it is difficult to set up

The authors also investigated the resiliency of their Captcha against machine learning. The best parameters for their machine learning technique lead to a success rate of 78.1% with a computation time of 421 seconds.

CaptchaStar authors also proposed some heuristics to automatically solve the Captcha, e.g. looking at the dispersion of the stars or the distance between the most distant stars. The faster heuristic leads to less than 1% of success rate with a computing time greater than 60 seconds, and the best heuristic in term of success rate is 1.92% with a computing time of $1,500$ seconds.

Recalling that a human solves the challenge in an average time of 23.1 seconds with a success rate of 91.0%.

4 A NEW AUTOMATED ATTACK AGAINST CAPTCHASTAR

This section presents the proposed ad-hoc attack against CaptchaStar. First, the heuristic of the attack is described. Second, the attack principle is detailed. Then, experiments together with the results are given. Finally, some information about the implementation are mentioned.

4.1 Heuristic of the Attack

MaxConcentration: Looking at the Figure 1, one can remark that, when the challenge is unsolved the stars are dispersed on the whole grid. When the cursor is near the solution the stars aggregate and when the cursor is at the coordinates of the solution, the stars are totally aggregated to reform the shape. At this position, the only dispersed stars represent the noise (because $NSol = 1$ in this figure). As a consequence, the proposed attack looks at the stars concentration in given areas. The heuristic aims to maximise the concentration of the stars into a part of the grid.

More precisely, given a state S^k, the grid is separated in a set T^k of squared tiles of $\ell \times \ell$ pixels. In this case, the pixels of each tile t are defined by $t_{i,j}$ with $1 \leq i, j \leq \ell$ and they assume 0 and 1 values for *black* and *white* colors. For each tile t^k belonging to T^k, a score is computed. This score represents the number of white pixels of the tile, and is computed as follows:

$$\text{score}(t^k) = \sum_{i=0}^{\ell} \sum_{j=0}^{\ell} t^k_{i,j}.$$

Then, let $\vec{P}^k$ be a vector containing all the scores $\text{score}(t^k)$ for t^k in T^k sorted with descending order. Then, the final score of the state S^k is computed by summing up the n_{max} first values of $\vec{P}^k$, as follows:

$$maxConcentration(k) = \sum_{i=0}^{n_{max}} \vec{P}^k_i.$$

Parameters value of ℓ and n_{max} are discussed in the next section. For example, given a state, using parameters value ℓ = 10 and n_{max} = 20, the heuristic computes the number of white pixels in each 10×10 tiles of the grid, and the heuristic adds the 20 largest scores.

4.2 The Attack

The proposed attack uses the previous heuristic as follows. First, the different states of the challenge are obtained by moving the cursor position. Then, for each state, a score is computed using an heuristic. Finally, the position that leads to the best score is submitted as the solution of the challenge.

More precisely, the attack is composed of two steps, the first phase determines an approximate position for the solution by looking at a sub-part of the possible states. The second phase, looking at all the states around the approximate solution, aims to determine the exact position of the solution. The two phases use the heuristic *maxConcentration* to compute the scores. The objective of this separation in two steps is the reduction of the execution time.

During the first phase, the scores are not computed for all the $84,100$ possible states but only for n_s of them, with $n_s < 84,100$. More precisely, the grid is split in a set of n_s squared tiles and each tile center is used as coordinates k to generate the n_s states S^k. The first part of the attack returns the coordinates c_k of the state S^k leading to the maximum score among the n_s states. It represents an approximation of the challenge solution. Then, the second phase considers the coordinates c_k as a the center of a tile of size $\ell_2 \times \ell_2$. A score is computed using the heuristic *maxConcentration* for all the states generated by the $\ell_2 \times \ell_2$ coordinates of the tile. Table 1 presents the parameters value used for the attack, their choice is discussed in the next section. Figure 2 shows the result after the first phase and after the second phase on one example.

Table 1: Parameters value of the attack.

Parameters	ℓ	n_{max}	n_s	ℓ_2
Value	10	20	2,500	20

4.3 Experiments

CaptchaStar authors propose 6 sets of parameters, as presented in Table 2.

In the demo version, we expected the parameters value for the noise, the sensitivity, the number of possible solutions, and rotation used to be those of T_3, but it seems that the rotation is not implemented. Actually, there are two versions of the paper, (Conti et al., 2016a) and another available on arxiv (Conti et al., 2015). (Conti et al., 2016a) obtains better success rate for T_3 parameters and the one available on arxiv obtains better success rate for T_2 parameters. The only difference between T_2 and T_3 is the use of the

Table 2: Parameters proposed by the authors.

Test	T_1	T_2	T_3	T_4	T_5	T_6
ψ	0%	70%	70%	10%	0%	250%
δ	5	7	7	7	10	5
NSol	1	1	1	2	3	1
Rate (%)	77.0	87.1	91.0	46.4	82.7	75.5
Time (s)	15.0	18.8	23.1	59.5	32.8	31.1

rotation. As a consequence, we suppose that on the demo version the parameters used are T_2.

Applying the proposed attack on the CaptchaStar demo website proposed by the authors, leads to a success rate of 96% with an execution time lower than 12 seconds per Captcha. These results are obtained by executing the JavaScript code of the attack with Chromium 51.0.2704.79 on Ubuntu 14.04 (64-bit). The attack was executed on $1,000$ challenges. Recalling that, having more than 1% of success rate is sufficient to consider a Captcha as broken. One can note that, the obtained success rate by the automated attack is greater than the one obtained by the user study (91%), and the execution time of the attack is lower than the solving time of a human ($\approx 20s$) for T_2.

The few challenges resisting to the proposed attack are those containing figure with an important amount of stars or figures with a blank zone. Only few cases of these figures lead to a failure. Figure 3 shows two challenges resisting to the proposed attack. The solution found by the attack is near from the solution of the challenge. We assume that, optimising the parameters, the attack will be successful against these challenges.

4.4 Attack Implementation

The grid uses a Canvas, an HTML5 object, from this object it is easy to extract the RGBA values for each pixel using JavaScript. Pixels are black or white, so using the RGBA values it is easy to transform it into a matrix of 0 and 1. When the cursor moves, the stars position are updated on the client side, with a JavaScript code. Therefore, it is possible to move the stars with a program using JavaScript calls. The challenge generation and verification are done with a PHP script on the server side.

There is no guarantee that the parameters (ℓ, n_{max}, n_s, and ℓ_2) used for our attack are optimal. It is explained by two reasons. First, for a sake of simplicity, we have run our experiments using JavaScript but the execution time is not optimal with this language. Increasing the value of ℓ_2 and n_s increases the accuracy of the attack, and so the success rate, but also the execution time. Using another programming language, as the C language, and make the computation of the scores in parallel, can strongly reduce the execution time. There exists, some solution as Selenium driver, to interact between JavaScript and other languages. Second, tuning the parameters gives only the best success rate for one implementation of CaptchaStar. Optimal parameters of the attack clearly depend on parameters of CaptchaStar implementation. The objective is to demonstrate that CaptchaStar is not ready for real use, by establishing a proof of concept of the attack.

Finally, in an optimal version of the attack, the parameters could be chosen dynamically depending the number of stars of the challenge.

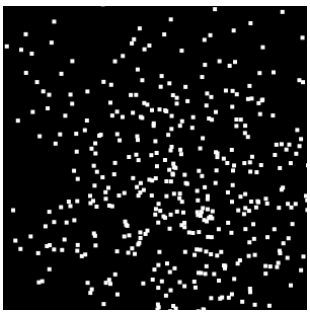
5 DISCUSSION ON SECURITY

The discussion is divided in two parts, the first one discusses about the parameters of the implementation of CaptchaStar, the second one discusses about other countermeasures that can be added to the Captcha scheme.

5.1 Modifying Parameters of CaptchaStar

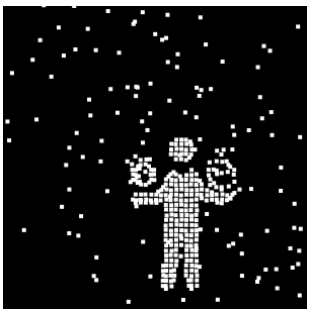
The success rate of 96% of the previous section has been achieved on the implementation of CaptchaStar available on the website of the authors of (Conti et al., 2016a). Nevertheless, five parameters are proposed by the authors: the noise (ψ, percentage of noisy stars), the sensitivity (δ, relationship between the mouse cursor and the stars movement), the number of possible solutions (*NSol*), and the rotation (a random degree of rotation). This section investigates the security of this Captcha scheme, independently to the proposed implementation. Clearly, the proposed attack is not affected by the rotation.

(a) A random starting picture.



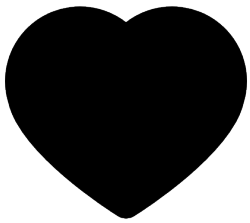
(b) A sample unsolved challenge.

(c) State after the first phase of the attack.

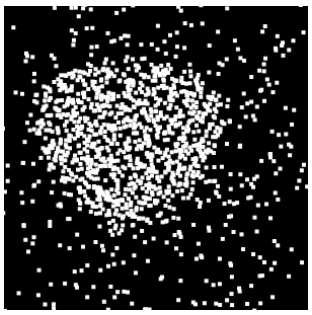


(d) State after the second phase of the attack.

Figure 2: The process of attacking a CaptchaStar challenge.

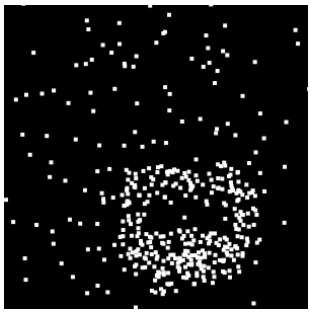


(a) Figure not recognised by the attack, containing an important amount of pixels.



(b) Wrong solution found by the attack.

(c) Figure not recognised by the attack, containing a blank area.



(d) Wrong solution found by the attack.

Figure 3: The process of attacking a CaptchaStar challenge.

5.1.1 Noise

Addition of noise is not really problematic for the proposed attack. Figure 4(a) shows the attack result on the same original image than the Figure 2, but with a noise of 200%. Tests T_2 and T_6 have the

same parameters, excepted a noise percentage of 70% against 250% and a lower sensitivity. T_6 provides a lower success rate for the user than T_2 (75.5% against 87.1%) but the difference is not really high (see Table 2). We have run the attack on 100 samples on the demo version of CaptchaStar, adding noisy stars up to 200% to each challenge. The obtained success rate is 91%, it is lower than the success rate obtained with 70% noise, but only 5% lower. As discussed previously, the attack parameters could be tuned to obtain a better success rate. Nevertheless, 91% is still a high success rate.

We remarked that the generated stars representing the noise are not always displayed on the grid, due to their coordinates when moving the cursor. A modification of the noise generation can increase the difficulty by generating only visible stars. This modification also increases the difficulty for a human.

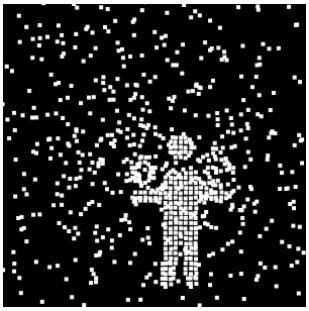
5.1.2 Sensitivity

Increasing the sensitivity increases the movement amplitude of the stars when moving the mouse cursor. We assume that it does not influence a lot our heuristic because it does not change the number of stars displayed on the grid. It modifies the concentration of pixels when the cursor is near the solution because the stars will aggregate faster or slower depending the value of the sensitivity. Nevertheless, we assume that by optimising the parameters value of the attack, the sensitivity does not reduce a lot the success rate of the attack.

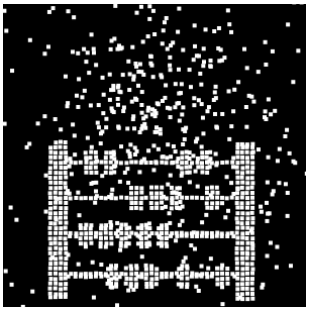
5.1.3 Number of Possible Solutions

Hiding several figures into the Captcha is not really problematic for the proposed attack. Figure 4(b) shows the attack result on a challenge containing 2 hidden figures. One figure is correctly retrieved even if there are two figures hidden in the challenge. We have run the attack on 100 samples on the demo version of CaptchaStar hiding a second figure in each challenge. We define a success when one of the two figures is retrieved by the attack. The obtained success rate is 94%, it is only 2% lower than the success rate with one figure. As we have run the experiments with only 100 samples, the obtained success rate is less accurate. Nevertheless, hiding a second figure does not prevent the attack.

Once the first figure is retrieved by the attack, the second figure can be retrieved by ignoring the position near the first solution, and so on if there are more than two hidden figures. For a genuine user, increasing the number of hidden figures, reduces the success rate and increases significantly the solving time. For example,



(a) Implementation with $\psi = 200\%$.



(b) Implementation with $NSol = 2$ and $\psi = 70\%$.

Figure 4: Executing CaptchaStar attack with other parameters.

the success rate falls from 90% to 50% between tests T_3 and T_4 with the same parameters excepted a lower noise and a second image hidden in T_4 (see Table 2).

5.2 Other Countermeasures

5.2.1 Solver Behaviour

As mentioned by the CaptchaStar authors, an automated attack can be detected by analysing the movement of the mouse cursor during the challenge solving. Nevertheless, it seems that this countermeasure is difficult to set up. Characterising the human solving is such a hard task because each human will solve differently the challenge by creating its own heuristic.

The proposed attack needs only n_s states for the first phase, the attack can simulate mouse cursor movements on the whole grid, capturing only the states that are needed for the attack. Execution time increases when a score is computed for a state, not when the cursor moves. Moreover, the order of capturing the states can be randomised, and random cursor movements can be added not to be systematic. The captured states, can be near the center of the tiles in order not to be detected by systematically reach the center tiles, and the parameters values of the at-

tack can vary a little between each attack to not be detected.

The second phase needs $\ell_2 \times \ell_2$ states that are near the approximate solution, found by the first phase. Random movements can be simulated around this approximate solution, and only needed states can be captured to compute the scores. Another possible behaviour can be to artificially draw ellipses, circles or squares, etc. with the simulated cursor around the approximate solution. We assume that these behaviours are not far away from a human behaviour when a genuine user is near the solution.

5.2.2 Recognise the Shape

The proposed attack is only able to detect that there is a shape. As a consequence, CaptchaStar can hide more than one image in the Captcha, for example a human and a dog. Then, the Captcha asks the user to retrieve the image representing a human. If the user retrieves the dog, he must not click. The proposed attack is not able to detect if the shape is a dog or a human, then it will submit the first shape discovered. Nevertheless, this countermeasure leads to a success rate of 50% by always selecting the first shape detected (assuming that solving CaptchaStar is near 100% success rate). Moreover, machine learning obtains good results on image recognition (Simonyan and Zisserman, 2014) and can be used to defeat this countermeasure.

6 CONCLUSION

This paper presents an automated ad-hoc attack against CaptchaStar, a recent image-based Captcha. Although CaptchaStar seemed to be promising in terms of usability and resiliency the proposed attack defeats it completely. The proposed attack reaches a success rate of 96%, in less than 12 seconds per Captcha. Consequently, it solves the Captcha better than a human according to success rate and computation time.

Establishing countermeasures in order to prevent this attack seems to be a difficult task. The Captcha needs to profoundly change as the attack targets its core. A possible countermeasure would be a combination of CaptchaStar with an entirely different second system.

REFERENCES

Ahmad, A. S. E., Yan, J., and Tayara, M. (2011). The robustness of google captchas. Technical Report.

Baird, H. S., Coates, A. L., and Fateman, R. J. (2003). Pessimalprint: a reverse turing test. *International Journal on Document Analysis and Recognition*, 5(2-3):158–163.

Basso, A. and Bergadano, F. (2010). Anti-bot strategies based on human interactive proofs. In *Handbook of Information and Communication Security*, pages 273–291.

Bursztein, E., Aigrain, J., Moscicki, A., and Mitchell, J. C. (2014a). The end is nigh: Generic solving of text-based captchas. In *USENIX Workshop on Offensive Technologies (WOOT)*.

Bursztein, E., Beauxis, R., Paskov, H. S., Perito, D., Fabry, C., and Mitchell, J. C. (2011a). The failure of noise-based non-continuous audio captchas. In *IEEE Symposium on Security and Privacy (S&P)*, pages 19–31.

Bursztein, E. and Bethard, S. (2009). Decaptcha: breaking 75% of ebay audio captchas. In *USENIX Coference on Offensive Technologies*.

Bursztein, E., Bethard, S., Fabry, C., Mitchell, J. C., and Jurafsky, D. (2010). How good are humans at solving captchas? a large scale evaluation. In *IEEE Symposium on Security and Privacy (S&P)*, pages 399–413.

Bursztein, E., Martin, M., and Mitchell, J. (2011b). Text-based captcha strengths and weaknesses. In *ACM Conference on Computer and Communications Security (CCS)*, pages 125–138.

Bursztein, E., Moscicki, A., Fabry, C., Bethard, S., Mitchell, J. C., and Jurafsky, D. (2014b). Easy does it: more usable captchas. In *Conference on Human Factors in Computing Systems (CHI)*, pages 2637–2646.

Chellapilla, K., Larson, K., Simard, P. Y., and Czerwinski, M. (2005). Building segmentation based human-friendly human interaction proofs (hips). In *Workshop on Human Interactive Proofs (HIP)*, pages 1–26.

Chellapilla, K. and Simard, P. Y. (2004). Using machine learning to break visual human interaction proofs (hips). In *Neural Information Processing Systems (NIPS)*, pages 265–272.

Chew, M. and Tygar, J. D. (2004). Image recognition captchas. In *International Information Security Conference (ISC)*, pages 268–279.

Conti, M., Guarisco, C., and Spolaor, R. (2015). Captchastar! a novel captcha based on interactive shape discovery. *arXiv preprint arXiv:1503.00561*.

Conti, M., Guarisco, C., and Spolaor, R. (2016a). Captchastar! a novel captcha based on interactive shape discovery. In *Applied Cryptography and Network Security (ACNS)*, pages 611–628.

Conti, M., Guarisco, C., and Spolaor, R. (2016b). Captchastar demo. http://captchastar.math.unipd.it/demo.php.

Cui, J.-S., Mei, J.-T., Zhang, W.-Z., Wang, X., and Zhang, D. (2010). A captcha implementation based on moving objects recognition problem. In *IEEE International Conference on E-Business and E-Government (ICEE)*, pages 1277–1280.

Datta, R., Li, J., and Wang, J. Z. (2005). Imagination: a robust image-based captcha generation system. In *ACM International Conference on Multimedia*, pages 331–334.

Elson, J., Douceur, J. R., Howell, J., and Saul, J. (2007). Asirra: a captcha that exploits interest-aligned manual image categorization. In *ACM Conference on Computer and Communications Security (CCS)*, pages 366–374.

Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., and Yan, J. (2013). The robustness of hollow captchas. In *ACM Conference on Computer and Communications Security (CCS)*, pages 1075–1086.

Gao, H., Yan, J., Cao, F., Zhang, Z., Lei, L., Tang, M., Zhang, P., Zhou, X., Wang, X., and Li, J. (2016). A simple generic attack on text captchas. In *Network and Distributed System Security Symposium, (NDSS)*.

Golle, P. (2008). Machine learning attacks against the asirra captcha. In *ACM Conference on Computer and communications security (CCS)*, pages 535–542.

Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. D. (2013). Multi-digit number recognition from street view imagery using deep convolutional neural networks. CoRR abs/1312.6082.

Kluever, K. A. and Zanibbi, R. (2009). Balancing usability and security in a video captcha. In *ACM Symposium on Usable Privacy and Security (SOUPS)*.

Mohamed, M., Gao, S., Saxena, N., and Zhang, C. (2014a). Dynamic cognitive game captcha usability and detection of streaming-based farming. In *Workshop NDSS on Usable Security (USEC)*.

Mohamed, M., Sachdeva, N., Georgescu, M., Gao, S., Saxena, N., Zhang, C., Kumaraguru, P., van Oorschot, P. C., and Chen, W.-B. (2014b). A three-way investigation of a game-captcha: automated attacks, relay attacks and usability. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 195–206.

Mori, G. and Malik, J. (2003). Recognizing objects in adversarial clutter: Breaking a visual captcha. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 133–144.

Motoyama, M., Levchenko, K., Kanich, C., McCoy, D., Voelker, G. M., and Savage, S. (2010). Re: Captchas-understanding captcha-solving services in an economic context. In *USENIX Security Symposium*, volume 10, pages 435–462.

Nejati, H., Cheung, N.-M., Sosa, R., and Koh, D. C.-I. (2014). Deepcaptcha: an image captcha based on depth perception. In *ACM Multimedia Systems Conference (MMSys)*, pages 81–90.

Nguyen, V. D., Chow, Y.-W., and Susilo, W. (2012). Breaking an animated captcha scheme. In *Applied Cryptography and Network Security (ACNS)*, pages 12–29.

Nguyen, V. D., Chow, Y.-W., and Susilo, W. (2014). On the security of text-based 3d captchas. *Computers & Security*, 45:84–99.

Osadchy, M., Hernandez-Castro, J., Gibson, S., Dunkelman, O., and Perez-Cabo, D. (2016). No bot expects the deepcaptcha! introducing immutable adversarial examples with applications to captcha. IACR Cryptology ePrint Archive.

Reynaga, G. and Chiasson, S. (2013). The usability of captchas on smartphones. In *International Conference on Security and Cryptography (SECRYPT)*, pages 427–434.

Rui, Y. and Liu, Z. (2004). Artifacial: Automated reverse turing test using facial features. *Multimedia Systems*, 9(6):493–502.

Shirali-Shahreza, S. and Shirali-Shahreza, M. (2008). Captcha for children. In *IEEE International Conference on System of Systems Engineering (SoSE)*, pages 1–6.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sivakorn, S., Polakis, I., and Keromytis, A. D. (2016). I am robot: (deep) learning to break semantic image captchas. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 388–403.

Tam, J., Simsa, J., Hyde, S., and von Ahn, L. (2008). Breaking audio captchas. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1625–1632.

Thomas, K., McCoy, D., Grier, C., Kolcz, A., and Paxson, V. (2013). Trafficking fraudulent accounts: the role of the underground market in twitter spam and abuse. In *USENIX Security Symposium*, pages 195–210.

Truong, H. D., Turner, C. F., and Zou, C. C. (2011). icaptcha: The next generation of captcha designed to defend against 3rd party human attacks. In *IEEE International Conference on Communications (ICC)*, pages 1–6.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

von Ahn, L. (2009). Teaching computers to read: Google acquires recaptcha. googleblogspot.

von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. (2003). Captcha: Using hard ai problems for security. In *International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 294–311.

von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. (2008). recaptcha : Human-based character recognition via web security measures. *Science*, 321.

Wilkins, J. (2009). Strong captcha guidelines. Technical Report (v1.2).

Xu, Y., Reynaga, G., Chiasson, S., Frahm, J. M., Monrose, F., and van Oorschot, P. C. (2012). Security and usability challenges of moving-object captchas: Decoding codewords in motion. In *USENIX Security Symposium*, pages 49–64.

Yan, J. and Ahmad, A. S. E. (2007a). Breaking visual captchas with naive pattern recognition algorithms. In *Annual Computer Security Applications Conference (ACSAC)*, pages 279–291.

Yan, J. and Ahmad, A. S. E. (2007b). A low-cost attack on a microsoft captcha. In *ACM Conference on Computer and communications security (CCS)*, pages 543–554.

Yan, J. and Ahmad, A. S. E. (2009). Captcha security: A case study. *IEEE Security & Privacy*, 7(4):22–28.

Zhang, W., Sun, J., and Tang, X. (2008). Cat head detection - how to effectively exploit shape and texture features. In *ECCV*, pages 802–816.

Zhu, B. B., Yan, J., Bao, G., Yang, M., and Xu, N. (2014). Captcha as graphical passwords - a new security primitive based on hard ai problems. *IEEE Transactions on Information Forensics and Security*, 9(6):891–904.

Zhu, B. B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., and Cai, K. (2010). Attacks and design of image recognition captchas. In *ACM Conference on Computer and Communications Security (CCS)*, pages 187–200.