In Proceedings of 2011 the 4th Symposium on Configuration Analytics and Automation (SafeConfig 2011), Arlington, VA, USA, October 31 - November 1, IEEE, 2012, DOI: 10.1109/SafeConfig.2011.6111665.

Captchæcker: Reconfigurable CAPTCHAs Based on Automated Security and Usability Analysis

Yousra Javed, Maliha Nazir,
Muhammad Murtaza Khan, Syed Ali Khayam
National University of Science & Technology (NUST)
Islamabad, Pakistan
Email: {yousra.javed, maliha.nazir,
muhammad.murtaza, ali.khayam}@seecs.nust.edu.pk

Shujun Li
University of Surrey
Guildford, UK
Email: Shujun.Li@surrey.ac.uk

***Abstract*—CAPTCHAs have been deployed ubiquitously by web sites to combat automated malicious programs. Security against web bots and usability to legitimate users are two main goals that have to be simultaneously satisfied when designing a useful CAPTCHA scheme. However, there exists a well-known and intricate trade-off between these goals. So far, balancing this trade-off remains an art rather than a science, as we do not have any automated tools to evaluate the security and usability of CAPTCHAs and then to configure the CAPTCHA generation engine accordingly. In this position paper, we propose a general framework called *Captchæcker* that aims to solve this configuration problem by automating the security-usability analysis of CAPTCHAs. The proposed framework will allow dynamic reconfiguration of a CAPTCHA scheme after its security-usability goal is changed or its security is compromised due to an attack.**

***Index Terms*—CAPTCHA, usability, security, design, reconfiguration**

I. Introduction

CAPTCHAs [1] (Completely Automated Public Turing tests to Tell Computers and Humans Apart) are used for preventing malicious automated programs from abusing online services that are mainly designed to serve human users. They have been widely deployed on web sites, especially web forums and online registration pages. The basic requirement of a robust CAPTCHA is that it should be easy to solve for humans with high probability (e.g. over 80%), whereas automated bots should not be able to solve it except with a negligible probability (e.g., 0.01%).[1]

The most widely used CAPTCHAs are text-based visual CAPTCHAs which present the user with an image containing distorted text. The user's task is to recognize the text inside the image and then enter it into a field on the protected web page. The text in a CAPTCHA is distorted in such a way that an average human user can recognize the text without too much effort but an automated program cannot recognize the text with a meaningful success rate. Many web sites also offer audio CAPTCHAs to enhance accessibility of their services to blind users. Other forms of CAPTCHAs have also been proposed over the years. Among these CAPTCHAs, video-based CAPTCHAs are quite popular in online advertising web sites [2]–[4], where the text-based visual CAPTCHAs are embedded into commercial videos.

As all other security techniques, the advent of CAPTCHAs has also driven numerous attempts by researchers and attackers to develop automated methods to defeat CAPTCHAs. Initial CAPTCHA schemes have been broken by different combinations of image processing and pattern recognition techniques [5]–[10]. For text-based visual CAPTCHAs, a widely-believed fact is that the CAPTCHAs need to be segmentation resistant because it has been shown that computers are better than humans in recognizing well segmented letters [11].

In recent years, several major web service providers including Google, Microsoft and Yahoo! have started deploying CAPTCHAs based on a technique called "crowding characters together" (CCT) [9], whose aim is to make automated segmentation difficult by making adjacent characters overlapped with each other. This turns out to be a good strategy to disable segmentation attacks, but unfortunately lowers the usability by confusing even advanced human users with difficult CAPTCHAs (see Fig. 1 for some examples from Google). One consequence of those hard CAPTCHAs is that annoyed human users may simply refresh the web page until an easy CAPTCHA appears. This implies that the actual hardness of the CAPTCHA scheme is reduced to a subset of easier (more usable) but likely weaker (less secure) CAPTCHAs that human users are happy to work with.

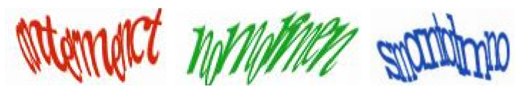

Fig. 1. Three hard Google CAPTCHA images: (a) "antermenct" or "anterrnenct"? (b) "nomorinen" or "nomormen"? (c) "smomtotmno" or "smorntotrnno"?

On the other hand, since it is difficult to find a good balance between security and usability, many web sites choose to deploy more usable but less secure CAPTCHAs to avoid driving away potential customers. For example, many financial institutions around the world have deployed weak e-banking

[1]In some applications, the bar can be lowered to somewhere below the rate of an average human solver.

CAPTCHAs in favor of usability (see Fig. 2 for some examples), but this inevitably makes these CAPTCHAs prone to attacks as reported in [10].

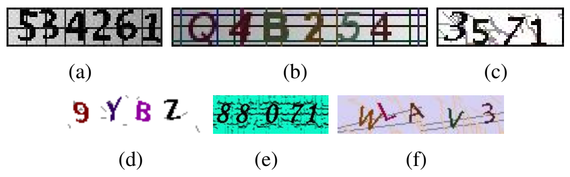

Fig. 2. Some selected weak e-banking CAPTCHAs broken in [10].

The security-usability dilemma has its roots in the difficulty of evaluating (or even defining) security and usability requirements of a CAPTCHA scheme towards the target application. As a consequence, balancing the delicate security-usability tradeoff in a CAPTCHA scheme remains an art rather than a science. This calls for research in automated security and usability analysis of CAPTCHAs. Once we have some automated tools to determine the security and usability of a CAPTCHA scheme, we will be able to configure it properly to fulfill some pre-defined requirements. The configuration can be dynamically adjusted to meet the change of requirements (such as the need to resist newly reported attacks), thus leading to *reconfigurable CAPTCHAs*.

The idea of reconfigurable CAPTCHAs has been partly adopted by some web sites. Microsoft produces CAPTCHAs from a multi-CAPTCHA engine, where the engine seems to be determined randomly without considering security. There is no public report about how Microsoft's multi-CAPTCHA engine works, but we doubt that there is any mechanism of evaluating security and usability – if there is indeed one, then only the best engine should be used, not all of them. A similar widely-adopted multi-CAPTCHA scheme is BotDetect CAPTCHA, a product of Lanapsoft, Inc. The BotDetect CAPTCHA supports 60 different images and 10 sound styles to meet the need of different web sites, and those styles can also be changed dynamically to have a flavor of reconfiguration. However, there is no evidence that Lanapsoft, Inc. has a quantitative measure of the security and usability of the 70 different configurations of BotDetect CAPTCHA since they claim that "*each of them is easily comprehensible to human users, randomly using multiple CAPTCHA generation algorithms makes the CAPTCHA challenge practically impossible to pass automatically.*" Without automated security-usability evaluation, reconfigurable CAPTCHAs are merely a collection of different CAPTCHAs, thus being less useful due to the vagueness in what a particular configuration can offer to us.

At the time of this writing, the only work related to automated CAPTCHA security-usability evaluation is our work-in-progress reported at SOUPS 2011 [12]. We are currently extending that work towards a general framework that can support reconfigurable CAPTCHAs. We call this framework Captchæcker, meaning "Captcha Checker". In the next section we present our main ideas on Captchæcker with preliminary and future work. The last section concludes this position paper.

II. CAPTCHÆCKER

Captchæcker is a framework involving a reconfigurable CAPTCHA engine, a security checker, a usability checker and a reconfigurator, as shown in Fig. 3. In the following, we describe each component of the framework.

A. *Reconfigurable CAPTCHA Engine*

As its name suggests, this engine should contain a number of parameters so that it can be reconfigured towards the desired security and usability requirements. This engine is equipped with a CAPTCHA database so that a particular CAPTCHA scheme can be selected with a particular set of parameters. It also includes a pseudo-random number generator (PRNG) so that the selected CAPTCHA scheme can be randomized if needed. The PRNG is also used to randomize the generation of CAPTCHAs out of the selected CAPTCHA scheme. The design loop may need to be repeated several times until a good balance between security and usability is found; each iteration has to be confirmed by the security and usability checkers.

B. *Security Checker*

To evaluate the security of a given CAPTCHA or a number of CAPTCHAs produced by a given CAPTCHA scheme, we need to automate the formation of CAPTCHA breaking algorithms. In other words, we need to make CAPTCHA breaking algorithms reconfigurable so that different editions can be created and tested on the input CAPTCHAs automatically. To achieve such a goal, we propose to build a CAPTCHA breaking tools library (CBTL), which should cover most of the commonly-used image processing and pattern recognition tools, such as those used in the CAPTCHA breaking network reported in [10]. An abstract description of each tool and its I/O behavior should be well defined so that a CAPTCHA breaking network can be automatically synthesized following a number of rules of synthesis. Considering the data-driven nature of any CAPTCHA breaking network, we propose to use a dataflow programming language to model the CBTL.

While there are many data-flow programming languages, we propose to use the recently standardized MPEG RVC (Reconfigurable Video Coding) framework for this purpose [13], [14]. The RVC standard [15], [16] defines an abstract language called RVC-CAL for modeling a functional unit (FU) and another language called FNL (FU network language) to describe the connections among FUs. One of the advantages of using the RVC framework is the possibility of having a platform-independent implementation of CBTL and then generating implementations of CBTL in different target programming languages (C, C++, Java, LLVM, VHDL/Verilog).

Given a CBTL, security evaluation becomes a process of looking for a network of CBTL tools that can break the CAPTCHAs under test with a non-negligible probability. Since the CAPTCHA engine knows the ground truth solutions to the CAPTCHAs, it can send them to the security checker without human involvement. The main parameter that needs to be set by a human user is the number of constraints defining which tools in CBTL can be connected with other tools,

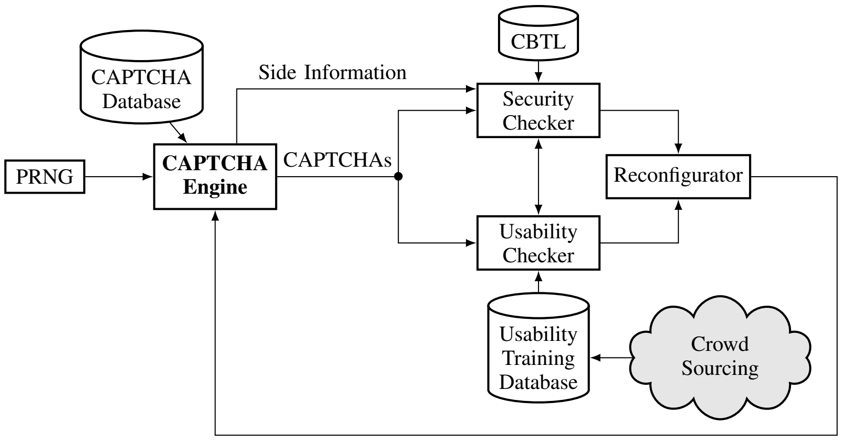

Fig. 3. The process of reconfiguring a CAPTCHA engine based on automated security and usability analysis of the generated CAPTCHAs.

and how many instances one can have for each tool. Such a descriptive language is not provided by the RVC framework, so we need to develop it based on the FNL. In addition, the human user also needs to set a threshold of success rate as the stopping criterion. If all possible configurations of the CAPTCHA breaking network have been tested but none of them reached the pre-defined threshold, we can assume that the tested CAPTCHA scheme is secure under the current input connection constraints. While this should not be considered as an absolute proof of the security of the CAPTCHA scheme, it can be treated as a strong evidence of it.

Since some pattern recognition tools are supervised, a training set will be needed to train the classifier. This can be a difficult task for an automated security checker. Fortunately, since we are standing at the position of a CAPTCHA designer rather than a CAPTCHA attacker, the training set can be automatically built from the CAPTCHA engine, which makes it possible to fully automate the whole process.

Another important aspect is the computational complexity of the successful attack (if any). If one attack is computationally too heavy, it may not be practical in solving CAPTCHAs in many real-world applications, e.g. abusing a public email service to register a large amount of email accounts for sending spam emails. While the complexity can be estimated directly from the running time, we can also estimate it from pre-profiled complexity of each tool and then calculate according to our knowledge of the network topology. Since the profiling of all CBTL tools only needs to be done once, this will cause only a negligible overhead on the normal process of

One interesting point about the security checker is that it inherently defines a mechanism that can be used by an attacker to automatically figure out an attack or evolve his existing attack (if the pattern recognition tools have been trained). This is not surprising since the nature of a security checker implies that it has to simulate what an attacker will do in reality.

C. Usability Checker

Automated usability checking is a more challenging task as compared to automated security checking because we have to find a way to model humans. To avoid this difficulty, we propose to exploit the wisdom of the crowd, i.e, to depend on crowdsourcing to build ground truth databases that can then be used to train a supervised classifier. In recent years, crowdsourcing has become particularly popular in research on usable security because many crowdsourcing web services offer an API that makes it possible to include humans in the loop so that the system can run fully automatically as if no human is involved [17]. One of the main drawbacks of using crowdsourcing is the additional costs one has to pay. While the cost per human intelligent test is indeed very small (can be as low as 0.01 USD), the need of labeling a large number of training samples still requires a considerably amount of budget to support the system. Another downside of crowdsourcing is that the performance of paid crowdsourcing is not high enough even after strict qualification testing [18]. This means that we will need a significant number of human solvers to make a vote, thus increasing the cost significantly.

In case the budget does not allow paid crowdsourcing, one can try to leverage the user base of one or more large universities (like ours). This can be done by developing a web CAPTCHA module and embedding it into the online course management systems (CMSs) and/or intramural web forums, which may be done by negotiating with the IT services of the universities without paying any fee.

In the usability training database, we should record both subjective evaluation of usability and some objective metrics that can indirectly reflect the usability of tested CAPTCHAs. The former can simply be a level of hardness returned by the user and the latter can include the average response time and the average response error. It is likely that the user's subjective evaluation does not fully match the hardness reflected by the objective metrics. Therefore, the best ground truth may not be the subjective evaluation, but the objective one.

Once we get a database with usability data of a large number of CAPTCHAs, the usability checker can be established by training a classifier. One may wonder if human users can make consistent evaluation on CAPTCHA hardness so that their evaluations can be learned. Our preliminary work on a small database and a few typical text-based visual CAPTCHA schemes has given an affirmative answer [12]. In that work, we collected subjective hardness scores from 20 users on 50 CAPTCHAs, which were selected from four typical text-based visual CAPTCHA schemes: Google CAPTCHA, Google reCAPTCHA, one Microsoft CAPTCHA with two rows, and Yahoo! CAPTCHA. Some examples of Google CAPTCHA have been shown in Fig. 1, and samples of the other three CAPTCHA schemes are shown in Fig. 4. By extracting a few simple geometric features from each CAPTCHA, we found out that two combined features can lead to an NN classifier with an accuracy above 80% on a testing set with 38 new CAPTCHAs and five new users. Based on this encouraging result, we are now collecting more CAPTCHA samples and will run a larger-scale user study via Amazon Mechanic Turk to build a larger subjective database.

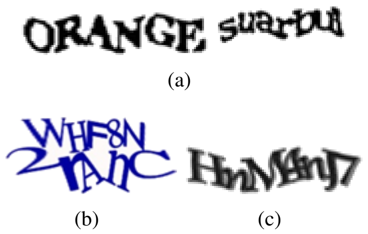

Fig. 4. Examples of (a) Google reCAPTCHA, (b) Microsoft CAPTCHA, and (c) Yahoo! CAPTCHA tested in [12].

It is noteworthy that different CAPTCHA schemes will likely require different combinations of features to allow a good classification result. Hence we are also working on categorization of text-based visual CAPTCHAs and different geometric features so that we can have a list of features whose different combinations will help classify all the CAPTCHAs in our test set. Another issue is about possible pre-processing on visual CAPTCHAs before feature extraction, which is necessary because some CAPTCHA schemes introduce background noises that can distract the feature extraction algorithm.

D. Reconfigurator

The reconfigurator takes the quantitative scores from the security and usability checkers, and then tries to make recommendations on how to reconfigure the CAPTCHA engine to achieve a better balance between security and usability.

In principle, the automated reconfiguration is an optimization problem, where the objective is to minimize the distance to the target security and usability requirements and the constraints are the bounds of all the parameters. Since the security and usability checker cannot be formulated analytically, we have to resort to an evolutionary strategy such as genetic algorithm or particle swarm optimization.

If the automated reconfiguration turns out to be impractical, the reconfigurator can try to distinguish and visualize key CAPTCHA-breaking tools that cause the insufficient security level, and leaves the final design changes to the human designer. In this case, the reconfigurator will become an interface for computer assisted CAPTCHA re-design.

III. Conclusion

This position paper briefs our ongoing work on reconfigurable CAPTCHAs based on automated security and usability analysis, which we call *Captachæcker*. This concept will provide CAPTCHA designers with a systematic tool for analyzing security and usability of their existing CAPTCHA schemes. It will also create a computable framework that can dynamically update a live CAPTCHA scheme that need to adapt with evolving security and usability requirements of a deployment.

References

[1] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in *Advances in Cryptology – EUROCRYPT 2003*, ser. Lecture Notes in Computer Science, vol. 2656. Springer, 2003, pp. 294–311.

[2] Leap Marketing Technologies Inc., "NuCaptcha," http://www.nucaptcha.com, 2011.

[3] ADSCAPTCHA, Ltd., "ADSCAPTCHA," http://www.adscaptcha.com, 2011.

[4] CaptchaAd GmbH, "CaptchaAd," http://www.captchaad.com, 2011.

[5] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA," in *Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2003)*, vol. 1. IEEE Computer Society, 2003, pp. 134–141.

[6] G. Moy, N. Jones, C. Harkless, and R. Potter, "Distortion estimation techniques in solving visual CAPTCHAs," in *Proceedings of 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2004)*, vol. 2. IEEE Computer Society, 2004, pp. 23–28.

[7] S. Hocevar, "PWNtcha: Pretend we're not a Turing computer but a human antagonist," http://caca.zoy.org/wiki/PWNtcha, 2004.

[8] J. Yan and A. S. E. Ahmad, "Breaking visual CAPTCHAs with naïve pattern recognition algorithms," in *Proceedings of 23rd Annual Computer Security Applications Conference (ACSAC'2007)*. IEEE Computer Society, 2007, pp. 279–291.

[9] ——, "A low-cost attack on a Microsoft CAPTCHA," in *Proceedings of 15th ACM Conference on Computer and Communications Security (CCS'2008)*. ACM, 2008, pp. 543–554.

[10] S. Li, S. A. H. Shah, M. A. U. Khan, S. A. Khayam, A.-R. Sadeghi, and R. Schmitz, "Breaking e-banking CAPTCHAs," in *Proceedings of 26th Annual Computer Security Applications Conference (ACSAC'2010)*. ACM, 2010, pp. 171–180, http://www.hooklee.com/default.asp?t=eBankingCAPTCHAs.

[11] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski, "Computers beat humans at single character recognition in reading based human interaction proofs (HIPs)," in *Proceedings of 2nd Conference on Email and Anti-Spam (CEAS'2005)*, 2005.

[12] M. Nazir, Y. Javed, M. M. Khan, S. A. Khayam, and S. Li, "Poster: Captchæcker – automating usability-security evaluation of textual CAPTCHAs," in *Proceedings of 7th Symposium On Usable Privacy and Security (SOUPS'2011)*. ACM, 2011.

[13] M. Mattavelli, I. Amer, and M. Raulet, "The Reconfigurable Video Coding standard: [standards in a nutshell]," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 159–167, 2010.

[14] S. Bhattacharyya, J. Eker, J. W. Janneck, C. Lucarz, M. Mattavelli, and M. Raulet, "Overview of the MPEG Reconfigurable Video Coding framework," *Journal of Signal Processing Systems*, vol. 63, no. 2, pp. 251–263, 2011.

[15] ISO/IEC, "Information technology – MPEG systems technologies – Part 4: Codec configuration representation," ISO/IEC 23001-4, 2009.

[16] ——, "Information technology – MPEG video technologies – Part 4: Video tool library," ISO/IEC 23002-4, 2009.

[17] B. Frei, "Paid crowdsourcing: Current state & progress toward mainstream business use," http://www.smartsheet.com/files/haymaker/PaidCrowdsourcingSept2009-ReleaseVersion-Smartsheet.pdf, 2009.

[18] P. Wais, S. Lingamneni, D. Cook, J. Fennell, B. Goldenberg, D. Lubarov, D. Marin, and H. Simons, "Towards building a high-quality workforce with Mechanical Turk," in *NIPS 2010 Workshop on Computational Social Science and the Wisdom of Crowds*, 2010, http://www.cs.umass.edu/~wallach/workshops/nips2010css/papers/wais.pdf.