UserTrust

Allen Madsen

Computer Science

B. Thomas Golisano College of Computing and Information Sciences

August 27, 2010

Abstract

TODO: Abstract

I. Introduction

Work in progress. Ellipses are used to show sections that need to be filled in.

In 1950, Alan Turing wrote a paper called *Computing Machinery and Intelligence* [17]. In this paper he described a test, now called the Turing test, which tests a machine's ability to behave intelligently. The test is set up as an imitation game with three players: a human interrogator, another human, and a machine. All the participants are separated from each other and communicate through a computer terminal. The human interrogator is responsible for deciding which of the other two participants is a human by asking questions and viewing their responses. If the interrogator cannot distinguish accurately between the human and machine, then the machine is behaving intelligently.

Luis von Ahn et. el. [18] proposed a modification to the Turing test in 2003 called the *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA). A CAPTCHA system takes advantage of the fact that machines cannot yet pass the Turing test to differentiate between humans and machines. In the context of the Turing test a CAPTCHA system fills the role of the interrogator by generating a question only a human should be able to pass and grading the answer the user provided to the question. When a user attempts to access some resource, the CAPTCHA system does not know if the user is a human or machine. To differentiate the two, the CAPTCHA system asks a difficult artificial intelligence (AI) problem

that is known to be hard for machines to solve, but simple for humans. Therefore, it is reasonable to assume that if the AI problem is solved, the user is a human. For a CAPTCHA system to be effective, four things must be true about the AI problem used:

- It should be simple to generate questions.
- It should be simple to grade questions.
- It should be difficult for machines to solve.
- It should be simple for humans to solve.

Since CAPTCHA systems can differentiate between humans and machines, they are useful for preventing automation. Preventing automation can provide benefits for messaging systems (emails, comments, tweets),

...

Another goal of a CAPTCHA system is to provide motivation to solve difficult AI problems. There are notable examples [2], [19], [23] of this, where the AI problem of prominent CAPTCHA systems have been solved. The typical response [8], [14] to a broken CAPTCHA is to make a more difficult CAPTCHA. This can be problematic though, because it only addresses the need for a problem machines cannot solve and ignores the need for humans ability to solve it simply [9]. As a result the concept of usable CAPTCHAs [10], [21], [22], [24] have evolved with the aim to keep the original properties of CAPTCHAs; humans should have a high rate of success and computers should have a low rate of success. Yan and Ahmed [24], propose three criteria for determining the usability of a CAPTCHA:

- **Accuracy**: How often a user is able to successfully complete a CAPTCHA.
- **Response time**: How quickly a user is able to successfully complete a CAPTCHA.
- **Perceived difficulty**: How difficult a user perceives a CAPTCHA to be.

These are useful, because they give specific goals for reducing the amount of interruption in a users workflow. However, they focus on the experience of a single CAPTCHA. A site is likely to employ CAPTCHAs for multiple operations. Regardless of how usable a CAPTCHA is, the compounded effects of multiple CAPTCHAs will decrease a users efficiency on a site. Because of the disparity in addressing the number of occurrences of CAPTCHAs, I propose a system called UserTrust which can be used in conjunction with current CAPTCHA systems to improve usability.

UserTrust will be a centralized system that acts as an intermediary between a site and a CAPTCHA service. When a site wants to know if a user is a human it will query UserTrust. If the user has a sufficient reputation of passing CAPTCHAs, then there is no need for her to fill out another and UserTrust will tell the site a CAPTCHA is not necessary. However, if the user does not have a high reputation UserTrust will tell the site a CAPTCHA is necessary. After the user has attempted to pass the CAPTCHA there answer is sent to UserTrust, which verifies it with the CAPTCHA service, and returns the response to the site. UserTrust also records the outcome of the CAPTCHA and uses that to calculate the reputation for that user. Over a set of CAPTCHA tests UserTrust will be able to tell whether a user is a human or bot. Also, since UserTrust is centralized, it can take advantage of CAPTCHA histories across all the sites a user belongs to for the calculation.

There are two levels of attacks that can be performed against UserTrust. The first is at the user level, where a user behaves in a strategic manner to obtain a high reputation. With a high reputation the strategic user would then be able to exploit the relaxed security of UserTrust. Srivatsa, Xiong, and Liu [15]

...

Hypothesis: It is possible to increase usability and maintain security of websites when reducing the number of CAPTCHAs shown by tracking user performance on CAPTCHAs and extracting information from that data.

In order for UserTrust to be considered successful, it should meet the following properties:

Open: Following Kerschoff's Principle, the algorithm used to generate reputations should be freely available. The public availability of the algorithm should not affect the security of the UserTrust as long as the key (the complete set of user histories) is unknown.

Secure: UserTrust should be resilient against attacks from strategic users and sites that produce false feedback. In most circumstances UserTrust should correctly identify a user as a bot or a human.

Performant: UserTrust should be able to meet the demand of calculating reputations. It should not be possible, for example, to have a failed CAPTCHA test queued up for inclusion so long that it affects UserTrust's ability to be secure.

Usable: A normal user should see a significant decrease in CAPTCHAs shown after a

history of passing CAPTCHAs is established.

...

UserTrust assumes that the CAPTCHA service being used is secure and trustworthy. This is necessary because of the central role that CAPTCHAs play in this system. If it were possible for a bot to achieve a high level of accuracy in completing CAPTCHAs it would make it easy to obtain a high reputation and undermine the security of the system. UserTrust also makes the assumption that it is hard to discover a large amount of UserTrust user identifiers on a site. If this were possible, it would be much easier to produce a high similarity to another site and influence the ratings of users on that site. In general this should be hard to do unless a site were to explicitly publish these, which would not be in the best interest of the site.

In Section II, related works are discussed. These include work related to CAPTCHAs, Reputation Systems, and Collaborative Sanctioning. Section III describes how websites will interact with UserTrust. Section IV talks about the algorithms to be used, why they were selected, and how the system is to be implemented.

II. Background

A. CAPTCHAs

B. Generating Reputations

C. Filtering Feedback

III. Functional Specification

TODO

IV. Design Specification

TODO

V. Ignore This

However, the only way they can do that is with a CAPTCHA which is cumbersome for their normal users to solve [4], [10], [24].

Adopting either of these views represents two extremes on a scale. They can be stated as always showing and never showing a CAPTCHA on a form. It would be nice if a middle ground could

be found where a CAPTCHA is only shown sometimes, because this would allow a balance between security and the user experience. A nave approach would be to show a CAPTCHA randomly. However, this method is insufficient because a computer could be designed to refresh a page with a form until there is no CAPTCHA. This method is also problematic because it still treats the normal user the same as the automated user. The ideal approach would be to always show a CAPTCHA to automated users and never show one to normal users. In order to do this, more information is needed to differentiate between the two types of users.

The type of information that would be useful is information that can be used to predict the future performance of a user on a CAPTCHA. If it is known that a user will likely pass a CAPTCHA if shown one, then it is a reasonable conclusion to not show one. A users history of previous performance on CAPTCHAs is a good indicator of how they will perform in the future. From this history can be defined as:

$H = (T_1, T_2, \ldots, T_n)$, where $n \geq 1$

$$T_k = \begin{cases} 1, & \text{if } CAPTCHA_k \text{ passed} \\ 0, & \text{if } CAPTCHA_k \text{ failed} \end{cases}$$

, where $1 \leq k \leq n$

There is one problem inherent with a CAPTCHA that needs to be addressed, however. Passing a CAPTCHA provides reasonable certainty that the user is not automated, however, failing a CAPTCHA does not necessarily mean a user is automated. It is not uncommon for a normal user to incorrectly fill out a CAPTCHA. In this case a site would typically retest the user with another CAPTCHA. This lends itself to the idea of a session where a user can attempt to pass a CAPTCHA multiple times until they succeed or give up. Thus a session is defined as:

$S = (T_1, T_2, \ldots, T_n)$, where $n \geq 1$

$T_k = 0$, where $n > 1$ and $1 \leq k < n$

$$T_n = \begin{cases} 1, & \text{if } CAPTCHA_k \text{ passed} \\ 0, & \text{if } CAPTCHA_k \text{ failed} \end{cases}$$

Before session is incorporated into history it is important to decide how a sessions value is determined. A very simplistic approach would be to use the value of T_n for each session.

This approach neglects to account for the number of tries a user performs before they succeed. This information is important because a normal user should complete a CAPTCHA correctly in relatively few tries, whereas an automated user may be able to complete a CAPTCHA correctly after many tries. Another approach would be to use an average; however, this may not punish the automated user enough. Instead, a weighted average can be used, where p^{k-1} is used as the weight and $0 < p \leq 1$. Thus we get the equation:

$$SV = \sum_{k=1}^{n} T_k * \frac{p^{k-1}}{\sum_{k=1}^{n} p^{k-1}}$$

Since all $T_k = 0$ where $n > 1$ and $1 \leq k < n$, the above equation can be simplified to:

$$SV = T_n * \frac{p^{n-1}}{\sum_{k=1}^{n} p^{k-1}}$$

REFERENCES

[1] Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. on Knowl. and Data Eng.*, 16:843–857, July 2004.

[2] Streamlined anti-captcha operations by spammers on microsoft windows live mail, febuary 2008.

[3] Captcha: Telling humans and computers apart automatically, Jan 2011.

[4] N. Ben-Asher, J. Meyer, S. Moller, and R. Englert. An experimental system for studying the tradeoff between usability and security. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pages 882 –887, march 2009.

[5] J. Delgado. Memory-Based WeightedMajority Prediction for Recommender Systems, 1999.

[6] E. Folmer and J. Bosch. Architecting for usability: a survey. *Journal of Systems and Software*, 70(1-2):61 – 78, 2004.

[7] T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. Certified reputation: how an agent can trust a stranger. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 1217–1224, New York, NY, USA, 2006. ACM.

[8] I. F. Ince, I. Yengin, Y. B. Salman, H.-G. Cho, and T.-C. Yang. Designing captcha algorithm: Splitting and rotating the images against ocrs. In *Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology - Volume 02*, pages 596–601, Washington, DC, USA, 2008. IEEE Computer Society.

[9] M. Jakobsson. Captcha-free throttling. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, AISec '09, pages 15–22, New York, NY, USA, 2009. ACM.

[10] K. A. Kluever and R. Zanibbi. Balancing usability and security in a video captcha. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, pages 14:1–14:11, New York, NY, USA, 2009. ACM.

[11] Z. Malik and A. Bouguettaya. Rateweb: Reputation assessment for trust establishment among web services. *The VLDB Journal*, 18:885–911, August 2009.

[12] S. Park, L. Liu, C. Pu, M. Srivatsa, and J. Zhang. Resilient trust management for web service integration. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '05, pages 499–506, Washington, DC, USA, 2005. IEEE Computer Society.

[13] J. D. Sonnek and J. B. Weissman. A quantitative comparison of reputation systems in the grid. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, GRID '05, pages 242–249, Washington, DC, USA, 2005. IEEE Computer Society.

[14] V. Srikanth, C. Vishwanathan, U. Asati, and N. C. S. N. Iyengar. Think-an image based captcha mechanism (testifying human based on intelligence and knowledge). In *Proceedings of the International Conference on Advances in Computing, Communication and Control*, ICAC3 '09, pages 421–424, New York, NY, USA, 2009. ACM.

[15] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 422–431, New York, NY, USA, 2005. ACM.

[16] M. Tennenholtz. Reputation systems: an axiomatic approach. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, pages 544–551, Arlington, Virginia, United States, 2004. AUAI Press.

[17] A. M. Turing. Computing Machinery and Intelligence. *Mind*, LIX:433–460, 1950.

[18] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: using hard AI problems for security. In *Advances in cryptology—EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Comput. Sci.*, pages 294–311. Springer, Berlin, 2003.

[19] J. Wane. Yahoo! captcha is broken, january 2008.

[20] A. Whitby, A. Jsang, and J. Indulska. Filtering out unfair ratings in bayesian reputation systems. 2004.

[21] T. Yamamoto, T. Suzuki, and M. Nishigaki. A proposal of four-panel cartoon captcha: The concept. In *Proceedings of the 2010 13th International Conference on Network-Based Information Systems*, NBIS '10, pages 575–578, Washington, DC, USA, 2010. IEEE Computer Society.

[22] T. Yamamoto, J. D. Tygar, and M. Nishigaki. Captcha using strangeness in machine translation. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, AINA '10, pages 430–437, Washington, DC, USA, 2010. IEEE Computer Society.

[23] J. Yan and A. S. El Ahmad. A low-cost attack on a microsoft captcha. In *Proceedings of the 15th ACM conference on Computer and communications security*, CCS '08, pages 543–554, New York, NY, USA, 2008. ACM.

[24] J. Yan and A. S. El Ahmad. Usability of captchas or usability issues in captcha design. In *Proceedings of the 4th symposium on Usable privacy and security*, SOUPS '08, pages 44–52, New York, NY, USA, 2008. ACM.