UserTrust

Allen Madsen

Computer Science

B. Thomas Golisano College of Computing and Information Sciences

August 27, 2010

Abstract

TODO: Abstract

I. Introduction

A Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is a test which uses a difficult artificial intelligence problem to distinguish between a human and a computer. The most common form for a CAPTCHA is an image with obscured or transformed text that the user must decipher. This type of CAPTCHA is effective because it is difficult for computers (bots) to recognize letters that are distorted, whereas humans find it much easier. As a result, this sort of test is useful for defending various types of systems from automated usage by computers [12].

For example consider a website called Snapshare where pictures can be shared. Only users with accounts on Snapshare can upload and comment on images. However, an account is not needed to view images and comments. Eve has noticed the popularity of the site and decided to exploit it to advertise her own site. To do this she writes a bot that automatically creates accounts, uploads images with a url to her site, and creates comments with a link to her site. Eventually, Snapshare's operators starts receiving complaints from normal users about Eve's abusive behavior. They decide to delete Eve's activity and add CAPTCHAs to user registration, image uploading, and commenting. Eve's bot is useless now since it is unable to pass a CAPTCHA.

Snapshare's operators are happy about the results, because they stop seeing Eve's spammy messages and complaints about them. However, their victory is short lived. A couple days after adding CAPTCHAs Snapshare's operators start receiving complaints from their users about the CAPTCHAs. Bob, a common user of the site, does not like that he has to fill out a CAPTCHA for every action on the site and is threatening to switch to another service. Snapshare is in a conundrum because they cannot just remove the CAPTCHAs. That would allow Eve to start spamming the site again which the normal users also did not like. One approach they consider is to remove the CAPTCHAs from the most common actions (uploading and commenting) and leave a CAPTCHA on registration. They abandon this idea because Eve could register an account manually and produce a lot of spam before the operators could stop it.

What Snapshare's operators would like to do is stop spamming on their site while minimizing the negative impact on usability for normal users. In order to meet this goal, UserTrust attempts to intelligently guess if a user is a bot by tracking user performance on CAPTCHAs. Suppose Bob gradually builds a history of passing CAPTCHAs. Eventually he will develop a high trust reputation. With a high trust reputation Snapshare is fairly certain that Bob is not a bot and is going to decrease the amount of times Bob needs to fill out a CAPTCHA. Both Bob and Snapshare benefit from this system. Bob sees less spam because bots must fill in CAPTCHAs and he sees less CAPTCHAs because he has built a good reputation. Snapshare has made Bob happy which means he will continue to use the site.

Bob is a very active user on the web and Snapshare is not the only site he has to fill out CAPTCHAs on. In order to improve Bobs experience UserTrust also considers the history of CAPTCHAs from all the sites that Bob belongs to that also use UserTrust. This allows Bob to carry part of his reputation to each site so that he does not have to build a new reputation for every site he joins. Snapshare and other sites like it are motivated to contribute to UserTrust because of the benefits it produces for their users.

Eve has not given up. She would still like to advertise her own site on Snapshare. To do this Eve has two levels of attacks. At one level she can act as a strategic user and try to take advantage of how UserTrust generates reputations. For example, she could alternate behavior between her using an account and her bot using the account. During the time Eve uses the account she is able to build a good reputation. When she allows the bot to use the account it is able to operate freely until it is presented with a CAPTCHA again. To address this attack two approaches are used. The first is to make it expensive to achieve a good reputation so that the effort involved is not worth the payoff gained when a high reputation is achieved. As an arbitrary example, a user may need to pass one hundred CAPTCHAs to achieve a high reputation. Since the point of a bot is to automate tasks, this amount of manual work can be prohibitive. The second approach is to make the amount of time the bot is allowed to operate freely short lived. Suppose a bot is able to make a hundred comments before it is presented with a CAPTCHA. On sites with an active community, this amount of comments is likely a drop in the bucket compared to how many comments are typically generated. As a result the payoff when a reputation is high may not be worth the effort to achieve the high reputation.

Eve's other level of attack is to operate a website or many websites and provide false feedback about a user. Eve's idea here is that if she repeatedly says an account has passed a CAPTCHA it will be able to gain a high reputation without actually completing any CAPTCHAs. This would subvert the goal of making a high reputation costly to achieve. UserTrust proposes two strategies to mitigate this risk. First, UserTrust would act as a middleman in the CAPTCHA verification process. For example, Snapshare would forward the users answer to a CAPTCHA to UserTrust, which would verify if the CAPTCHA with the CAPTCHA service and then return the response to Snapshare. This allows UserTrust to know the CAPTCHA service is reliable and that feedback is not being forged to obtain a high rating. Eve is still able to produce a low rating by submitting incorrect CAPTCHA solutions, though the utility of that is questionable. However, perhaps Eve has collected a farm of people who, knowingly or not, can solve CAPTCHAs for her in high quantity. In this case an additionally strategy is employed. Since a site may be untrustworthy it is useful to not treat all sites equal when generated a reputation. In order to rank how trustworthy a site is a similarity measure is used that considers the similarity between users on two sites and the number of similar users they share. Using a threshold for the amount of similarity required produces a cluster for each site that can be used to generate a reputation for a particular user. This makes it exceptionally hard for Eve to influence the reputation of a user on a particular site because she must replicate the behavior of a lot of users to achieve a similarity higher than the threshold.

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 behavior 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 [2], [6], [14].

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] 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, 2009.

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

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.