A Solver-Resistant Challenge Response Spam Protection System

Alexander Pyattaev
Tampere University of Technology
Korkeakoulunkatu 1, Tampere, Finland
alexander.pyattaev@tut.fi

Vladimir Sadovnikov
Saint-Petersburg University of Telecommunications
Moika 61, Saint-Petersburg, Russia
sadko@skri.sut.ru

Abstract

The original challenge response systems like popular CAPTCHA have been made to protect against automated spam bots. They worked exceptionally well, so the spammers had to find another way to get through. Currently, they no longer have to employ computer systems to crack challenges. It is in fact easier to employ lots of people to solve challenges instead of developing a program that would do that automatically. Those people may be, for example, forced to do that by virus software. There are also commercial crowd-sourcing services providing capacities of up to 1000000 solutions per day. The price tag on such services is typically about $1 per 1000 solutions. This situation makes conventional spam protection systems like CAPTCHA useless, since the attackers are humans themselves. At the same time, modern OCR systems sometimes have better recognition success rates than real humans on the conventional distorted text challenges. As those become widely available, classic challenge response systems would become completely useless.

In this paper authors address the problem with novel approach, aiming at three goals: make it impossible to devise a reliable algorithm for automated solving, make the process as hard as possible for CAPTCHA-solvers and try to make it easier for the legitimate users. The novel approach is based on dynamic content capabilities of HTML5 that allow creation of interactive challenges that are easy yet time consuming to solve for humans and are close to impossible for bots. Unlike most semantics-based challenges, the proposed one is resistant to dictionary attacks since the challenge data is generated randomly.

Index Terms: CAPTCHA, challenge response, spam, web, OCR, usability.

I. Introduction

The problem of spam appeared long time ago, and at first was only important in context of electronic mail. The spam-protection systems have even been standardized [1]. With introduction of Web2.0 and dynamic content, the spammers recognized the potential of the new platform and began systematic poisoning of forums and blogs with advertisements. An end was put to this with introduction of ”Completely Automated Public Turing test to tell Computers and Humans Apart”, widely known as CAPTCHA [2]. The idea of baseline CAPTCHA test is to distort a portion of text in such a way, that *optical character recognition* (OCR) systems have trouble recognizing it, while humans can do that relatively effortlessly. For long time the only real attack against CAPTCHA was possible by introduction of better OCR system. There exist also a wide group of similar algorithms, all together called *challenge response* (CR) spam filters.

Nowadays, web administrators face new problems. First, the OCR systems are now capable of cracking many CAPTCHA systems [3],[4]. Second, the crowd-sourcing based CAPTCHA solving companies are ready to put their effort into breaking challenges almost for free. Another attack vector are bot nets like KOOBFACE, which make users of infected PC’s

solve challenges [5]. Finally, users themselves are getting tired of the challenges [6], since they do not really provide protection, yet take time to solve. As a result, administrators tend to disallow any sort of content upload without prior registration. As a result, users leave the site without commenting, or need to register using email as validation method. As web administrators assume that mail accounts are authentic, they effectively relay all the bot-filtering to email providers, who, in turn, are forced to send SMS to verify user's identity. Apart from being expensive, this is rarely feasible in worldwide services. So email providers keep on using old CAPTCHA and delete thousands of spam accounts daily.

Observing the situation depicted above, we have decided that the main problem of conventional CR systems is their static nature. Any conventional CR system can be separated into two parts:

- The question part - a human-understandable representation of the task that has to be performed. Typically there is only one question, some systems employ a dictionary of tasks, which has at most Q questions that can be selected randomly.
- The medium part - a media that transfers the actual challenge data. In most cases, it is a random text or image, distorted or obfuscated, that should be analyzed. Let us call this case recognition challenge. Second option might be a set of predefined answers, that has total size of A, a small portion of which is displayed. Third option is a logical or mathematical expression that should be solved.

Typical attacks concentrate on the following drawbacks of the design scheme presented above:

- The question part has just Q question variants, so an algorithm could be made for each of those. In most cases making question logic is not easier than making answering logic, so attack is almost always possible.
- If medium is recognition-based, then an algorithm could be made to extract the original text or image from the distorted version. In some cases a good filter may be capable of recognition beyond human capabilities, and overall usability of such approach is questionable [7].
- If medium is based on predefined answers, like in [8], then all A of then could be pre-analyzed by humans, and then used by bots to find correct solutions. Since the size of the problem is relatively small, this does not present any problem in the long run.
- In case of mathematical task, a general-purpose symbolic solver could be used to produce solutions much faster than any human.
- Since the server has quite big timeout, it is possible to outsource the problem to human solvers. Reduction of server timeout leads to increased difficulty for normal users and false positives.

To circumvent the shortcomings of classic schemes, we propose a system with new sort of medium, that is not based on distortion. Humans can easily process motion in 3D space, and can track complex shapes in real-time, they can also infer scene segmentation based on relative motion of its parts. Computers, on the other hand, can not easily extract depth information from flat picture and have difficulties tracking animated objects, especially when they intersect with each other. Segmentation of 3D scene, based on relative motion of flat projections, without any extra data like shadows or lighting, seems to be tough enough to fend off any OCR for a while.

As a result of the considerations above, our system is an interactive animation, representing a 3D space with several objects. The scene is represented in such a way, that no information

about depth can be inferred from flat representation. The questions asked require matching of the events in the animation with location of those events. Since the target events do not happen every frame, this effectively delays any possible solution by some time, that can be configured. At the same time, the challenge requires attention even if the target event has not yet happened. As a result, the crowd-sourcing systems, especially commercial ones, should experience performance degradation due to exhaustion of workers. An OCR system would need to analyze each frame and decide if the click event should be sent, and where it should happen. This makes any kind of brute-force solutions useless, and raises requirements for a real analyzer, since it would have to reconstruct the complete scene before sending the reply. This also has to be done in real-time, and thus requires massive computational power.

In this paper, the authors present the designed algorithm in detail, explain its operation and possible expansion capabilities. Part II concentrates on the semantics employed and some security considerations. Since a prototype system was developed, its implementation is discussed in section III. In section IV, possible questions and high-level filtering strategies are discussed.

II. Challenge algorithm design

As in any other CR test the procedure consists of request-response sequence. In our scheme, we use N consequential requests, each to be replied correctly. in order and within allocated time, denoted T. The next request is not available before previous one is answered. The correctness of the whole session is determined after the test is complete. So the test may be set up to take a specific amount of time $T_0 = NT$. This allows some limited control of crowd-sourcing based attacks. Sources in the Internet indicate the average solving speed of about 800 solutions per hour [9], so we can assume that average challenge takes about 4 seconds to solve and input the correct answer for a trained person. An untrained person would, on the other hand, spend about 20 to 30 seconds while solving the test. This means that if our test forces the solver to spend at least 20 seconds it would not affect normal users, but would definitely make commercial solving 5 times more expensive. While not providing a radical solution to the problem, this approach could at least reduce the impact of such businesses. The Algorithm 1 summarizes the operation principle.

In our challenge the task is to analyze a three-dimensional animated scene. The scene is rendered on a server as wireframe, and then converted to flat drawing commands sent to client. Each object is generated according to some algorithm and is unique, one may observe object realizations in Figure 1. The absence of textures and randomization of the object shapes and rotations should make it harder for automatic system to detect and track them.

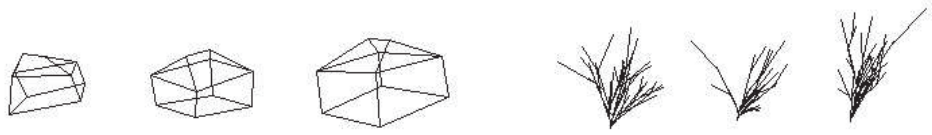

Fig. 1. Example objects - trees and houses

Unlike most common graphical CAPTCHA systems that are based on image distortion and noise, our system resembles hash-function, in a sense that once 3D wireframe scene

Algorithm 1 Test operation algorithm

```
Send the web form to be filled to the user
Initialize test queue of N tests
while Test queue not empty do
  Pop the next test from the queue and create scene
  Send the scene rendering script to the user
  Wait for the reply with timeout T
  if The answer is received in time then
    if The answer is correct then
      Record completion
    else
      Record failure
    end if
  else
    Assume the test was too complicated and add new test to the queue
  end if
end while
if The amount of correct answers is sufficient then
  Report success
else
  Report failure
end if
```

is rendered onto a flat surface, the depth information is lost in a computer-graphic sense. Humans, on the other hand, can easily reconstruct it by using external data, which in our case is relative movement of the object's projections. One may observe a sample picture in Figure 2.

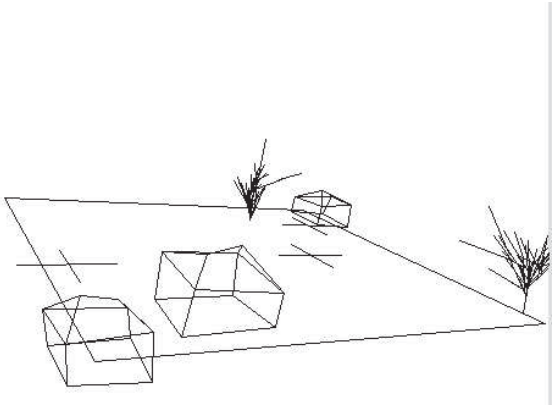

Fig. 2. Example of wireframe rendering

Since the depth information is still present in semantic sense and can therefore be extracted by humans, the questions are asked about the relative depth of a some objects. The scene is displayed set in random rotation with random deformation, which is not controlled by user. By observing the scene the user has to figure the answers to the questions and click at the

object location that seems to be the answer spot. The answer is encoded as a data tuple of coordinates and time of a mouse click. If the area occupied by the challenge is S by S pixels (let $S = 500$), and the correct answer region has radius of $R = 20$ pixels, the total probability of a blind guess in any particular frame is $p = \pi R^2/S = 0.005$. So for a test with 3 questions the probability of blind guess is $p_g = p^3 = 1.27e - 7$, which rules out brute force approach.

The scene rendering in Figure 2 is in fact not suitable for tricking the bots. The problem lies in the bounding box around the objects that allows the attacker to reconstruct the actual transformation that was performed while rendering the picture. So the actual picture that would be shown to the user should not include that box, and should have more objects distributed in such a way that most of them intersect with each other. In this sense Figure 3 is a lot better, since it is difficult to distinguish which line belongs to a particular object. The task could is made even harder by avoiding intersections of the lines belonging to the same object. A tiny offset is made for each object to break the links between lines. While not necessarily visible to humans, it results in unstable connections of the wireframe edges.

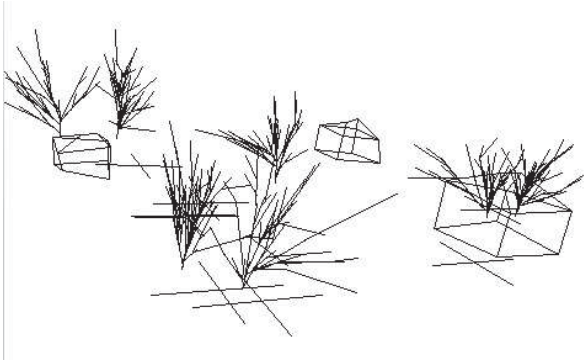

Fig. 3. A more segmentation-resistant rendering

III. The system implementation

In this part we will concentrate on implementation specifics. Any web system should have high coverage and should not employ non-standard technologies. Luckily for us, the system described above is completely standard. The server part can easily be implemented in a high-level language. For example, the prototype server part is made in Python [10] and is cross-platform and easily manageable program. The client part can be implemented with HTML5 [11] in Java Script, which is also a standard solution. HTML5 specification includes the ”canvas” component, that allows scripts running on the client computer to perform rendering of arbitrary graphics in a clean and efficient way. Client-side Java Script programs are also capable of intercepting mouse events and creating timers, which is enough for our simple application.

The question text can be rendered separately as a normal text, because it should be as clear as possible. Since the script has its own timing source, it is capable of notifying users about the remaining time without overburdening the server. Every time the script detects a click it is sent to server as an answer, and then a new portion of animation data is received.

Since the rendered picture is transmitted in a vector format, it is very compact. In a very primitive format that we are using for initial implementation, the scene on Figure 3 would take just about 1 kB as a still image. For a sustained animation at 12 frames per second a

bit rate of $12 * 1 * 8 = 96$ kbit/s is required. This is available for virtually any device and on any server nowadays. A continuous uncompressed stream of 1 minute length would then take 720 kB of data, which is much more then typical CAPTCHA image, yet much less then any possible video-based challenge of the same length and resolution. If bandwidth is critical, the video resolution could be decreased at the expense of robustness against brute-force attacks.

Currently, we believe that the new concept of animated CR system would gain popularity and gradually replace classic CAPTCHA images which prove to be excessively inconvenient, especially for mobile users that are not capable of fast keyboard input. Even if our implementation of the challenge would prove to be weak, it may still be used as a basis for another applications that require animated 3D space to be presented to the users. We have also considered a possibility of someone actually making an algorithm that would crack the particular scene we have presented. Since it could not be a generic algorithm due to the nature of the objects we are using, it would be simply a matter of few minutes to create a new set of objects, and it would take a long time to create and debug a reliable algorithm to work on those new objects. Unlike the case with letters, here we are not restricted to just few symbols that all share specific proportions, but can actually use any distinct wireframe as a model.

IV. Question generation and semantics of the system

THe final topic that requires attention are the questions that would be asked. This is probably the most tricky part, because it is directly related to how hard it would be to make an algorithm to solve them. For example, a task to select the closest object would prove to be quite easy to break due to the nature of the projection used. At the same time, selecting a house that rotates counter-clockwise could present much greater challenge for a bot. The main reason is that second task requires identification of the object and motion tracking, while first one only requires locating the object presence.

To sort out the complexity issues, let us look at the list of system dynamics that are possible and could be used to create questions. We will consider possible questions and their answers, as well as requirements for the software that would solve the questions automatically.

1) Absolute position of the objects has 4 possible values - leftmost, rightmost, furthest away and closest. Also the semantic of height could be used to query for highest object or an object that is flying.
   Requires: separation, identification and transformation reconstruction
2) Relative position of objects, such as between or to the right of. Possible semantics may be object-specific, like a tree that grows on a roof of a house.
   Requires: separation of intersecting objects, identification, relation identification
3) Relative motion of objects, such as moving around of something. Possible semantics may be object-specific, like a house with rotating roof.
   Requires: separation of intersecting objects, identification, relation identification, motion tracking
4) Absolute motion of objects, such as moving towards the viewpoint or away from it.
   Requires: separation, identification, motion tracking, transformation reconstruction
5) Transformation of objects, such as a tree that is growing taller and shorter over time.
   Requires: separation, identification, motion tracking, transformation reconstruction

Probably many more semantics could be located, but the ones presented above should already provide a solid base for the question generator. The idea is to combine several relations, like ”a tree that grows on a wall of house that is rotating counter-clockwise”, create an object that matches the description at some points of time (rotation could be randomized, for example)

so the correct solution is not present in each frame and then integrate it into the scene. The other, irrelevant objects, should not fulfill the requirements of the question.

With such question generation techniques, the possible bot creators would have to spend lots of time working on each possible relation, and would essentially have to reconstruct the whole scene in real-time. While being very complicated as a signal processing task, this also would require massive amounts of computational power, so any useful algorithm should then have a very high recognition rates for it to be of any use.

V. Conclusion

As the progress marches on, it is getting harder to distinguish between humans and machines. Unfortunately, people with malicious intentions tend to exploit this phenomena to post spam on forums and abuse file sharing servers. In this paper authors have presented a new CAPTCHA concept, that is believed to replace the annoying distorted text challenges at some point. Unlike older systems that required mostly filtering capabilities of the human visual processing, the new one exploits also the capability to reconstruct relative positions of moving objects based on their projections. As this task does not yet have any common solution like character recognition, it is believed to provide much better protection.

Currently it is getting extremely hard to find a task that is easy for humans and impossible for machines. It gets even harder when a program should be capable of formulating the task and verifying the solution. Our proposal contains both the computer-generation of the task that is really tough to crack, as well as reliable algorithm for answer verification, that is not easy to crack with brute force. We will continue to improve the implementation and will also integrate the prototype into real websites for testing and feedback. The algorithm has been released as open-source public domain and is available at the project web page [12].

References

[1] K. Moore, “Rfc-3834 recommendations for automatic responses to electronic mail.”
[2] “Captcha official webpage.” http://www.captcha.net.
[3] J. Yan and A. S. E. Ahmad, “A low-cost attack on a microsoft captcha,” in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008.
[4] G. Mori and J. Malik, “Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA.” http://www.cs.sfu.ca/ mori/research/papers/mori_cvpr03.pdf, 2003.
[5] J. Baltazar, J. Costoya, and R. Flores, “The real face of koobface: The largest web 2.0 botnet explained.” http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/the_real_face_of_koobface_jul2009.pdf, 2009.
[6] C. Henry, “CAPTCHAs’ Effect on Conversion Rates.” http://www.seomoz.org/blog/captchas-affect-on-conversion-rates, July 2009.
[7] J. Yan and A. S. E. Ahmad, “Or usability issues in captcha design.”
[8] “Confident technologies graphical CAPTCHA.” http://demo.confidenttechnologies.com/captcha/.
[9] D. Danchev, “Inside indias captcha solving economy.” http://www.zdnet.com/blog/security/inside-indias-captcha-solving-economy/1835, August 2008.
[10] “Python official webpage.” http://python.org/.
[11] W3C, “HTML5 A vocabulary and associated APIs for HTML and XHTML.” http://dev.w3.org/html5/spec/Overview.html, March 2011.
[12] “Project webpage.” http://shimlar.tontut.fi/captcha3d/.