Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 1, January 2017, pg.135 – 147

Web Forms Security, Attack and Defense: Theory and Practice of CAPTCHA Technique

Ali F. Dakhil
Department of Computing Information Systems
Faculty of Computer Science and Information Technology, University of Summer, Iraq
allee.fattah@gmail.com

***Abstract:** As we know that internet is being utilized for several activities by a great range of users and these activities include communication, e-commerce, edification, education and regalement. Users are required to register regarding to a website in order to enroll web activities. However, registration can be done by automated hacking software. That software make erroneous enrollments which occupy the resources of the website by reducing the performance and efficiency of servers, even stop the entire web accommodation. It is critical for the websites to possess a system that has the inclination of varying human users and computer programs in perusing pictures of content. Consummately Automated Public Turing Test to Tell Computers and Human Apart (CAPTCHA) is a technique that protects websites against bots by generating and grading tests that humans can pass but current computer programs cannot. For example, humans can read distorted text, but current computer programs cannot. This study focuses on the CAPTCHA abilities of defensing a website from scammers by discussion its algorithm and practical steps to apply it in a web form.*

***Keywords:** CAPTCHA, Security, Website, Forms, Online, Spam, Bots, Hack, PHP, Application*

1. Introduction

CAPTCHA is a basic test to decide whether a user is a PC or a human. It is utilized to anticipate spam perversion on the websites. So on the off chance that we utilize CAPTCHA on our website, this can help in halting a few bots and making life harder for different bots in getting to or utilizing website structures. Along these lines, CAPTCHA insurance works by producing an arbitrary string, composing it to a picture, and afterward putting away the string within a session or by some other technique. This is then checked when the shape is submitted. Utilizing a contact form on the website is exceptionally valuable as it helps the website users to deal with it in a simple and basic way. Be that as it may, there are spammers and programmers who are searching for exploitable website. It is basic to secure web form against all "gaps" that those programmers are looking for. In this paper we will construct a PHP form that utilizations CAPTCHA code and also talk all the related issues respects to it, captcha.net (2016).

1.1 Previous Study

Luis von Ahn (2016) shows that CAPTCH was coined in 2000 by Luis Von Ahn, Manuel Blum, Nicholas Hopper and John Langford of Carnegie Mellon University. CAPTCHAs were originally developed by AltaVista to avoid the submission of URLs to the search engine, Moy et al (2004). It was a simple CAPTCHA which asks users to type a distorted English word. Also, Carnegie Mellon designed the Gimpy method which selects a word from dictionary and asks users to type what they see as an image after rendering the distorted image containing the text as in captcha.net (2016). The most common type of CAPTCHA was first invented in 1997 by two groups working in parallel: Mark D. Lillibridge, Martin Abadi, Krishna Bharat, and Andrei Z. Broder; and Reshef, Raanan and Solan (2016). Yahoo uses the simple version of this method; EZGimpy. EZ-Gimpy's image modification includes background grids, gradients, non-linear deformations, blurring, and pixel noise. Most humans can read three words from the distorted image, while current computer programs cannot, Chan (2003).

2. Guidelines for Applying CAPTCHA

In the event of website needs assurance from attack, it is suggested that it ought to utilize a CAPTCHA. There are numerous CAPTCHA accomplishments, some superior to others. The accompanying rules are emphatically prescribed for any CAPTCHA, Jose and Lakshmi (2014).

A. Accessibility

CAPTCHAs must be accessible. CAPTCHAs based solely on reading text or other visual-perception tasks prevent visually impaired users from accessing the protected resource. Any implementation of a CAPTCHA should allow blind users to get around the barrier, for example, by permitting users to opt for an audio or sound CAPTCHA.

B. Image Security

CAPTCHA images of text should be distorted randomly before being presented to the user. Many implementations of CAPTCHAs use undistorted text, or text with only minor distortions. These implementations are vulnerable to simple automated attacks.

C. Script Security

Building a secure CAPTCHA code is not easy. In addition to making the images unreadable by computers, the system should ensure that there are no easy ways around it at the script level. Common examples of insecurities in this respect including systems that pass the answer to the CAPTCHA in plain text as part of the web form and systems where a solution to the same CAPTCHA can be used multiple times (this makes the CAPTCHA vulnerable to so-called "replay attacks"). Most CAPTCHA scripts found freely on the Web are vulnerable to these types of attacks.

D. Security Even After Wide-Spread Adoption

There are various "CAPTCHAs" that would be insecure if a significant number of sites started using them. An example of such a puzzle is asking text-based questions, such as a mathematical question ("what is 1+1"). Since a parser could easily be written that would allow bots to bypass this test, such "CAPTCHAs" rely on the fact that few sites use them, and thus that a bot author has no incentive to program their bot to solve that challenge. True

CAPTCHAs should be secure even after a significant number of websites adopt them.

E. Should we make our Own CAPTCHA?
In general, making our own CAPTCHA script (e.g., using PHP, Perl or .Net) is a bad idea, as there are many failure modes. We recommend that we use a well-tested implementation such as reCAPTCHA.

3. The Pornography Attack Is NOT A Concern

It is sometimes rumored that spammers are using pornographic sites to solve CAPTCHAs: the CAPTCHA images are sent to a porn site, and the porn site users are asked to solve the CAPTCHA before being able to see a pornographic image. This is not a security concern for CAPTCHAs. While it might be the case that some spammers use porn sites to attack CAPTCHAs, the amount of damage this can inflict is tiny (so tiny that we haven't even noticed a dent!). Whereas it is trivial to write a bot that abuses an unprotected site millions of times a day, redirecting CAPTCHAs to be solved by humans viewing pornography would only allow spammers to abuse systems a few thousand times per day. The economics of this attack just don't add up: every time a porn site shows a CAPTCHA before a porn image, they risk losing a customer to another site that doesn't do this.

4. CAPTCHA Validation Algorithm

Let us examine CAPTCHA user validation algorithm, paying attention to protection against vulnerabilities which are not concerned with image recognition: user loads protected page, we create session for him. It will be best if the session created by CAPTCHA image script itself. The script generates random text, writes it to session and outputs image with this text to the user. While validating, user entered text is compared with the one from session. it is important if encoded text can't be calculated anyway from the data transferred to browser. for instance, idea to transfer the text (even encrypted) as an argument (in address line or in cookie), not storing on server in session, is bad. The text can be decoded or substituted by other one. if random text would be generated while generating page with input

form, not while generating image, bot will has an opportunity to make many requests to image script directly having many variants of same text (if distortion of the text varies). having pack of variants may simplify text recognition. Code generation by image inself also allow to create feature "generate another code, if this one unreadable" simply refresh the image. Common mistake is comparing entered text with one from session while validating user input. Hacker can give us identification of non-existent session and enter empty validation text. And empty text will be equal to one from non-existent session. So we must check text in the session if it is not empty. it is important to clean session after each validation (successful or not). Not considering on generation of new text while refreshing an image. Hacker can read image at first time, give answer to bot, so it could send this answer submitting form many times not loading a CAPTCHA image. If we would pay attention to this moments, our CAPTCHA will be protected against bots which uses no image recognition. See next topic about recognition-protected images generating. This algorithm is shown in fig. 1.

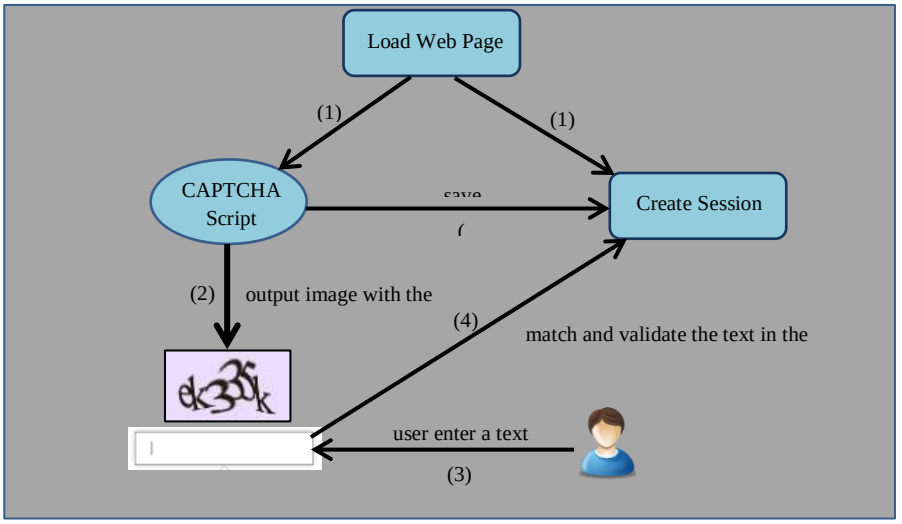

Fig. 1 Captcha Algorithm and validation steps

4.1 CAPTCHA Mechanism and Components

Mainly, CAPTCHA is a simple verification system made up of:

1. CAPTCHA image its code

   A computer generated image, sometimes distorted and in most cases the image is a .png or .jpg which usually consist of a background and letters and numbers of a different color that appear on that background.

2. A form field

   This field is into which the code a user sees in the CAPTCHA image must be inserted and an enter button. The field can either be part of a mail form, or a separate entity and is directly connected code wise to the CAPTCHA generated image.

3. A separate code

   This code is probably in PHP, which checks that the code inserted manually into the field was correct. Once the user enters the code that see in the image, the verification code checks to see if the code entered was correct, and if yes the form is sent off correctly or the user is redirected to the CAPTCHA protected pages, or in the case of an error, the form won't be sent, the CAPTCHA verification code generates an error.

5. Implementing CAPTCHA with PHP

In this topic we are going to apply such real CAPTCHA protected form. So, eventually we will create a form as in the fig. 2. First, we create the form with its basic elements. These elements are text fields, text area and button as well as CAPTCHA image.

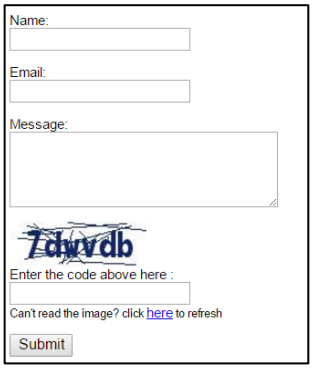

Fig. 2 Web form with CAPTCHA protection

This code in fig. 3 is to create the whole form as shown in fig. 2.

```
<form method="POST" name="contact_form"
action="<?php echo htmlentities($_SERVER['PHP_SELF']); ?>">

<label for="name">Name: </label>
<input type="text" name="name"
value="<?php echo htmlentities($name) ?>">

<label for="email">Email: </label>
<input type="text" name="email"
value="<?php echo htmlentities($visitor_email) ?>">

<label for="message">Message:</label>
<textarea name="message" rows=8 cols=30
><?php echo htmlentities($user_message) ?></textarea>

<img src="captcha_code_file.php?rand=<?php echo rand(); ?>"
id="captchaimg" >
<label for="message">Enter the code above here :</label>
<input id="6_letters_code" name="6_letters_code" type="text">

<input type="submit" value="Submit" name="submit">
</form>
```

Fig. 3 Web form elements code

In addition, customizing the CAPTCHA is allowed. The CAPTCHA script in the sample code can be customized. If we open the script, we can see the first few lines of the code as shown below in fig. 4.

```
$image_width = 120;
$image_height = 40;
$characters_on_image = 6;
$font = './monofont.ttf';

//The characters that can be used in the CAPTCHA code.
//avoid confusing characters (l 1 and i for example)
$possible_letters = '23456789bcdfghjkmnpqrstvwxyz';
$random_dots = 0;
$random_lines = 20;
$captcha_text_color="0x142864";
$captcha_noise_color = "0x142864";
```

Fig. 4 Generating CAPTCHA script code

Now, we validate the CAPTCHA, when the form is submitted, then the value in the session variable (6_letters_code) is compared with the submitted CAPTCHA code (the value in the text field 6_letters_code). If the codes match, then it will be preceded with emailing the form submission. Else it displays an error. Here is the code that does the server side processing fig. 5.


```php
if(isset($_POST['submit']))
{
  if(empty($_SESSION['6_letters_code'] ) ||
    strcasecmp($_SESSION['6_letters_code'], $_POST['6_letters_code']) !=
  {
      //Note: the captcha code is compared case insensitively.
      //if you want case sensitive match, update the check above to
      // strcmp()
    $errors .= "n The captcha code does not match!";
  }

  if(empty($errors))
  {
    //send the email
    $to = $your_email;
    $subject="New form submission";
    $from = $your_email;
    $ip = isset($_SERVER['REMOTE_ADDR']) ? $_SERVER['REMOTE_ADDR'] : '';

    $body = "A user  $name submitted the contact form:n".
    "Name: $namen".
    "Email: $visitor_email n".
    "Message: n ".
    "$user_messagen".
    "IP: $ipn";

    $headers = "From: $from rn";
    $headers .= "Reply-To: $visitor_email rn";

    mail($to, $subject, $body,$headers);

    header('Location: thank-you.html');
  }
}
```

Fig. 5 Code to embed the form with the CAPTCHA script image

Finally, we built our form with the CAPTCHA image as shown in fig. 2 by these codes above.

6. CAPTCHA Security

The shortest and most straightforward answer to this question is ‘yes’. Given enough time and effort, absolutely every single CAPTCHA implementation can be broken. This has been most prominently shown in the past by high-profile CAPTCHA-breaking incidents such as the Ticketmaster, Yahoo and Microsoft CAPTCHA cases (among others). Much of this can be explained by the fight between defenders and attackers, which is a normal phase of development in the formative years of any security solution. And even when the underlying technology matures and reaches an acceptable level of basic security, some elementary constraints remain. With CAPTCHA protection, as with all security solutions, risk can only be decreased, but there is no such thing as

a single security measure that is 100% safe. However, once these basic facts have been established, we can begin to realistically assess the effectiveness of CAPTCHA as a security measure. Just because somebody could theoretically spend a lot of resources to create a bot that bypasses a large enough percentage of any individual CAPTCHA implementation challenges, does not mean that they will do so. People, including Internet criminals, do things for a reason. The most common reason behind automated form submissions is spamming i.e. spreading a huge amount of nonsense and hoping enough people will fall for it to pay off to the spammer. Knowing the adversary is half the battle, and once we know what the attackers are after, we can prepare to defend against them. One of the points of using any kind of protection in the first place is to make would-be attackers have to expend effort and resources to bypass it. And think about the lock on a front door: it can also be broken, given enough time and effort (or a large enough power tool), but it still provides adequate protection. Locking a door does not 100% ensure that nobody will ever enter an apartment without permission but it helps a large deal, because it means they will have to try to pick or break the lock, and that takes time and risks attracting attention, Shahreza (2016).

Now how Protection against Recognition? We can ask how to draw recognition-protected images? At first, we must answer to question "how images being recognized usually". Then we can realize how we can make more difficult this recognition. Contrary to common opinion, for CAPTCHA defeating, as usual, they do not use scanned text recognition software as MS Office Document Imaging etc. So, if the CAPTCHA does not recognize by that software, it does not mean, that the text cannot be easily read by specially designed bot. Recognition usually divided into two main stages:

- Each symbols location finding.
- Recognition of each found symbol.

If symbols have constant positions see picture at fig. 6, only second stage remains. So we must vary symbol positions at least. If symbol positions are not constant, next way for symbols finding is comparing with background by contrast. If symbols color differs background one, it does not give any protection:

Fig. 6 Opaque CAPTCHA

Thus, we must add noise that cannot be easily separated from symbols, or we must make difficulties for symbols separation each from each, connecting or intersecting, Kaousar (2013). Symbol recognition in essence may be carried out by various ways. Simplest one is per pixel comparing. So, it compares each symbol with one from etalon font and selects symbol with maximal coincidence. CAPTCHA is vulnerable to per-pixel comparing if one does not use geometric distortion of symbols and uses one font (or very few fonts). Other, more sophisticated algorithms recognizes symbol by its peculiarities: branching, closed areas... There is class of algorithms named "neural networks". It is a "black box", trained to linking input shape with output answer. However, training procedure is usually lengthy and laborious. For protection against these algorithms one can add noise disfigures symbol shape, but we can receive image unreadable by human. It is believed that especial attention must be paid to first protection point: against symbol bounds determination it is difficult to recognize symbol if we do not know where it starts and ends as in fig. 7.

Fig. 7 CAPTCHA code shadow

This figure (fig. 8) below is to show these some companies with their CAPTCHA protection symbols showing the grades of difficulties.

Fig. 8 Symbols of CAPTCHAs

7. CAPTCHA types and comparison

ReCAPTCHA is a service that is freely available which ensures protection of our website from spam and abuse. ReCAPTCHA makes a use of an advanced risk analysis engine and adaptive CAPTCHAs for the purpose of obtaining automated software from engaging in abusive and unauthorized activities on our site, Athanasopoulos (2006). It does this by allowing a valid user to pass through it with ease. The ReCAPTCHA is an open captcha service that serves for a purpose for digitizing books and daily paper, to buckler our site from spam and bots. The ReCAPTCHA service gives two words that are randomly drawn from the database of server. The first word is control word and another one is a questionable word or vice-versa. The user interprets the words and after that the framework expects that if user sorts the control word accurately, then the questionable word are right after this the word is send to outcome the digitization of projects. In 2012, ReCAPTCHA started employing photos of house number taken from Google's Street View projects, Ahn et al (2014).

7.1 Text based CAPTCHAs

It is the simplest form of CAPTCHA. The simplest yet innovative method is to present the user with some queries, which only a human being can answer.

7.2 IMAGE based CAPTCHA

In this type CAPCHA the users has to identify image by performing image recognition task. First image CAPTCHA which used named as ESP Pix and it was developed at Carnegie Mellon University. A picture of ESP Pix CAPTCHA is shown in fig 9. In ESP Pix CAPTCHA user has choose one image and to pass the test user has to select related word from a list of 72 choices.

Fig. 9 Image CAPTCHA

8. Conclusion and Recommendations

The idea driving this field begun from the issues confronted by enormous Internet organizations, for example, Yahoo and AltaVista. Arrangement was created by requesting that human users fathom a CAPTCHA test before they include online exercises. The resistance systems and assaults are still powerfully refreshed. In this paper, we exhibited a CAPTCHA strategy which works for recognizing a human user from destructive PC programs.

Finally, we recommend web forms to use CAPTCHA. Also, this CAPTCHA must be as hard as possible against any kind of attack. To conduct, using CAPTCHA needs to be implemented under its guidelines and requirements.

References

[1] The Official CAPTCHA Site. [Online]. Available: http://captcha.net/ 2016

[2] Luis von Ahn. [Online]. Available: https://en.wikipedia.org/wiki/Luis_von_Ahn 2016

[3] Moy, G., Jones, N., Harkless, C. and Potter, R., "Distortion estimation technique in solving visual CAPTCHAs", Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol.2, pp.23-28, 2004.

[4] The CAPTCHA Project. People And Colaborators. [Online] Available: http://www.captcha.net/people 2016.

[5] Rajaa K .Hasoun, Soukaena H. Hashem and Rehab F. Hasan "Propose Image Captcha System" vol 26 p. 86, Dec. 2016

[6] Chan, T.Y., "Using a Text-to-Speech Synthesizer to Generate a Reverse Turing Test", Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, pp. 226 – 232, 2003.

[7] Jose A. and Lakshmi V., "Web Security Using Visual Cryptography Against Phishing", Middle-East Journal Of Scientific Research, Vol. 20, 2014

[8] Shahreza, M., Shahreza, S., "Preventing Mobile Software Cracking Software", IEEE, Innovations in Information Technology, Dubai, pp. 1-5, 2006.

[9] Kaousar K. M. A., "Trio Framework For Secure Online Transaction Using Visual Cryptography", International Journal Of scientific and Research Publication, Vol. 3 May 2013.

[10] E. Athanasopoulos and S. Antonatos. Enhanced captchas: Using animation to tell humans and computers apart. In IFIP International Federation for Information Processing, 2006

[11] L. von Ahn, M. Blum, J. Langford, "Telling Humans and Computers Apart Automatically Communications of the ACM", vol. 47, no. 2, pp. 57-60, Feb. 2014.

