Journal of Advanced Computer Science & Technology, 4 (1) (2015) 15-22
www.sciencepubco.com/index.php/JACST
©Science Publishing Corporation
doi: 10.14419/jacst.v4i1.3935
Research Paper

Strategies and scenarios of CSRF attacks against the CAPTCHA forms

Hossein Moradi [1]*, Hossein KardanMoghaddam [2]

[1] *Faculty Member of Birjand University of Technology, Birjand*
[2] *Faculty Member of Birjand University of Technology, Birjand, Iran*
**Corresponding author E-mail: h.kardanmoghaddam@birjandut.ac.ir*

Abstract

In this article, we've tried to examine the hypothesis of the robustness of a form by using CAPTCHA against CSRF and login CSRF attacks. Our investigations showed that unlike public opinion, common attacks to bypass CAPTCHAs such as Optical Character Recognition (OCR) and 3rd party human attacks are not applicable in the CSRF case and instead, Clickjacking is the most important scenario of CSRF and login CSRF attacks against a secure session-dependent CAPTCHA form. Remember that the Clickjacking is also applicable to bypass the well-known CSRF protections, such as the secret token and the Referer header. Therefore, although the frequent application of CAPTCHA on every page of a website negatively impacts the user experience, but the robustness of a robust session-dependent CAPTCHA against the CSRF and login CSRF attacks is almost the same as the session-dependent security token. Moreover, when a website is using a session-independent or week pattern of CAPTCHA, attackers can bypass the CAPTCHAs and launch the CSRF or login CSRF attacks by using XSS, session hijacking, replay attacks or submitting a random response.

Keywords: *CAPTCHA; CSRF; HTTPS; CSRF Attacks.*

1. Introduction

In 2013, OWASP security community reported that CSRF is 8th among top 10 security risks of the web applications which can cause serious problems for the victims through threatening users` web browsers [1]. In this kind of attack, "a malicious site instructs a victim's browser to send a request to an honest site, as if the request was part of the victim's interaction with the honest site, leveraging the victim's network connectivity and the browser's state, such as cookies, to disrupt the integrity of the victim's session with the honest site" [2].
Another type of CSRF attacks is login CSRF "which is currently widely possible, damaging, and under-appreciated. In login CSRF, an attacker uses the victim's browser to forge a cross-site request to the honest site's login URL, supplying the attacker's user name and password. A vulnerable site will interpret this request and log the victim into the site as the attacker. The impact of login CSRF attacks varies by site, ranging from allowing the attacker to mount XSS attacks on Google to allow the attacker to obtain sensitive financial information from PayPal" [2].
There are different ways to confront these attacks. Barth et al. claimed that the best defensive methods against CSRF and login CSRF are the secret token, the Referer header, custom HTTP header (XMLHttpRequest) and Origin header. The secret token method includes a secret token with each request of the user, and can be implemented based on the user`s session identifier, session-independent nonce, session-dependent nonce, and HMAC of user's session identifier. The first two methods (i.e. using the user`s session identifier as a token and session-independent nonce) are not so applicable against CSRF and login CSRF attacks, but the two other methods (i.e. session-dependent nonce and HMAC of user's session identifier) can provide high safety against CSRF and login CSRF attacks, however, the practical implementation of these two methods is difficult and requires special skills. NoForge is an example of the unsuccessful implementation of the session-dependent nonce [2].

Another popular security method is the validation of Referer header in which, the site can defend itself against CSRF by checking whether the request in question was issued by the site itself.
Unfortunately, the Referer contains sensitive information that violates the web users' privacy. For example, the contents of the search query that lead the user to visit a particular site may be revealed by the Referer header, which leading to block the HTTP Referer headers by some network proxies. Therefore, this method is advised for the HTTPS, but not for the HTTP [2].
The third way to protect against CSRF attacks is the custom HTTP headers such as XMLHttpRequest. These headers can prevent CSRF attacks, because browsers do not let websites to send custom HTTP headers to other websites, but a website can send these headers to itself [2], although the prerequisite for this approach is to redesign all the web pages, which makes it difficult to use it in conjunction with the existing websites.
As an alternative to three mentioned solutions, Barth et al. proposed a long term solution, called the Origin header in which, web browsers settings are changed in a way that browsers send the Origin header instead of the Referer header. In this method, Origin header only sends scheme, host and the port associated with the URL of the active document, and unlike the Referer header, it does not send the user query or the URL path of the referrer website [2]. Unfortunately, implementing this method requires to change the web browser of all users, which is difficult and limits its application [4].
Although most popular websites are using the secret token or Referer header as the major protections against CSRF and login CSRF, but in the case of the combinational attacks, these protections will also become weak and useless. For example, CSRFGuard framework which integrating the ideas of the secret token and the validation of Referer header can be a good solution to protect against the CSRF and login CSRF attacks, but Chen et al. showed that we can use of the vulnerabilities, including the XSS, Clickjacking or session hijacking to bypass the CSRFGuard and create a CSRF attack. However, the CSRFGuard can eliminate the risk of session hijacking by taking a defense in depth strategy (by checking the Referer header) to mitigate the risk of an unintentional leverage token leverage by unauthorized users [5].
According to the above discussion, it seems necessary to study the other defensive strategies and analyzing the vulnerability of them against CSRF and login CSRF attacks.
Cheng & Paulina stated that in addition to using unpredictable validation token, we can use the challenge-response methods such as re-authentication (password) or CAPTCHA [6]. Xing et al. also explained that three defensive methods against CSRF attacks are the validation of Referer header, secret validation token and CAPTCHA [7], though frequent application of CAPTCHA or re-authentication in every page of the website would negatively impact the user experience [6], [7]. Moreover, Blatz (from MacAfee Company) also claims that using CAPTCHA would secure websites against CSRF attacks. Since the attackers cannot see the CAPTCHA picture, they would have no chance to identify and send the proper text. Additionally, the degree of distortion has no impact on CSRF protection and even the worst possible CAPTCHA provides effective defense against CSRF [3].
In addition, the OWASP security community also considers challenge-response, including the CAPTCHA, re-authentication and one-time token (assuming the proper implementation) as a very strong defense to protect against CSRF attacks, though CAPTCHAs negatively impact user’s experience. For high security applications, secret token and challenge-response should be used in high risk functions [8].
However, unlike above viewpoints, some experts do not consider CAPTCHA as a suitable defensive strategy. For example, Barth et al. claimed that “Although CAPTCHAs have many other applications; they offer few advantages over secret validation tokens as a CSRF defense. If it is known to the attacker which CAPTCHA is displayed, then the attacker can manually solve CAPTCHAs and attack one user per CAPTCHA solved, which is expensive but probably still cost-effective. If the decision of which CAPTCHA to display is a session-dependent secret, then this information could be used as a session-dependent secret validation token without burdening the user with the task of solving a CAPTCHA” [2]. Homakov also claims that CAPTCHA is not a defense against CSRF attack. He stated that CAPTCHA consists of challenge and response. The challenge is the identifier of the picture and challenge must not disclose anyhow what the response is? Afterwards, he has designed an internet tool to test the Google CAPTCHA and claims that the CAPTCHA doesn't consider the Origin of the request. It is only a mitigation from automated requests and spam (although not perfect too). The challenge can be stored in the cookie, but it is still an incomplete approach [9].
Since CAPTCHA has many applications such as preventing comment spam in blogs, protecting website registration, protecting email addresses from scrapers, online polls, preventing dictionary attacks, preventing search engine bots, worms and spam [10], most websites are using CAPTCHAs in their critical web pages. According to the importance of these kinds of web pages, they may be the target of CSRF attacks. Although the previous studies [6], [7], [8] believe that using CAPTCHA in the whole web pages would be annoying for the users, but the different researchers have different views about the strength of the forms which using CAPTCHA against CSRF and login CSRF attacks. Therefore, it is important to study the robustness of the CAPTCHAs against CSRF attacks, because if their robustness is approved, redesigning the existing CAPTCHA-based web pages and providing additional protections such as secret validation token would not matter anymore. Therefore, the main hypothesis of this article is: “A web page which using CAPTCHA is secure against CSRF and login CSRF attacks”.
The following questions are answered to confirm or reject this hypothesis. The main question of this article is: “Whether a form which using CAPTCHA is secure against CSRF and login CSRF attacks?”
The alternative question of this article is:”If a form that uses CAPTCHA is vulnerable against CSRF or login CSRF attacks, what are the scenarios and prerequisites of these attacks?”

To answer these questions, this article has been organized in following sections. First of all, the methodology is presented, then the literature review follows, and after that, CAPTCHA attacking scenarios for a CSRF attack are described. In the following, the most important attack scenario will be validated empirically and finally, our findings and conclusions are explained.

2. Methodology

In this article, the vulnerability of the forms which using CAPTCHA will be discussed through a broad literature review and executing two practical experiments and the research hypothesis would be confirmed or rejected according to each attack scenario. Therefore, the total process of this article includes:

- Review of the literature
- Identification and extraction of attacking scenarios to bypass CAPTCHAs and launching CSRF attacks
- Deep study of the literature related to the identified scenarios
- Practical validation of the major attack scenarios
- Interpreting and qualitatively analyzing of the previous findings
- Providing the results and conclusions

3. Literature review

In the first part of this section, we review the general approaches to bypass CAPTCHAs.

3.1. General approaches to bypass CAPTCHAs

Truong et al. stated that there are two effective ways to break the different kinds of CAPTCHAs. The usual way to solve the challenge of the CAPTCHA is the Optical Character Recognition (OCR) technique. Jitendra Malik & Greg Mori broke the E-Z Gimpy CAPTCHA using this technique with the success rate of 92 percent. Yan & El Ahmad could also break MSN CAPTCHA template with 61 percent of success [11].

The other method to break the CAPTCHAs is 3rd party human attacks. A usual CAPTCHA should be solved easily by a human, but computer software should not be able to read it. Therefore, using human is a guaranteed method for CAPTCHA breaking. ”In fact, some unethical businesses are profiting by providing 3rd party human CAPTCHA solving as a service. In India, a CAPTCHA solving economy has developed in which people work as human CAPTCHA solvers, earning $2 for every thousand CAPTCHAs solved” [11].

Although, there are multiple anti-CAPTCHA OCR software solutions, which can be used to break CAPTCHAs and sending spam to forums, but there are no methods to use these software solutions in CSRF attacks. In the case of a form which using a secure CAPTCHA, the attacker has to read and solve or overcome it using brute force technique. It is not possible to read the challenge because of the Same Origin Policy (SOP), so OCR or 3rd party human attacks are not applicable for CSRF attacks [12].

However, if the CAPTCHA is originated from an identifier which reveals its value, the attacker can restore a CAPTCHA, solve it manually, and use it for the further CSRF attacks, as long as the CAPTCHA is not associated with the user session [13]. It is obvious that breaking the CAPTCHA according to the mentioned scenario is dependent on implemening details of the CAPTCHA. For example, Alex says that two types of CAPTCHA systems usually are being used. The first type is presented in the following, in which the web page encompasses a hidden input tag, including a value that refers to the URL of the image [14].

```
<input type="hidden" name="captcha_id" value="1234567890" />
<img src="captcha.php?id=1234567890" />
```

Since the identifier is not dependent on the user session, and the CAPTCHA does not store the sent identifier, it is easily broken, because the attacker can identify the CAPTCHA type, then solve it and finally, use the associated response to forge a request for the further CSRF attacks. This scenario is a kind of replay attacks.

However, if we use the second type of CAPTCHA implementation in which, the value is stored in the server session and the CAPTCHA is requested and displayed in a form tag such as <img src="captcha.php"/>, it would not be possible to break the CAPTCHA according to the replay attacks [14].

RAMISETTY also claims that CAPTCHA solution has been discarded some time ago due to the possibility to retrieve the CAPTCHA image using the so called MHTML bug, which affected several versions of Microsoft Internet Explorer (including version 5.5, 6 and 6.x) [15]. However, Wikimedia reported that in 2013 year, less than 2 percent of the internet users have used 5.5, 6 and 6.x versions of Internet Explorer [16], and it is obvious that many of them have installed related patches. But even if the CAPTCHA picture successfully restores, there would be a serious doubt for attacker’s access to the restored picture in the victim's browser. It should be stated that there are no sources, mentioning the ways of using this vulnerability in CSRF attacks.

3.2. Current approaches to bypass secret tokens

According to the similarities of CAPTCHAs and secret tokens, in this part, we review the current approaches to bypass the secret token for CSRF attacks.
Xing et al. stated that when a website has the XSS vulnerability, implementing Referer header and secret validation token are completely useless [7]. Amit et al. also described that Clickjacking (i.e. embedding a web page in an iframe) is one of the important methods, which is used by attackers to overcome the random nonce between client and server [17]. Amrutkar et al. also provided the practical implementation of the Clickjacking for login CSRF attacks [18].
OWASP also claims that it is possible to use the XSS vulnerability to break a token, double-submit cookie and other CSRF guards like Referer header and Origin header. Since XSS scripts can read every page by using XMLHttpRequest, these scripts can extract the token from the related response and attach the extracted token to a forged request [19].
According to [5] and [19], to launch CSRF and login CSRF attacks, there is a possibility to bypass security token and Referer header mechanisms via one of the XSS vulnerabilities, intercepting the traffic to hijack the user session, and Clickjacking. Therefore, in the following part, we will discuss the studies related to using those vulnerabilities to bypass CAPTCHA-based forms.

3.3. Combinational scenarios to bypass CAPTCHAs

CAPTCHA is not vulnerable against CSRF, unless we can bypass it according to exhaustive attacks or other combinational attacks such as Clickjacking [12]. If the CAPTCHA challenge binds to the user session, it performs just like cookies and would be vulnerable against iframe overlays, but not against automated form sending [13]. Alex also says if the CAPTCHA value is saved in a server-side session and called and displayed in the related form via the <img src="captcha.php" /> tag, the best method for CSRF attacks would be embedding the CAPTCHA form in an iframe and requesting the user to solve it [14].
Moreover, HP report claims that in order to build a secure CAPTCHA, we have to make sure that computers cannot read the picture and the CAPTCHA cannot be bypassed using script. Moreover, if one CAPTCHA identifier is used repeatedly in different pages, the application will become vulnerable against the replay attacks [20].
OWASP Community also states that XSS cannot be used to break challenge-response defenses such as CAPTCHA, re-authentication or nonce [19], but Adrian claims that in the case of XSS vulnerabilities, his brilliant scenario can be used to bypass re-authentication and launching CSRF attacks [21].
He also says that if there is the XSS vulnerability, it is possible to bypass CAPTCHA security mechanism with special techniques [21], though he has not provided any evidence or details for his second claim.

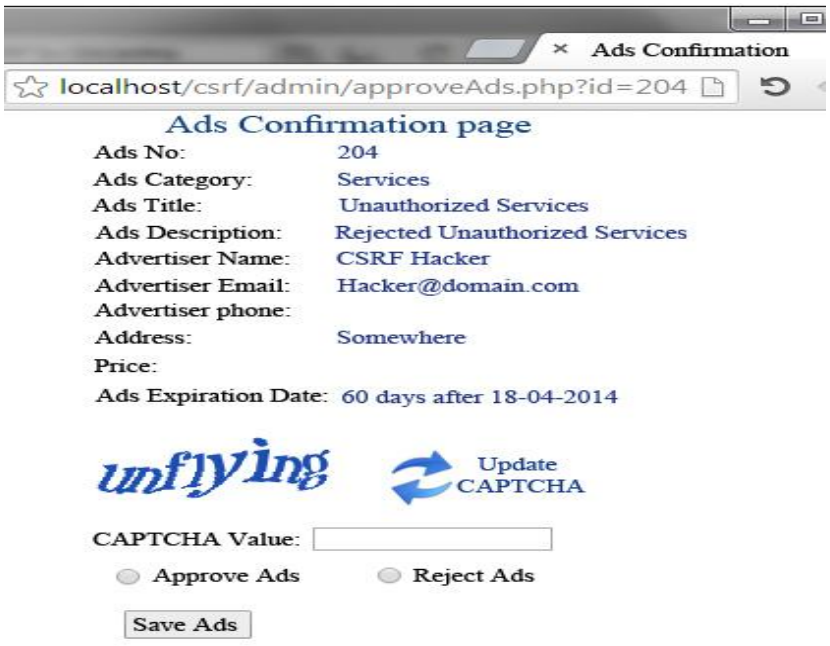
4. Scenarios to bypass CAPTCHA and launching CSRF attacks

After reviewing the related literature and identifying the different scenarios to bypass CAPTCHA and launching CSRF attacks, we extracted and listed these scenarios and their associated references (Table 1). As seen in two left columns of this table, nine scenarios were identified and classified into two categories of specific attacks to bypass CAPTCHAs (5 highlighted scenarios) and general attacks to bypass CAPTCHAs (next 4 cases).

4.1. Practical validation of the major scenarios of attack

One of the major scenarios of attack, which recommended by different sources is Clickjacking to solve the CAPTCHA by the victims. Therefore, in this phase, we designed an advertising website for simulating and evaluating the Clickjacking to solve the CAPTCHA by the victim user and launching CSRF and login CSRF attacks.
The advertisement website is using a CAPTCHA pattern to prevent login CSRF, but through framing the website and hiding some parts of the advertisement validation page, the victim user will be deceived to enter the attacker`s username and password and solve the CAPTCHA. It should be mentioned that because of the similarity of this aspect of the empirical validation with the Armutkar et al one [18], we didn't provide the related pictures here [18].
Moreover, as you can see in Figure 1, the advertisement website is using a CAPTCHA to prevent CSRF during the confirmation of advertisements by the website manager.
But as you can see in Figure 2, the attacker frames the illegal ads in an iframe and hides some parts of it by a few pictures, then deceiving the manager to solve the CAPTCHA and pressing the fake download button to download an interesting software or movie (for free).
In Figure 2, the Save ads button is overlaid by a fake download button, and whenever a user clicks on the download button, the Save ads button will be pushed actually. However, to overlay the Save ads button with a fake one (such as the Download button), we had to reduce the opacity of the original iframe, which may lead to the user awareness of the Clickjacking attack, since the textbox opacity will be unusual during the solving of the CAPTCHA by the victim. However, if the designer of the Confirmed Ads webpage had used a common value such as the Save or Send for saving

the ads (instead of using the Save ads value), the attacker wouldn't have compelled to overlay the mentioned button with a new one, and consequently, the possibility of the CSRF attacks would be increased.



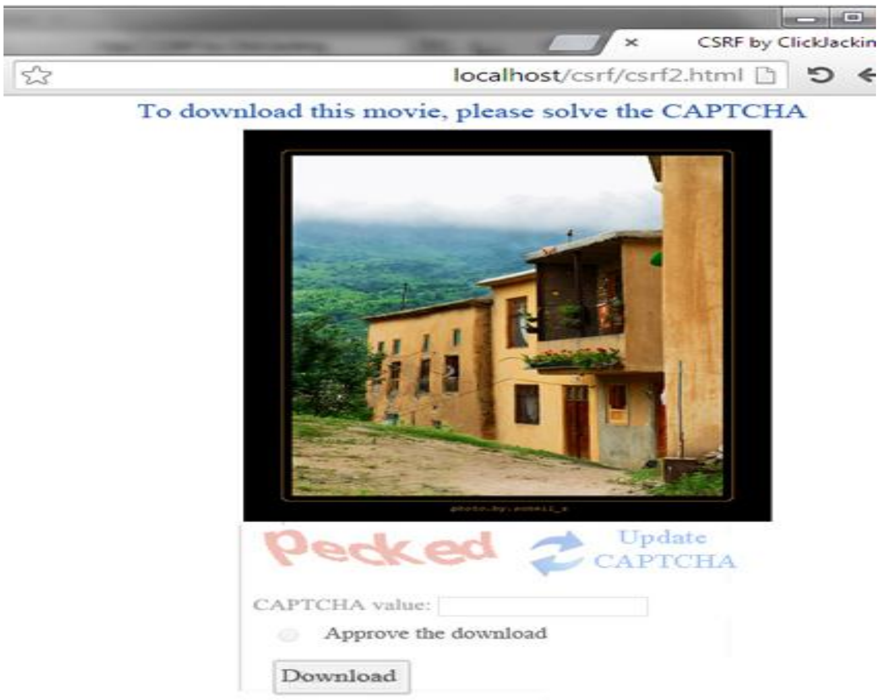
Ads Confirmation

localhost/csrf/admin/approveAds.php?id=204

Ads Confirmation page

Ads No:	204
Ads Category:	Services
Ads Title:	Unauthorized Services
Ads Description:	Rejected Unauthorized Services
Advertiser Name:	CSRF Hacker
Advertiser Email:	Hacker@domain.com
Advertiser phone:	
Address:	Somewhere
Price:	
Ads Expiration Date:	60 days after 18-04-2014

Fig. 1: The Original Webpage of "Ads Confirmation", Presenting for the Website Manager



CSRF by ClickJackin

localhost/csrf/csrf2.html

To download this movie, please solve the CAPTCHA

Fig. 2: Clickjacking Attack to Solve the CAPTCHA by the Victim User and Launching CSRF

5. Results

After the broad reviewing and analyzing the literature associated with 9 scenarios and running two empirical experiments on Clickjacking scenario, we identified the prerequisites and the success probability of these scenarios, and provided them in the two right columns of Table 1.

As you can see in this table, there are two kinds of categories to bypass the CAPTCHAs and launching a CSRF attack. The first category, which is highlighted, is related to the specific solutions for bypassing CAPTCHAs. Since the attacker does not have access to the CAPTCHA picture which displayed for the victim [3], it is not feasible to use the OCR or 3rd party human attacks to launch CSRF and login CSRF attacks.

Table 1: Different scenarios to bypass CAPTCHA and launch CSRF and login CSRF attack

Attack scenario	Refs	Attack prerequisites	Success probability
1. OCR attacks to automatic discovery of the response	[3] [11] [12]	* Attacker`s access to the CAPTCHA picture which displaying for the victim	Not applicable
2. 3rd party human attacks to solve the CAPTCHA by humans	[11] [12]	* Attacker`s access to the CAPTCHA picture which displaying for the victim	Not applicable
3. CAPTCHA picture restoring via the MHTML bug	[15]	* Using users of IE browser (versions 5.5, 6 or 6. X) * Lack of the installation of the published patches *Attacker`s access to the restored picture of the CAPTCHA	Not applicable
4. Solving a random instance of CAPTCHA and using its associated identifier and response for the further CSRF attacks	[9] [13] [14] [21]	*Independence of the CAPTCHA from user session * Repetition of the CAPTCHA challenge *Existence of an identifier to identify the associated response of a CAPTCHA instance	Not applicable for the session-dependent CAPTCHAs & High probability for the session-independent CAPTCHAs
5.Submitting random responses to solve the CAPTCHA		*Do not interrupt the attacks by the server`s intrusion detection system (IDS) * Do not display an error in the browser of the victim *Limited variety of the symbols in the CAPTCHA pattern *Limited number of the symbols in the CAPTCHA pattern	Very low probability for secure CAPTCHA Patterns & Medium probability for insecure CAPTCHA patterns
6. Exhaustive attack to CAPTCHA to test all the possible responses	[12]	* Do not renew the CAPTCHA after each false response *Do not interrupt the attacks by the server`s intrusion detection system (IDS) *Do not display an error in the browser of the victim	Not applicable
7.XSS attacks to hijack the user session after filling the CAPTCHA by the victim	[5] [7] [12] [19] [20] [21]	*Existence of the XSS vulnerability in one of the web pages of that domain *Lack of the defensive mechanisms against XSS *Ability to launch the CSRF attack through XSS	Not applicable for websites without XSS vulnerability & Medium probability for websites which have XSS vulnerability
8. Intercepting the CAPTCHA challenge and related response to launch a replay attack or forge the response	[5] [12] [21]	* Independence of the CAPTCHA from user session * repetition of the CAPTCHA challenge *Feasibility of the long term interception of communications between client and server	Not applicable for the session-dependent CAPTCHAs & Medium probability for the session-independent CAPTCHAs
9. Clickjacking attack for filling the CAPTCHA by the victim	[5] [12] [14] [16] [17]	*No defensive strategies against Clickjacking in the critical web pages	Not applicable for websites which have defensive strategies against Clickjacking & High probability for websites which do not have defensive strategies against Clickjacking

Therefore, the most important attack of this category would be to solve a random instance of CAPTCHA and using its associated identifier and response to launch the further CSRF attacks. Since the main prerequisites of this scenario are the independence of CAPTCHA from the user session, repeating the CAPTCHA challenge and the existence of a CAPTCHA identifier, this technique would have little success on the robust session-dependent CAPTCHAs. However, in the case of session-independent CAPTCHAs such as the following one, attacker would have high chance to launch CSRF:

```
<input type="hidden" name="captcha_id" value="1234567890" />
<img src="captcha.php?id=1234567890" />
```

Moreover, submitting random responses for solving the CAPTCHA and launching CSRF has low chance in the case of the robust CAPTCHA patterns, but it has medium chance for insecure CAPTCHA patterns that the possible responses are limited.
The second category of the attack scenarios is the general attacks to bypass the different kinds of CSRF protections. These attacks are also applicable to bypass the common CSRF protections, such as the secret tokens and Referer header validation.
Since the CAPTCHA is renewed after every false response, it is not possible to use the exhaustive attack to examine all the possible responses. Moreover, in the case of session-independent CAPTCHAs such as the above example, attackers can use of the general techniques such as the interception of the CAPTCHA challenge and related response to launch a replay attack or forge the response or leverage the XSS vulnerability to hijack the user session after filling the CAPTCHA by the victim. However, in the case of XSS vulnerability on the website, a robust session-dependent CAPTCHA would have higher security than some kinds of secret tokens for protecting against CSRF.
Consequently, in the case of a secure session-dependent CAPTCHA, the most important attack scenario is Clickjacking for filling the CAPTCHA by the victim user. Remember that Clickjacking has been considered between the 10 additional attacks on the web applications by the OWASP community [1]. Moreover, to decrease the chance of the attackers to launch CSRF and login CSRF attacks based on Clickjacking, it is advised that web designers change the default value of the submit buttons to a more specific one. In addition, fortunately Clickjacking can be prevented by implementing X-Frame-Options within HTTP response headers [23].

6. Conclusion

To examine the hypothesis of "a web page which using CAPTCHA is secure against CSRF and login CSRF attacks", we've tried to bypass the CAPTCHA and launch the CSRF and login CSRF according to several attack scenarios. Our investigations showed that most practical implementations of the CAPTCHAs (such as Google CAPTCHA) are vulnerable against CSRF and login CSRF attacks.
Moreover, unlike the public opinion, common attacks to bypass CAPTCHAs (such as OCR and 3rd party human attacks) are not applicable in the CSRF case and instead, the most important scenarios of CSRF and login CSRF attacks against a secure session-dependent CAPTCHA form are Clickjacking. Remember that the Clickjacking is also applicable to bypass the well-known CSRF protections, such as the secret token and the Referer header.
Therefore, although the frequent application of CAPTCHA on every page of a website negatively impacts the user's experience, the strength of a robust session-dependent CAPTCHA against the CSRF and login CSRF attacks is almost the same as the session-dependent security token. However, in the case of XSS vulnerability on a website, a robust session-dependent CAPTCHA would have higher security than some kinds of secret tokens for protecting against CSRF.
Moreover, when a website uses a session-independent or week pattern of CAPTCHA, there are four other scenarios to bypass the CAPTCHA and launch a CSRF attack, including XSS, session hijacking, replay attacks or submitting a random response.

References

[1] OWASP, "OWASP Top 10-2013: The ten most critical web application security risks," OWASP open community, 2013.
[2] Barth, A., et al., "Robust defenses for cross-site request forgery," in Proceedings of the 15th ACM conference on Computer and communications security, Chicago, pp. 75–88, 2008. http://dx.doi.org/10.1145/1455770.1455782.
[3] Blatz, J., "CSRF: Attack and Defense White Paper," McAfee Inc, 2011.
[4] Li, X., Yuan, X., "A Survey on Server-side Approaches to Securing Web Applications," ACM Computing Surveys, Vol. 46, No. 4, 2014. http://dx.doi.org/10.1145/2541315.
[5] Chen, B., et al., "A Study of the Effectiveness of CSRFGuard," presented at the IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing, 2011.
[6] Cheng, C., Paulina, M., "Protecting Web-based Applications," NATIONAL UNIVERSITY OF SINGAPORE, SCHOOL OF COMPUTING, 2013.
[7] Xing, L., et al., "A client-based and server-enhanced defense mechanism for cross-site request forgery," Recent Advances in Intrusion Detection, Springer Berlin Heidelberg, pp. 484–485, 2010.
[8] OWASP, "Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet." OWASP open community, December 2013, https://www.owasp.org/index.php/ Cross-Site_Request_Forgery(CSRF) Prevention_Cheat_Sheet.
[9] Homakov, "Why captcha is not a CSRF protection." GitHub Website, November 2014, https://gist.github.com/homakov/5607607.
[10] Captcha.net, "Applications of CAPTCHAs" captcha.net website, January 2014, http://www.captcha.net.
[11] Truong, H. D., C. F. Turner, and C. C. Zou, "iCAPTCHA: the next generation of CAPTCHA designed to defend against 3rd party human attacks," presented at the IEEE International Conference on Communications (ICC), pp. 1–6, 2011.
[12] Stackexchange, "can a csrf captcha be defeated." Stackexchange website, November 2014, http://security.stackexchange.com/questions/11768/can-a-csrf-captcha-be-defeated.
[13] Commander, A., "Captcha as CSRF protection." Slacker's website, November 2014, http://sla.ckers.org/forum/read.php?4,24090.
[14] Alex, "Exploiting CSRF Protected XSS." Alex's Corner Weblog, April 2014, http://kuza55.blogspot.com/2008/02/exploiting-csrf-protected-xss.html.
[15] Ramisetty, R., "Heuristics For Preventing Cross Site Request Forgery Attacks," Msc Thesis, National Institute Of Technology Karnataka, 2009.

[16] Wikimedia, "Wikimedia Traffic Analysis Report – Browsers e.a." Wikimedia Website, November 2014, http://stats.wikimedia.org/archive/squid_reports/2013-06/SquidReportClients.htm.
[17] Amit, Y., et al., "Patent: Thwarting cross-site request forgery (csrf) and Clickjacking attacks," U.S. Patent US20110321168 A128-Jun-2010, November 2014, http://www.google.com/patents/US20110321168.
[18] Amrutkar, C., et al., "On the Disparity of Display Security in Mobile and Traditional Web Browsers Technical Report," Library of Georgia Institute of Technology, April 2014, https://smartech.gatech.edu/bitstream/handle/1853/36978/GT-CS-11-02.pdf?sequence=3.
[19] OWASP, "Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet." OWASP open community, April 2014, https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet.
[20] HP, "Cross-site request forgery: are your web applications vulnerable? Whitepaper," HP Company, 2007.
[21] Adrián, "XSS killed the anti-CSRF star." Security et alii website, April 2014, http://securityetalii.es/2013/01/23/ xss-killed-the-anti-csrf-star
[22] OWASP, "Clickjacking Defense Cheat Sheet," OWASP open community, July 2014, https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet#Defending_with_X-Frame-Options_Response_Headers.