Cross-Site Request Forgery Attack and Defence: Literature Search

Chaohai Ding
School of Electronics and Computer Science,
University of Southampton
Email: cd8e10@ecs.soton.ac.uk

***Abstract*—This literature search presents a summary of the Cross-Site Request Forgery (CSRF) attacks and some existing defences related to this intrusion. Section One mainly sketches an introduction of CSRF attacks while Section Two enumerates some current intrusion methods of CSRF attacks and distinctive threat models. Some countermeasures related to these attacks are briefly described in Section Three. Then Section Four presents the further information about this area and recommends some effective defences to detect and prevent CSRF attacks.**

A detailed description about preventing CSRF is not included in this literature research and more details related to defences will be presented in the technical report.

I. Introduction

Cross-Site Request Forgery (CSRF or XSRF, also known as Session Riding or Confuse Deputy) is an attack that tricks the victim to load a web page that contains malicious codes and forces the victim to send a request to the honest web site. If the victim has not logged out the website or the session is still active, the attacker could forge the victim as an authenticated user to perform specific operations that the victim would not want[1] [2]. Alexenko et al. pointed out that this attack also occurs in social networking applications or emails and it is not restricted to vulnerable websites [3].

Compared with other intrusion means such as Cross-Site Scripting (CSS or XSS) or SQL injection, few effective defences are available for CSRF attacks. Zeller et al. [4] presented that CSRF attacks are more vulnerable than CSS because of the lace concern on CSRF and the administrators are uneducated about risks of CSRF, Therefore the CSRF attack is listed among the top ten web application security risks in 2010 by the Open Web Application Security Project (OWASP) [5].

II. CSRF Attacks Research

In recent research, both Lawton [6] and Mao [7] provided some background information of what CSRF attacks are while Johns et al. explained the reason why there is the existence of CSRF vulnerabilities in [2]. Moreover, a particular scenario is presented in [4], which describes how an attacker exploits the authentication mechanisms of the target website. Both Gollmann [8] and Lin et al. [9] divided CSRF attacks into stored and reflected, which will be described as follows.

A. Stored CSRF Attacks

In [9] Line et al. presented a principle of stored CSRF attacks and implemented a threat modelling about this intrusion. Stored CSRF attack means that the target website is within the same domain with the attacker. Burns [10] also provides a brief introduction about the stored CSRF. But this intrusion always succeeds because the session of the victim is still active.

B. Reflected CSRF Attacks

According to [9] [10], more than half percentage of CSRF attacks are reflected CSRF attacks due to a wide range of attack methods such as social networkings, blogs or emails. But this intrusion frequently fail because the victim is not logging into the target website.

III. Existing CSRF Countermeasures

Currently,the Same-Origin Policy has become the primary method to defend CSRF intrusions. Jackson et al. [11] described a comprehensive introduction about the Same-Origin Policy as well as the Session Block. Adida [12] reviewed the current approaches to secure users' sessions such as digesting authentication, locking sessions to IP address. Furthermore he implements a method to block sessions, which is an effective way to defend CSRF attacks. But the limitation of this application is the requirement of JavaScript. Barth et al. [1] reviewed the existing CSRF defences and pointed out the limitation of current CSRF defences such as the secret validation token and the referrer header. There are several defences to detect and prevent CSRF attacks, but few of them could automatically defend this intrusion.

A. Server-side Countermeasures

The countermeasures in server-side mainly concern on CSRF attacks. Jovanovic et al. [13] proposed a server-side proxy called NoForge, which could be pluged into the existing system to detect and prevent CSRF attacks and it is transparent to users and applications. This proxy primarily detects and protects PHP applications against CSRF attacks. Zeller et al. [4] enumerated the characteristics of server-side precautions to protect users. They also developed a plug-in in server side for preventing users from the attacks.

B. Client-side Countermeasures

Most of client-side countermeasures propose a way to extend the function of web browser. Mao et al. [7]proposed a browser-based mechanism that infers whether an authentication token is sensitive and implemented it as a Firefox extension to detect CSRF attacks. Similarly, Johns and Winter [2] developed an application in client-side to defend CSRF attacks called RequestRodeo, which works as HTTP proxy to examine the URLs of the request and response. Also, Maes et al. [14] analysed some crucial requirements to prevent from CSRF attacking in client-side and implemented a stand-alone application to detect and prevent this intrusion.

C. Other Countermeasures

There are also other countermeasures not only in one side but combining the both. Jayaraman et al. [15] exploited a Web DFA model to build a web application which would analyse the users' intent and effectively defend the intrusion of CSRF attacks by combining with non-sensitive GET/ sensitive Post, Secret Token Validation and Intent Verification. Barth et al. [1] implemented a special HTTP header called Origin Header to prevent the leak of sensitive information caused by CSRF login attacks and submited it to W3C. Guha et al. [16] recommended a tool to defend CSRF and CSS attacks in Ajax.

IV. Further Reading

In recent years, CSRF is increasingly concerned by security experts and related reports about this topic are becoming popular according to the analysis of Christey and Martin [17]. Moreover the Open Web Application Security Project (OWASP) [5] has listed CSRF attacks into top ten security risks from 2006 to 2010. Mansfield-Devine [18] points out the risks of CSRF vulnerabilities in social networking according to the dramatic development of Social Networking Services (SNS). Other areas such as cloud computing related to CSRF attacks are mentioned in [6] [19] [20]. Although an effective and automatic defence to guard CSRF attacks is not available now, some countermeasures are still effective to detect and prevent these intrusions such as the model based application in [15], the session authentication based application in [1] which is still processing in W3C organization and a client plug-in extension in [2]. Particularly, the application based on defence model in [15] presents an innovative way to prevent CSRF attacks. There are also some projects available for protecting CSRF attacks, such as CSRF Guard Project which based on unique request tokens [21] and JSCK Project which focused on random code blocks with Javascript [22].

References

[1] A. Barth, C. Jackson, and J. Mitchell, "Robust defenses for cross-site request forgery," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 75–88.

[2] M. Johns and J. Winter, "RequestRodeo: client side protection against session riding," in *Proceedings of the OWASP Europe 2006 Conference, refereed papers track, Report CW448*. OWASP, 2006, pp. 5–17.

[3] T. Alexenko, M. Jenne, S. Roy, and W. Zeng, "Cross-site request forgery: attack and defense," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. IEEE, 2010, pp. 1–2.

[4] W. Zeller and E. Felten, "Cross-site request forgeries: Exploitation and prevention," 2008.

[5] OWASP. (2010, April) Top 10 for 2010. OWASP. [Online]. Available: http://www.owasp.org/index.php/Category:OWASP Top Ten Project

[6] G. Lawton, "Web 2.0 creates security challenges," *Computer*, vol. 40, no. 10, pp. 13–16, 2007.

[7] Z. Mao, N. Li, and I. Molloy, "Defeating cross-site request forgery attacks with browser-enforced authenticity protection," *Financial Cryptography and Data Security*, pp. 238–255, 2009.

[8] D. Gollmann, "Computer security," *Wiley Interdisciplinary Reviews: Computational Statistics*, 2010.

[9] X. Lin, P. Zavarsky, R. Ruhl, and D. Lindskog, "Threat Modeling for CSRF Attacks," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 3. IEEE, 2009, pp. 486–491.

[10] J. Burns, "Cross Site Reference Forgery: An introduction to a common web application weakness," *Information Security Partners, LLC*, 2005.

[11] C. Jackson, A. Bortz, D. Boneh, and J. Mitchell, "Protecting browser state from web privacy attacks," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 737–744.

[12] B. Adida, "Sessionlock: securing web sessions against eavesdropping," in *Proceeding of the 17th international conference on World Wide Web*. ACM, 2008, pp. 517–524.

[13] N. Jovanovic, E. Kirda, and C. Kruegel, "Preventing cross site request forgery attacks," in *Securecomm and Workshops, 2006*. IEEE, 2007, pp. 1–10.

[14] W. Maes, T. Heyman, L. Desmet, and W. Joosen, "Browser protection against cross-site request forgery," in *Proceedings of the first ACM workshop on Secure execution of untrusted code*. ACM, 2009, pp. 3–10.

[15] K. Jayaraman, P. Talaga, G. Lewandowski, S. Chapin, and M. Hafiz, "Modeling User Interactions for (Fun and) Profit: Preventing Request Forgery Attacks on Web Applications," in *Proceedings of the 16th Conference on Patterns Language of Programming (PLoP10)*, 2010.

[16] A. Guha, S. Krishnamurthi, and T. Jim, "Using static analysis for Ajax intrusion detection," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 561–570.

[17] S. Christey and R. Martin, "Vulnerability type distributions in CVE," *V1. 0*, vol. 10, p. 04, 2006.

[18] S. Mansfield-Devine, "Anti-social networking: exploiting the trusting environment of Web 2.0," *Network Security*, vol. 2008, no. 11, pp. 4–7, 2008.

[19] ——, "Danger in the clouds," *Network Security*, vol. 2008, no. 12, pp. 9–11, 2008.

[20] A. Kapadia, S. Myers, X. Wang, and G. Fox, "Secure cloud computing with brokered trusted sensor networks," in *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*. IEEE, 2010, pp. 581–592.

[21] OWASP. (2008, June) Owasp csrfguard project. [Online]. Available: http://www.owasp.org/index.php/Category:OWASP CSRFGuard Project

[22] G. Heyes. (2007, October) Javascript cross site request forgery protection kit. [Online]. Available: http://www.thespanner.co.uk/2007/10/19/jsck/

Bibliography

[23] A. Barth, C. Jackson and I. Hickson.(2009, September) The HTTP Origin Header. [Online].Available:http://tools.ietf.org/id/draft-abarth-origin-03.html

[24] A.van Kesteren.(2010,July) Cross-Origin Resource Sharing. W3C Working Draft. [Online]. Available:http://www.w3.org/TR/cors/

[25] J. Grossman. WhiteHat Website Security Statistics Report. WhiteHat Security, October, 2007.

[26] Fielding, R. and Gettys, J. and Mogul, J. and Frystyk, H. and Masinter, L. and Leach, P. and Berners-Lee, T. (1999, June). Hypertext transfer protocol–HTTP/1.1, RFC 2616.