A Browser-based Cross Site Request Forgery Detection Model

Jingyu Zhang*, Hongchao Hu and Shumin Huo

Information Engineering University, Zhengzhou, Henan, China

*Corresponding author’s e-mail: 1120141933@bit.edu.cn

Abstract. CSRF is one of the most serious cyber attacks, and has been recognized as the main threat and one of the ten most serious vulnerabilities in web applications. The CSRF attack is when the attacker uses the implicit authentication mechanism of the HTTP protocol and the cookies cached in the browser to pass the authentication, and then executes the attack on the target website. In this article, we propose a browser-based CSRF detection model, which can detect CSRF attacks by analysing HTTP requests and web page content, thereby identifying the attacker’s CSRF attacks.

1. Introduction

With the continuous development of Web technology, browsers, as the main carrier of applications and content dissemination, provide users with many conveniences: search engines, community sites, online shopping, online banking, etc., but Web applications are also a high-risk base that causes security problems. The full name of CSRF is Cross Site Request Forgery, which is a common web attack and ranks among the top ten security vulnerabilities counted by the Open Web Application Security Project (OWASP). It has been regarded as the main threat to web applications and one of the ten most serious vulnerabilities in web applications [1].

The principle of CSRF is that the attacker induces the victim to enter a third-party website and sends a cross-site request to the attacked website in the third-party website. The attacker use the registration credentials that the victim has obtained on the attacked website to bypass the user authentication on the website backend to achieve the purpose of impersonating a user to perform an operation on the attacked website.

The CSRF attack method was proposed by foreign security personnel in 2000, but in China, it was not paid attention until 2006. In 2008, CSRF vulnerabilities were exposed in many large communities and interactive websites at home and abroad, such as NYTimes.com (New York Times), Metafilter (a large BLOG site), YouTube and Baidu. Before 2015, CSRF attacks were considered one of the five major network vulnerabilities. Even so, as of 2016, at least 270 CSRF attacks have been reported [4]. Since the CSRF problem appeared in 2000, the information security community has not made much progress in CSRF defense [5]. Now, many sites on the Internet are still unprepared for this, so that the network security community calls CSRF "sleeping giant".

This article first introduces typical CSRF attack methods, and then proposes a browser-based CSRF detection method model design, and gives the module design, and finally discusses related work and summarizes the full text.
2. CSRF detector model design

2.1. CSRF attack process

Figure 1 describes a typical CSRF attack process. The CSRF attack involves three aspects: the victim, the trusted website of the victim who has been authenticated and the attacker.

1. The user logged in to a.com and kept the login credentials (Cookie); 2. the attacker lured the victim to visit b.com; 3. b.com sent a request to a.com: a.com/act=xx. The browser will carry the cookie of a.com by default; 4. a.com will verify the request after receiving the request, and confirm that it is the user’s credential, mistakenly thinking that it is the request sent by the user; 5. a.com to complete the authentication of the victim Act=xx was executed in the name of the attacker; 6. the attack was completed, the attacker impersonated the victim without the victim’s knowledge, and let a.com execute the attacker-defined operation.

The existing CSRF defense methods are divided into two categories [2]:
- Server-based CSRF defense methods, such as verifying Token, using standard verification Same-site Cookie, and using XMLHttpRequest to include other headers. Such methods need to be modified on the server side, which is time-consuming and laborious.
- Client-based CSRF defense methods are divided into two protection technologies: browser add-ons and proxy frameworks. Aiming at a typical CSRF attack, this paper proposes a CSRF attack detection model based on Google Chrome browser and proposes some basic defense methods.

The CSRF detector has the following characteristics: The detector includes HTTP request analysis and content inspection—uses a hybrid mechanism to detect CSRF attacks, and then processes the detected CSRF attacks through the CSRF processing module.
2.2. Model design

Figure 2 shows the model architecture diagram of the CSRF detector. Our CSRF detector will analyse the HTTP request page before loading each page, and detect potential CSRF attacks, and then intercept suspicious HTTP requests and alert the user. It is up to the user to decide whether to load the request.

3. CSRF detector model architecture

3.1. HTTP request analysis module
The HTTP request analysis module mainly intercepts HTTP requests sent by web browsers [3], checks suspicious cross-domain requests, and determines the source domain name by analysing its Referer Header. For resource requests such as Ajax requests, images and scripts, Referer is the page address that initiated the request; for page jump, Referer is the address of the previous page that opened the page history. By analysing the Referer Header, the source domain name that initiated the request can be obtained. The source domain name may be the domain of the website, may be a subdomain, or an authorized third-party domain name, or it may come from an untrusted unknown domain name.

For this domain, subdomain or authorized third-party domain name of the website, it is considered a safe domain name. For untrusted unknown domain names, you cannot simply refuse the request directly.

For example, when a request, such as the homepage of a website, is a page request, the source of the request is search results from search engines such as Baidu, this will also be regarded as a suspected CSRF attack. Therefore, for requests from untrusted unknown domain names, content analysis is required to further determine whether it is a CSRF attack.

3.2. Content analysis module
For each potential CSRF vulnerability page, a hash of the entire page is created. Then, start a new session, request the current page again, create a hash of the entire page again, and match the new hash with the previous hash of the same page. By using the SimHash algorithm to compare two hash values, there is no difference in the hash values of the two pages, which is a safe request. Otherwise, it is considered CSRF vulnerability page.

3.3. CSRF handler module
This module sends out alert messages about all different types of CSRF attacks to users. After the CSRF request page is identified by the content analysis module, the CSRF handler module alerts the CSRF attack type to inform the user that she/he is accessing a suspicious CSRF page, and the user decides...
whether to continue to visit the page.

4. Conclusion
There is no doubt that CSRF attacks are dangerous attacks on the network, and the need for more efficient CSRF detection systems becomes critical. This article introduces the development process and attack process of CSRF, and then we propose a CSRF detection model to detect potential CSRF attacks by analysing HTTP requests and web page content. The detailed design details of the model are also given.

Acknowledgement
This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0804004, and in part by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China under Grant 61521003.

References
[1] Index., (2020)., The-owasp-top-ten-and-esapi-part-6-cross-site-request-forgery-csr. ,http://www.jtmelton.com.
[2] Ziqing Mao, Ninghui L, Ian Molloy “Defeating Cross-Site Request Forgery Attacks ”, 2009
[3] Chunyan Chen (2014) CSRF Attack Principle and Prevention., Computer Knowledge and Technology., (05):902-904.
[4] Dave Fergusonin Web Application SecURLtyon (26, 1,2017)., https://blog.qualys.com/webappsec/2017/01/26/cross-site-request-forgery-what-happened-to-the-sleeping-giant.
[5] Bill Zeller (2019) Popular Websites Vulnerable to Cross-Site Request Forgery Attacks”,. (04):801-803