

Cross-site Scripting attacks protection methods

Yang Ding

Abstract—Cross-site scripting (XSS) attack is one of the most common used method to attack a digital system. It can hide itself in HTML, JavaScript and many other programming languages, with the ability to do anything from gaining information of the victim, to getting full control to the system been attacked. In order to deal with this problem, many developer have figure out different way to protect a system from XSS attack. This paper listed several different approach to counter XSS attack and discuss their strengths and weaknesses.

Index Terms—Cross-site scripting attacks (XSS), cyber security.

I. INTRODUCTION

Cross-site scripting (XSS) attack is a computer vulnerability that allow attacker to insert their own script to the web page to modified the web page in a way they want. Attacker could let browser sending sensitive information of the user, monitoring user action without been noticed, or even gaining access to control user's system. Usually, attacker hide its code in web applications using HTML, JavaScript, Flash and many other programming languages. According to Open Web Application Security Project (OWASP), XSS attack is at the seven place of OWASP top 10 web application security risk in 2017 [1].

II. BACKGROUND

According to the data from CVE security vulnerability database [2], from 1999 to 2019, 12.5% of the recorded vulnerabilities are XSS attacks. In 2019, 1593 vulnerabilities are based on XSS (only considering the vulnerabilities that have been identified), make it the second common vulnerabilities of the year, detailed data shown in the figure 1 below.

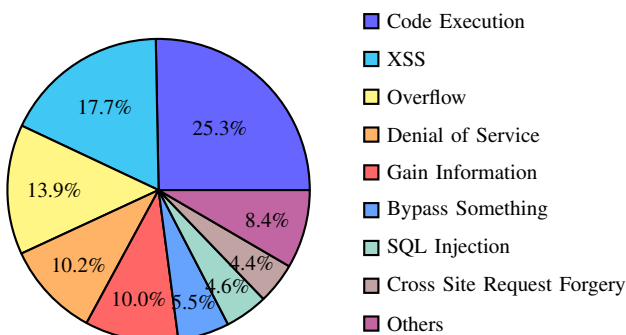


Fig. 1. percentage of different vulnerabilities been found in 2019

III. XSS ATTACK AND POSSIBLE CONSEQUENCE

As XSS can be insert into different programming languages, there are lots of ways to perform a XSS attacks. One commonly used method is to hide the code in a session during user using browser. If the browser doesn't recognized the malicious script and consider the session is from a trusted source, the script will be in the system and provide benefit for attacker to gain the access of user's system. According to OWASP, XSS attacks always occur when data enters a Web application through an untrusted source, most frequently a web request, or when data is included in dynamic content that is sent to a web user without being validated for malicious content [1].

The result of XSS attack can be vary, cause the consequence of been attack is highly depended on what the malicious code could do. For example, if the code is used for hijacking the transmission between client and server, as the result the attacker will be able to get the data been transferred and possibly even change it. If the code is for starting a connection between victim and attacker, then it could end up as attacker have full access of victim's system. In conclusion, the consequence of under a XSS attack has a high range of possibility.

Lots of websites have been found out having XSS vulnerabilities and that include well known websites like Facebook, Twitter, Youtube and ebay. In 2010, famous open-souse foundation Apache was hacked by a hacker using XSS attack and all user password stored on that server was stolen [3]. One more recent case is in January 2019, a game made by Epic games, Fortnite was found vulnerable to multiple attacks including XSS attack. By clicking the URL send by attacker, the gamer account can be took over control and personal information are exposed to the attacker [4].

IV. TYPE OF XSS ATTACK

There are three types of XSS attack, persistent XSS attack, non-persistent XSS attack, and DOM-based XSS attack.

A. persistent XSS attack

Persistent XSS attack (also called type-I XSS or Stored XSS), is the XSS attack that scripts has been inject into vulnerable server through invalid input. The script is then stored in the database, forms, user logs or other data that will be requested by client from server. When client communicate with the server, it will receive data that contain the malicious scripts and then the script can perform harmful action to the client device.

B. non-persistent XSS attack

Also called type-II XSS or reflected XSS. This kind of XSS hide the script into the information in the data that can be reflected from server's error, form or any part that can show a response that include some of information from the request. For example the search bar always show the information you are searching in the result page, this kind of operation is where non-persistent XSS could happen. When victim got a email or message from a website that ask victim to do somethings, like click a link, visit a website, or submit a form, the links/website/form can be modified by attacker and that can send the script with user request, then the vulnerable website will reflect the information been sent back to victim, as the data received from server is the responds from victim's own request, the data will be consider to be safe and the script hide in the data will be executed, which will do what attacker want to the victim.

C. DOM-based XSS attack

DOM-based XSS attack also known as type-0 XSS, it happens as a result of allowing the modification of document object model (DOM) when browser rendering the web page. Consider the hacker inject the script into the parameter in a URL, which is been used only when browser rending the page, then when victim click the URL, the request and response HTML will be no different from normal web page, but when browser rending the page based on HTML's client side script, the DOM can be modified by the parameter and the injection of hacker's script can change how the web page operate and probably run some code for hacker. This is a complete different XSS attack compare to other two, because the script hacker uses not affecting HTML source code from response, only when client rendering the web page, the script will be hid into the page and do what it wants to do, while other two attacks already got script in the code when server sending back the response.

V. DIFFERENT APPROACHES TO PROTECT THE SYSTEM FROM XSS ATTACK

Developer have different ways to protect there system from XSS attacks, none of them are perfect but all those methods have advantages and drawbacks. Five different approaches are discussed in this section.

A. Pattern filtering for user input

One simple way to protect server from XSS attack is to check every user input and us a filter to eliminate all insecure string that has similar pattern to a XSS attack code. Usually the system choose to replace/remove the malicious string. Others will use escaping method, which the string that has special meaning for computer will be modified, or just give a restriction to the input with limited amount of specified input is allowed.

This method has good result dealing with all kind of XSS attacks as the user input is always where XSS is hiding. Imran Yusof and Al-Sakib Khan Pathan build up a pattern

filtering method specifically for persistent XSS attacks [5], which has a very good result filtering the input by filtering different potential XSS attack code including event handlers, data URIs, insecure keywords, escape codes, common word in XSS payload and XSS Buddies. Some examples of the filter logics been used are: using pattern `"/on\w+=—f$command/i"` to find event handlers then replace them to `"` (null), or remove all insecure HTML keywords in user input like `isindex`, `script`, `form` to break the logic of potential XSS code.

According to their paper [5], they used a collection of XSS cheat sheets from the Internet and successfully filtered all patterns in those cheat sheets. This show that there method is effective to known XSS patterns. But as the effectiveness of pattern filtering is highly based on the database of known vulnerabilities, the effectiveness of newly developed pattern or complex pattern is not ideal. It also has a strong demand for updating the database regularly to keep up with new attack pattern, otherwise the filter will become less and less effective.

B. client side pattern matching technique

As mentioned in second section, XSS script always hide inside the user input or data inside the response. This means the malicious code will be send to user browser, so if the browser can identified the script and protect user from been attack, the problem will be solves. Many modern browsers have a function that allow the installation of extensions, and it's possible to create an extension that check all data been send and received, then protect system from XSS attacks.

There is a research on these kind of extension in 2018 [6]. In the paper they developed a Chrome extension called CounterXSS and try to identify three common XSS attack pasterns based on HTML5. The extension can identified direct attack patterns, which is script like `onscroll = "script"` been hide directly into a HTML element, a multiple format attack patterns that use attributes like `src` to directly put URL with modified parameter into the web page, or DOM-base pattern that use attribute like `onerror`, `onload` to modified HTML code using JavaScript when browser rendering the page. The extension can run in the background and will send a alert to user when a pattern has been found. Not much detailed on how the extension performed under multiple testing, but this approach show a new way to prevent XSS.

This research didn't provide a powerful method like the first one provide, but the idea is what can be borrowed. According to NetMarketShare [7], from May 2019 to April 2018, over 68% Internet users is using Google Chrome, taking other Chromium based browsers like new Microsoft Edge and Opera into consideration, the number will be easily above 70%. With all this high amount of user using browser that support Chromium extension, using a extension to provide XSS protect service will be a very effective way to improve the security of whole Internet. But there are also limitations. Just like the extension been developed in the research, extensions are usually small piece of code that couldn't acquire too much processing power, and it can only protect one browser, not the whole system. Client side approach also is said to be not effective toward web content manipulation [9].

C. server side approach

Beside from preventing XSS attack from client side, there are also server side approaches to mitigate the possibility of XSS attack. Usually server side approach need to cooperate with client side browser, server will generate the content that should be generated by browser, check it for any XSS possible content, then send the already generate page to browser. This will increase the work done by server and can also to be found not so effective [8]. So Hossain Shahriar and Mohammad Zulkernine create a new way to perform server side XSS protection based on the concept of “boundary injection” and “policy generation”, witch mainly focus on JSP programs [8].

In their approach, the server look for the dynamic content on every pages, then add a “boundary” with a specified token to it, for every content, it also generate a expected content features (attributes, JavaScript method name and other information of the content) as the “policy” and stored the policy on the server. When a client request for a web page, the server generate the response and then compare it to the boundary and policy of original page, if the feature in the boundary is different from what was expected in policy, the XSS attack is consider to be found and will deal with it using an attack handler program. If no policy deviation, server then check for boundary informations based on tokens, if there are new boundaries found or boundaries with same token, the server will remove all those boundaries. Finally the server sent the response to the server.

In there paper, the false positive rate of there approach vary between 0% - 5.2%, with a 2% - 6% delay compare to normal response due to the processing on server side, they also point out that their method has detect some advanced XSS attack other server side method failed to detect [8]. Compare to other server side method, their method is no doubt a improvement to current method, although the delay on response could be a further work to be improve.

D. static analysis method

Different from other method that focus on improving the system to be more resistant to XSS attacks, static analysis focus on identifying the vulnerabilities in server code. Static analysis usually tracked the input that are not trusty, see how the information flows and see if the data has reach a part, which it can be considered as a part of statement such as HTML or JavaScript statement.

Traditional static analysis can quickly detect XSS vulnerabilities, but it also suffers from problems like always giving wrong positive result [9]. To improve this method, Gary Wassermann and Zhendong Su bring string analysis to the traditional method [10]. They translate the testing code to a static single assignment(SSA) to encode data dependencies, they also use other techniques like context free grammar(CFG) to help improve the checking of blacklisted string values [9].

According to Gary Wassermann and Zhendong Su, there technique has improved the accuracy of static analysis, but couldn't analysis arbitrarily complex code, and the black-list policy rather than white-list policy also cause some problem. What's more, the new method still couldn't fix the weakness

that static analysis won't detect any DOM-based XSS attack [10].

E. static analysis combined with dynamic analysis

With the goal of improving static analysis, developers start to combine dynamic analysis with static analysis, in oder to solve the problems of producing too many false positive result. Based on a open source static analysis project [11], Davide Balzarotti and his colleagues build up a method that use both static and dynamic analysis to give a more accurate vulnerability analysis result [12].

For static analysis, they added a over-approximation for each testing string at every point of the program, so that they can use the value to check whether the string really poses a security risk when reaching that sink of the code. The overall performance is nearly the same. For dynamic analysis, they take the suspicious program path for every string, and try to identify if it can really harm the system. By simulating the program operating the string and check the result, dynamic analysis tried to identified false positive automatically for developer [12].

The method has proved to be more effective than static analysis, but it still has several problems. It contain some programing errors when dealing with regular expressions, the dynamic analysis process can still be insufficient [12]. As it is still a server based analysis, DOM-based XSS attack is still not been tested just like static analysis.

VI. CONCLUSION

Overall, 5 different approaches have been introduce, every-one of them has their own advantage and draw back. Pattern filtering is one of the most easiest way to counter XSS attack and it's very effective to known vulnerabilities. But the needed of a database with a regular update make it hard to enforce after deploy.

Client side pattern matching is deployed on client side, which make it a personal protection method for XSS attack, the pay back is that it won't be able to have complex protection like a normal server does, as client has much less computing power compare to server machine. it also won't detect the attack if web content manipulation is involved [9].

Opposite to client side approach, server side approach can be much more powerful than client side giving more processing power, with more method could be used to detect a XSS attack. Some server side approach is too complex that even need to corporate with client side browser, but there are new approach that are more effective and won't need the help of client browser [8].

Static analysis approach focus on if the vulnerability exist rather than defending an attack. This method need to track down the input string so it's not effective when input string is too complex, it also has issue about high false positive rate, with no ability to find vulnerability related to DOM-based attack [9].

To improve static analysis, developers combine it with dynamic analysis to get better performance. New method have lower false positive rate, but the method couldn't handle

regular expression effectively, and it still not going to test DOM-based XSS attack.

Considering all methods are far from perfect, and XSS attack is still one of the most popular vulnerabilities in the world, which means attacker is only going to find more ways to use XSS in attacking, the importance to develop more effective approach to prevent from XSS attack will keep increasing.

REFERENCES

- [1] "OWASP Top Ten" [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [2] "Vulnerability distribution of cve vulnerability by types" [Online]. Available: <https://www.cvedetails.com/vulnerabilities-by-types.php>
- [3] "Apache Foundation Hit by Targeted XSS Attack — Threatpost" [Online]. Available: <https://threatpost.com/apache-foundation-hit-targeted-xss-attack-041310/73815/>
- [4] "Hacking Fortnite Accounts - Check Point Research" [Online]. Available: <https://research.checkpoint.com/2019/hacking-fortnite/>
- [5] I. Yusof and A. K. Pathan, "Preventing persistent Cross-Site Scripting (XSS) attack by applying pattern filtering approach," *The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, Kuching, 2014, pp. 1-6.
- [6] A. P. Sivanesan, A. Mathur and A. Y. Javaid, "A Google Chromium Browser Extension for Detecting XSS Attack in HTML5 Based Websites," *2018 IEEE International Conference on Electro/Information Technology (EIT)*, Rochester, MI, 2018, pp. 0302-0304.
- [7] "Market share for mobile, browsers, operating systems and search engines — NetMarketShare" [Online]. Available: <https://netmarketshare.com/>
- [8] H. Shahriar and M. Zulkernine, "S2XS2: A Server Side Approach to Automatically Detect XSS Attacks," *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, Sydney, NSW*, 2011, pp. 7-14.
- [9] L. K. Shar and H. B. K. Tan, "Defending against Cross-Site Scripting Attacks," in *Computer*, vol. 45, no. 3, pp. 55-62, March 2012.
- [10] G. Wassermann and Z. Su, "Static detection of cross-site scripting vulnerabilities," *2008 ACM/IEEE 30th International Conference on Software Engineering*, Leipzig, 2008, pp. 171-180.
- [11] N. Jovanovic, C. Kruegel and E. Kirda, "Pixy: a static analysis tool for detecting Web application vulnerabilities," *2006 IEEE Symposium on Security and Privacy (S&P'06)*, Berkeley/Oakland, CA, 2006, pp. 6 pp.-263.
- [12] D. Balzarotti et al., "Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications," *2008 IEEE Symposium on Security and Privacy (sp 2008)*, Oakland, CA, 2008, pp. 387-401.