2012 IACSIT Hong Kong Conferences
IPCSIT vol. 29 (2012) © (2012) IACSIT Press, Singapore

Performance Comparison of XSS Mitigations based on Platform and Browsers

Ravi Kanth Kotha[1+] , Dinesh Naik[2] and Gaurav Prasad[3]

Information Technology National Institute of Technology Karnataka, India

Abstract. In the recent years, everything is on web. It may be Organization's administration software, Custom ERP application, Employee portals or Real estate portals. The Social networking sites like Face book, Twitter, MySpace which is a web applications being used by millions of users around the world. Hence they are observed and may be exploited by hackers. Researchers and industry experts state that the Cross-site Scripting (XSS) is the one of the top most vulnerability in the web application. XSS consists in the exploitation of input validation flaws, with the purpose of injecting arbitrary script code which is later executed at the web browser of the victim. According to OSWAP, Cross-site scripting attacks on web applications have experienced an important rise in recent year. This demands an efficient approach on the server side to protect the users of the application. While the reason for the vulnerability primarily lies on the server side, the actual exploitation is within the victim's web browser on the client-side. Therefore, an operator of a web application has only very limited evidence of XSS issues. However, there are many solutions for this vulnerability. But such techniques may degrade the performance of the system. In such scenarios, challenge is to decide which method, platform, browser and middleware can be used to overcome the vulnerability, with reasonable performance over head to the system.
Motivated by this problem, we present performance comparison of two mitigation techniques for Cross-site Scripting (XSS) at the server side based on the parameters like application's platform, middleware technology and browser used by the end user. We implemented Mitigation PARSING technique using database and REPLACE technique in different platforms, middleware's and checked its performance. We calculated the time taken by different browsers to render the pages using two techniques under different platform and middleware. In this paper we proposed the best combination of development platform, browser and the middleware for the two mitigation technique with respect to developer and end users

Keywords: Mitigation PARSING technique and REPLACE technique XSS attack, Platform, Middleware, Browser.

1. Introduction

Today web application is considered the backbone of every activity in internet. Almost all information are available in internet through web applications. A survey conducted in 2007 estimated that 70% of the web applications are at risk of being hacked [1] and the fact that these applications can be accessed from everywhere in the globe, makes them even more interesting for attackers . One such vulnerability is Cross Site Scripting [2][3]. According to Open Web Application Security Project.(OWASP) , web application level vulnerabilities have a major part in getting attacked by any hacker. Mainly the attacks are of injection type. Cross-Site Scripting is one of the main and important attack to be focused on as mentioned OWASP[4]. The Cross Site Scripting attack is performed by changing the logic, semantics or syntax of a HTML tag by inserting new keywords or operators. XSS Attack is a class of code injection attacks that happens when there is no proper input validation. Attacker can shape their illegitimate input as parts of formal scripts string which operate by databases. Many web applications like social networking sites or any web applications

[+] Corresponding author.
E-mail address: ravikanth1027@gmail.com.

could be the victims of this attack. Attacker by exploiting this vulnerability will be a threat to security attributes like confidentiality, integrity and authorization.

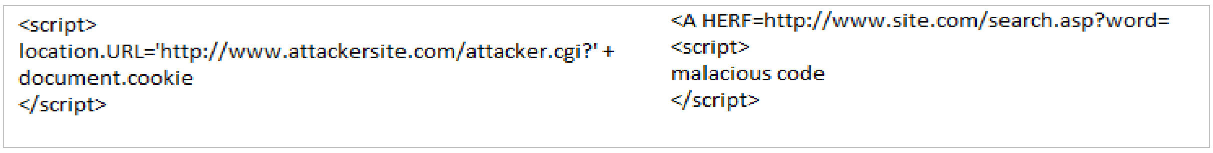
The problem is that some current techniques and tools are impractical in reality because they could not address all types of attacks or have not been implemented yet. These techniques require modification in the original code or addition of some modules into the application. However these methods will not be full proof and result in performance degradation. So, there is a need to find the best combination of platform, browser and mitigation.

The structure of the paper is as follows. Section 2 presents some background and discusses about XSS attack Section 3 describes in details the two mitigation methods implemented and compared performance based on different factors. Section 4 presents the implementation details and discusses the results .Finally section 5 concludes the paper.

2. Cross Site Scripting

Web applications are open to all and has wider attack surface .Even Attackers only need a web browser to access them and perform attack on them. If we are taking DREAD [8] as a mechanism or a tool to evaluate the severability of the systems then it is very high. The main reason is the attack can be reproduced easily and even can be performed using a web browser. If the attack is performed many user will be affected by it. Additionally, such vulnerabilities can serve as launching pads for other, more severe attacks on web users' local systems. Cross-site scripting (XSS) is the class of web application vulnerabilities in which an attack is caused by a victim's browser to execute malicious script from the attacker. In 2006, 21.5% of all newly reported vulnerabilities were XSS, making it the most frequently reported vulnerability of the year [6,7].

There are two basic techniques to accomplish an XSS attack .The first technique is to store malicious code in database and when accessed by client will be executed by the browser at the client side. The second technique requires that the victim without the knowledge of malicious link clicks on the link resulting in execution of malicious code.



```
<script>
location.URL='http://www.attackersite.com/attacker.cgi?' +
document.cookie
</script>
```

```
<A HERF=http://www.site.com/search.asp?word=
<script>
malacious code
</script>
```

Fig 1: Code for accessing cookie and Malicious inserted into the tag

3. Mitigation Techniques

There are various methods and techniques for preventing XSS attack. The mitigation techniques for the cross site scripting attack can be implemented at two sides SERVER SIDE and CLIENT SIDE. The CLIENT SIDE mitigation technique is implemented on the browser. Usually in this type of mitigation technique which is done at CLIENT side the main idea is to parse the incoming script and do proper validation. The browser which validates the scripts and then executes provided script is free from malicious code or results in malicious activity. And thus the client side mitigation helps to reduce the overhead of the web server by doing validation at browser side.

Though the client side mitigation technique seems to be good enough there are some limitations. The client side mitigation can be done using scripting languages, thus making the attacker to view and gives a chance to the attacker to try different attack vectors. In the client side every user has to undergo the overhead of making changes to the validation techniques where as in server side it is done only once. Hence the client side mitigations are not recommended. There are many server side mitigation techniques available. SERVER before sending the script to the intended client performs the mitigation Here, two server side mitigations are discussed and compared using different parameters.

3.1. Mitigation with support of database

The Implemented mitigation solution comprises of four components namely Blocker, parser, verifier and black listed tag cluster (Database). The Research paper Authors used XML to store the Black listed tag clusters but it is again difficult to add some more black listed (TAG, ATTRIBUTE) sets when compared to Database.

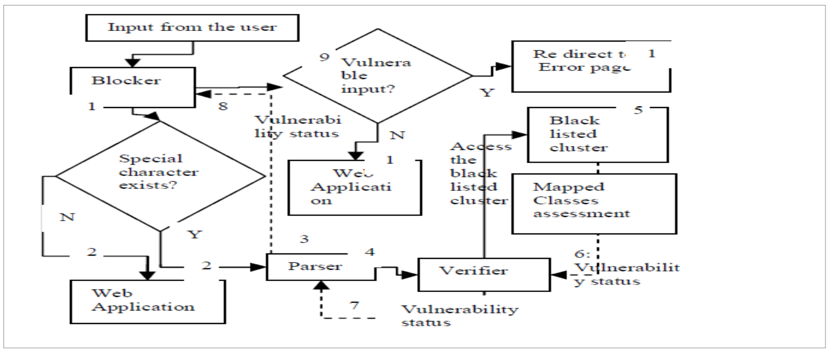


Fig 2 : Flow of input through the components

The remaining Procedures are followed as the paper proposed and the implementation of NOTIFICATION procedure is developed and added to notify the details of attacker and attack to the victim user (user to be attacked if mitigation is not present).

Blocker: When HTTP Request is received by the SERVER, those requests will redirected to this procedure, If any characters like { '<', '>', '%', '&', '\\', '#', '/'} are present in the Request, it sends the Request to PARSER to get the TAGS and ATTRIBUTES used in the Request otherwise it redirects again to the Application. If the Parser gives the Vulnerability status as *true* then an ERROR page will be send to the attacker and notification sends to the victim. If the status is *false* request will handled by the Application. **Parser:** This procedure creates the vector (tag, key, value) for each HTML or OTHER scripts in the request and sends each Vector to the Verifier to check whether given TAG is Malicious or not. **Verifier:** This procedure uses the Blacklisted Tag Cluster (Database) to check each vector given by parser contains any vulnerable scripts or not. If any of the vector falls true for vulnerable, It stops verifying and returns *true* to the Blocker. **Block Listed Tag Cluster (Database):** Here a rules table which contains the data as (TAG,KEY,CONDITION,VALUE) and it can updated by the Administrator and he can add new Block Listed Tags as he noticed. The many Common Block Listed Tags are already inserted to the Database .

3.2. Solution by Replacing the Code:

The script shown in figure 1 gives a idea of how the cookies can be stolen. It looks like a good idea of delimiting malicious words and deleting them from the script. However the incoming script can be carved effectively to deceive the idea. This can be viewed in figure:

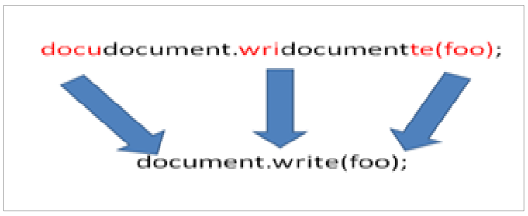


Fig 3: Exploiting the delete technique

The technique is to change those words in such a way that the pronunciation is same where as the spelling is different, instead of deleting the words. This makes the browser to consider the malicious code to be normal words. For example, the word “document” can be replaced with “dokument” so that the pronunciation remains same but will not execute the script. By this method the parsing need not to be done repeatedly because incase of deletion the algorithm followed is Least Common Subsequence which is time consuming process leading to performance degradation. Whereas here a REPLACE function is used such that there is no effect on the time of execution. . Thus making the script non executable by the browser.

4. RESULTS

The two techniques discussed are implemented in both Microsoft Visual Studio .NET 3.5 Framework and JAVA Standard Edition 6. Browser used to compare are Mozilla Firefox 3.6, Google Chrome and Internet Explorer. Here MYSQL is used as storage database.

4.1. Comparison of the Two Techniques

The two proposed mitigation techniques are compared by taking the execution time of the both techniques. The result related to the implementation of mitigation in java platform is shown in the figure

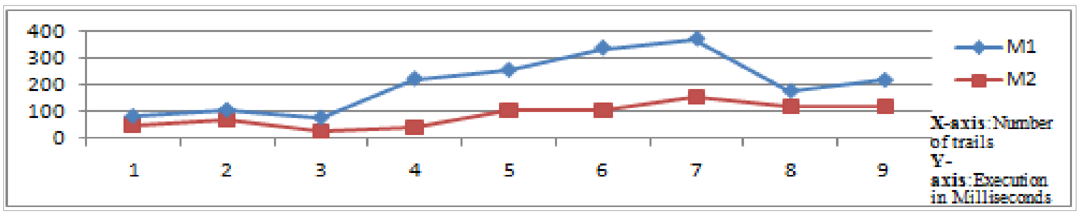


Fig 4: The comparison between Mitigation using data base and REPLACE Function

From the results it is infer that mitigation using database takes much time in execution than the replacement mitigation. The execution time can be explained by databases connectivity and the time parsing the script. And a small part is even contributed by the housekeeping work done at the time of function calls. Here the backend database used is MYSQL which is connected using JDBC connector. In mitigation with replacement technique the main share of the execution time is the FIND and REPLACE functions. For implementation built-in functions are used. From the results it shown that mitigation with replacement is performing better than the earlier.

4.2. Comparison of Mitigation Using Database

The discussed mitigation technique which uses database is compared with the JAVA and C# implementation. By making the database as fixed the techniques are implemented and compared. The below results shows that MYSQL with JAVA as the developing platform performs better than the C# .

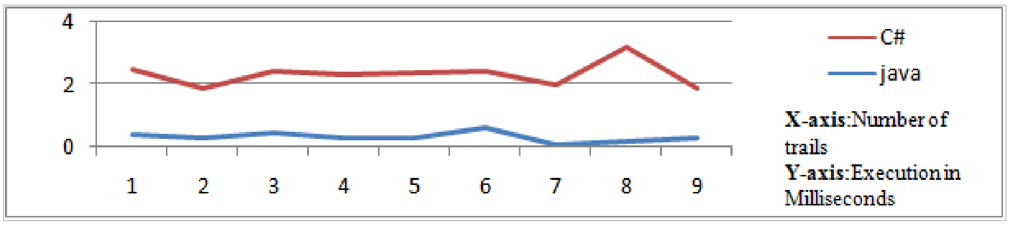
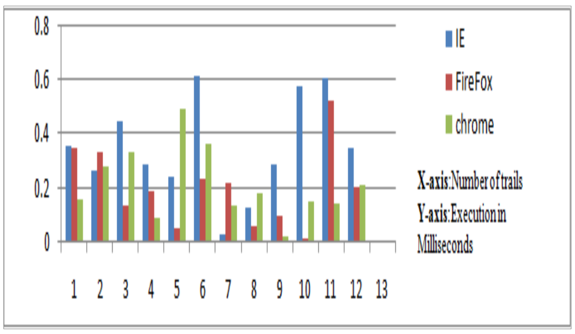


Fig 5: The Platform comparison for Mitigation using Database.

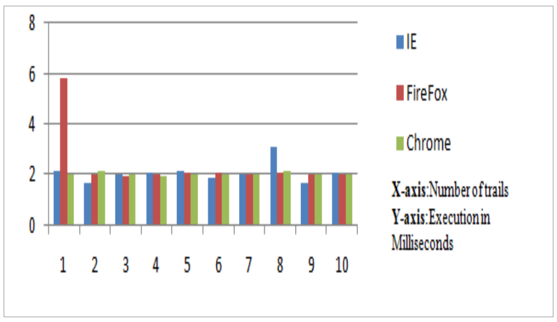
The graph clearly shows that time of execution for the proposed mitigation in C# is very high when compared to java .

4.3. Browser Comparison for Two Mitigation Using C# and JAVA

The same mitigation techniques were also observed using three different browsers to find the best browser based on the platform of the application. The graph shows that Internet Explorer and Chrome which are the considered good browsers in market perform almost at equal level because of their single java engine From Bench Mark Battle a standard benchmark for comparing browsers has given almost same results[6]. The chrome‘s java engine Grease Monkey even in enhanced version also consumes same amount of time as the boot time of the chrome contributes to it. Thus performing equal to Internet Explorer which is having comparatively less boot up time. The comparison is even carried out for the C# implementation of the mitigations. In general notion the products from same vendor perform well. But in contrast Internet Explorer and C# which are from same vendor MICROSOFT seems to be not well supportive to each other. The argument can be supported from the below results.



JAVA

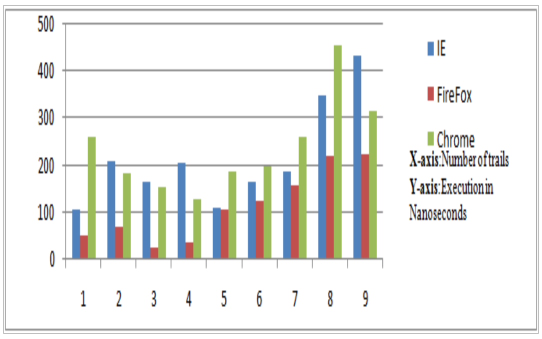


C#

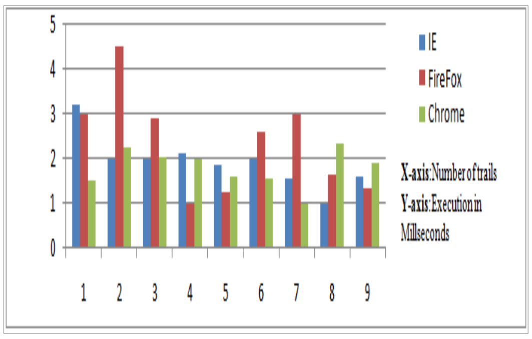
Fig 6: Mitigation Using Database in C# and Java

From the above chart it can be infer that Google Chrome and Internet Explorer are having same execution time for the C# implementation. This can be explained because of the chrome's process isolation and the add-ons. The features such as dynamic code generation, hidden class transitions, and precise garbage collection have made the execution time near to Internet Explorer.

The mitigation technique using REPLACE function is also compared based on the factor browser. The technique is implemented both in C# and JAVA and deployed into the three browsers.



JAVA



C#

Fig 7: Mitigation using REPLACE in C# and Java

The mitigation technique using FIND and REPLACE function is implemented using JAVA and C# ,compared based on the browser to be used for rendering. From the graph it is clear that the browser Mozilla Firefox takes very less time in rendering the JAVA implementation code compared to Google Chrome and Internet Explorer. The result also shows the browsers Internet explorer and Google chrome takes same time in rendering the ASP pages. This can be justified from the chrome handling of the C# code. Firefox lags in this results showing the huge deployment time for C# code.

5. Conclusion

The two server side mitigation techniques has been discussed and compared based on the platform, browser. Our work on comparison of mitigation techniques shows that the mitigation using REPLACE function performs better than the other. The java implementation of database mitigation technique performs better than C#. Among the three leading browsers say Google chrome and Internet Explorer perform almost equal either in JAVA or C#, but Fire Fox overtakes them when these two mitigation techniques discussed are considered..From all the results obtained we can clearly infer that an web application with mitigation technique with REPLACE function implemented in JAVA can give a good performance in browser FIREFOX.

6. Acknowledgements

I wish to express my deep sense of gratitude to Asst Prof. Dinesh Naik and Mr Gaurav Prasad, Department of Information Technology, National Institute of Technology Karnataka for giving support in doing this project.

7. References

[1] Acunetix Ltd, Feb. 2007, http://www.acunetix.com/news/securityauditresults.htm

[2] The IT security.com Dictionary of Information Security," Cross Site Scripting (XSS, cross-site malicious content)"

[3] "CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests

[4] OWASP Top 10-2010 rdc.

[5] Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks José Fonseca Marco Vieira, Henrique Madeira

[6] S. Christey. Vulnerability type distributions in CVE, Oct. 2006

[7] K.J.Higgins. Cross-site scripting: Attackers' new favorite flaw, September 2006. document.asp?doc_id=103774&WT.svl=news1_1.

[8] Security-aware Software Development Life Cycle (SaSDLC) – Processes and Tools Asoke K Talukder1, Vineet Kumar Maurya1, Santhosh Babu G1, Jangam Ebenezer1, Muni Sekhar V1, Jevitha K P1, Saurabh Samanta1, Alwyn Roshan Pais