Analysis of XSS attack Mitigation techniques based on Platforms and Browsers

Ravi Kanth Kotha, Gaurav Prasad and Dinesh Naik

Department of Information Technology
National Institute of Technology, Karnataka
ravikanth1027@gmail.com, chgauravprasad@gmail.com,
dinnaik@gmail.com

ABSTRACT:

In the recent years, everything is in 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 application is been used by millions of users around the world. So web applications have become very popular among users. 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 vulnerabilities in the web application. The cross-site scripting has become a common vulnerability of many web sites and web applications. 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 as 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 vulnerabilities, with reasonable performance over head to the system.

Inspired 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 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

Parsing mitigation technique, replace technique, XSS attack, platform, middleware, browser, JAVA,C#.

David C. Wyld, et al. (Eds): CCSEA, SEA, CLOUD, DKMP, CS & IT 05, pp. 395–405, 2012.
 DOI : 10.5121/csit.2012.2240

1. INTRODUCTION

Today everything is in internet. It has become part of every individual's daily activity. It may be related to information gathering or any other work. Web application is considered the backbone of every activity in internet. Almost all information are available in internet through web applications .The business applications (e-commerce, banking, transportation, web mail, blogs, etc) are now available as web-based applications. The demand for web applications also attracts adversaries that want to exploit the vulnerabilities. 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 .

Attacker is an individual whose main interest or aim is to obtain confidential data by performing malicious activities. He finds the vulnerabilities in the system and exploits it to gain information of the victim .There are many vulnerabilities in dynamic web applications. 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 by OWASP[4].

The Cross Site Scripting attack is performed by changing the logic, semantics or syntax of a HTML tag [11] 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 script string which is operated by databases. Many web applications like social networking sites or any web applications could be the victims of this attack. Attacker by exploiting this vulnerability will be a threat to security attributes like confidentiality, integrity and authorization.

XSS can also escape traditional tools such as firewalls and Intrusion Detection Systems (IDS) because they perform through ports used for regular web traffic which usually are open in firewalls. On the other hand, most IDS focus on the network and IP layers whereas XSS work at application layer. Researchers have proposed a range of techniques and tools to help developers to compensate the shortcoming of the defensive coding. Developers have to undergo some defensive coding practices to eliminate such vulnerabilities. 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 the performance based on different factors. Section 4 presents the implementation details and discusses the results .Finally section 5 concludes the paper.

2. XSS ATTACK

Web application attacks are of greater impact than any other kind of applications and software. Since it is 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 [6] as a mechanism or a tool to evaluate the severity 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 [7,8,9].In the year 2010 it still maintains the same place or ranking.

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>
```

Figure1: Code for accessing cookie

```
<A HREF=http://www.site.com/search.asp?Word= <script>
malicious code </script> >
```

Figure 2: Malicious inserted into the tag

2.1. CAUSES OF XSS VULNERABILITIES

In the category of common vulnerabilities , XSS in the internet community from several years XSS is vulnerability of a web application caused by the failure of the application in checking upon user input before returning it to client's web browser. Several factors contribute to the prevalence of XSS vulnerabilities. First, the system requirements for XSS are minimal: XSS afflicts web applications that display untrusted input. Secondly most web application programming languages provide an unsafe default for passing un trusted input to the client. Typically, printing the entrusted input directly to the output page is the most straightforward way of displaying such data. The third factor is proper validation for untrusted input is difficult to get right, primarily because of the many, often browser-specific, ways of invoking the JavaScript interpreter.

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. Sever 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 proposed system [10] in figure 3 is to have the features like, Configurable black listed tags, its attributes and object implementation procedure for misuse detection at the server side, and hence the existing web pages need not be modified for new threats.

Whenever a new web page is introduced there is no need to modify the web page, since the security mechanism is separated from page level implementation and is placed at the top most layer of the web application.

Security administrators need not know the entry points of individual web pages as there is a clear demarcation between the web application and security mechanisms implemented in this approach.

The Implemented solution (Mitigation) 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. 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).

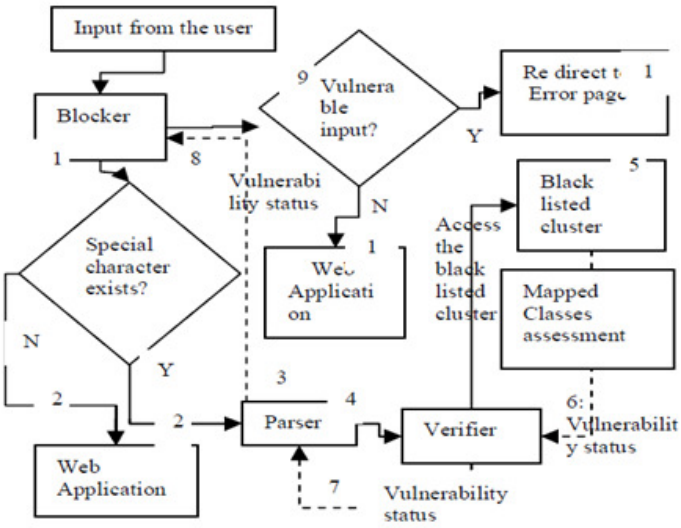

Figure 3 : Flow of input through the components

Blocker: When HTTP Request is received by the SERVER, those requests will redirected to this procedure, If any charecters 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 Vulnerabilty 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 Maliciousor not.

Verifier: This procedure uses the BlockListed 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.

BlockListed 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 BlockListed Tags as he noticed. The many Common BlockListed 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:

Figure 4: Exploiting the delete technique

The existing system the mitigation is done by parsing the whole data and deleting the words which may disclose the cookies or any escaping words .This process is repeated until there are no more malicious code or words.

The technique [11] 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. IMPLEMENTATION

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

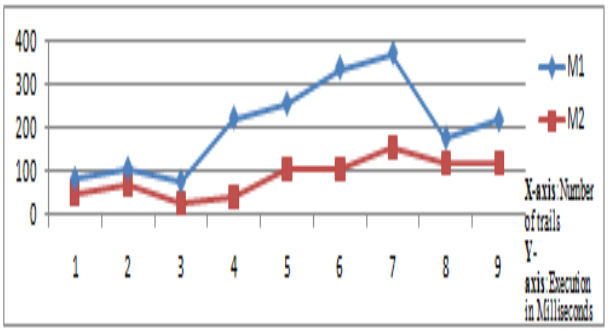

Figure 5: The comparison between Mitigation using database and REPLACE Function

From the results it is infer that mitigation using database(M1) takes much time in execution than the replacement mitigation(M2). 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. But the drawback of this technique can be explained as it won't allow any normal script that is the scripts used to display images and some links .

4.2. COMPARISON OF JAVA AND C#

The two techniques which were been in discussion are implemented in JAVA and check for the execution time.

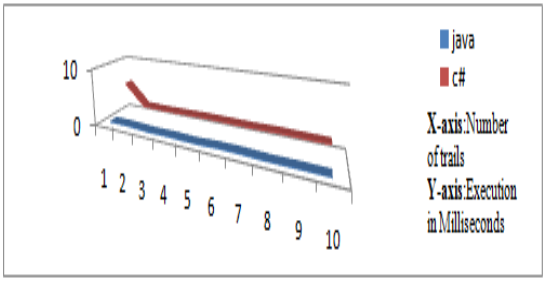

Figure 6: Platform Comparisons for REPLACE method.

The above graph is the execution time of the mitigation technique with REPLACE function. The t problem with REPLACE in C# is that it creates more string copies each time it is called. This can often lead to measurable performance problems in large projects. Microsoft notes that "This method does not modify the value of the current instance. Instead, it returns a new string in which all occurrences of old Value are replaced by new Value." Following picture from the MSDN

benchmark shows the result of execution of REPLACE function for short(20 characters) and long (1000 characters)[12].

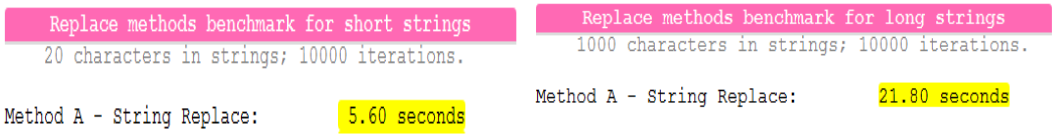

Figure 7: The Time of execution for replace methods

In Java the REPLACE function is implemented is different manner. Here the REPLACE function creates only one new string with replacing all the occurrences of the source string. And hence the performance of the function is better than in C#.

4.3. Comparison of Mitigation using Database

The discussed mitigation technique which uses database is compared with the JAVA and C# implementation. Here the BACK END database is MYSQL which is an open source database available. 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# .

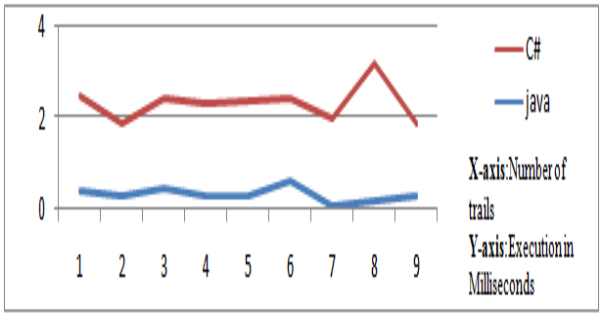

Figure 8: The Platform comparison for Mitigation using Database.

The graph clearly shows that time of execution for the proposed mitigation in C# is very high when compared to java .Since In Java the connection to database is done using JDBC connectors. The connection handling in java is implemented in an efficient method. Java uses a concept called CONNECTION POOLING where the recently used connection for the database is stored. For a web application which can have thousands of client requests for the database are connected using connection threads from this pool. Hence in java the execution time for SQL connection very low.

C# also uses concept of CONNECTION POOL but the process of creating and establishing connection is done using a process. The major part of the execution time for connection establishment is for the process creation and killing the process after task

4.4. Comparison Of Browsers

The same mitigation techniques were also observed using three different browsers to find the best browser based on the platform of the application. Firefox3.6 the leading browser in the market which uses four java engines called SPIDERMONKEY is found to be the best for java based application. The below graph can strengthens the above statement.

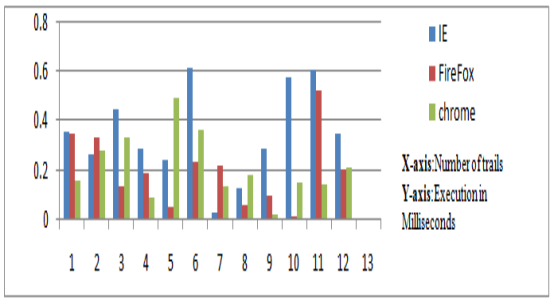

Figure 9: Mitigation Using Database In JAVA

The graph also shows that Internet Explorer and Chrome which are the considered good browsers in market perform almost at equal level.

The performance of IE and chrome is almost similar as the support for java from both the browsers is done by using single java engines only. From Bench Mark Battle a standard benchmark for comparing browsers has given almost same results [13]. The chrome's java engine Grease Monkey even in enhanced version also consumes same amount of time. This can be explained by the fact that the major part of the execution time is the boot time of the chrome. 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.

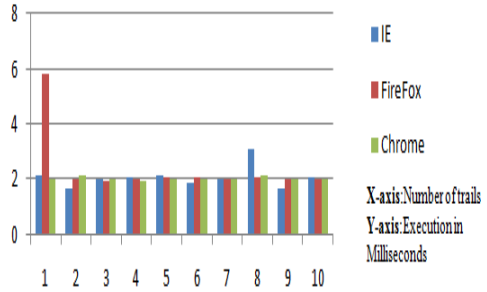

Figure 10: Mitigation using database in ASP

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.

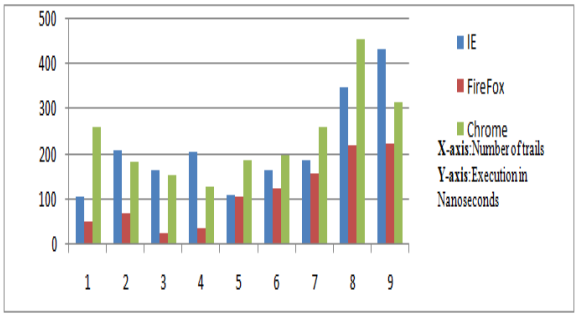

Figure 11: Mitigation using REPLACE function implemented in JAVA

The mitigation technique using FIND and REPLACE function is implemented using JAVA and the 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 implementation code compared to Google Chrome and Internet Explorer.

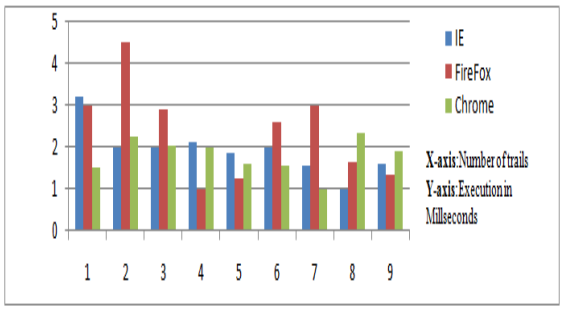

Figure 12: Mitigation Without using Database in C#

The same technique implemented in C# and compared for browser comparison. The result shows the browsers Internet explorer and Google chrome takes same time in rendering the 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.

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)", http://www.itsecurity.com/dictionary/xss.html

[3] CERT Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests, http://www.cert.org/advisories/CA-2000-02.html

[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 13th IEEE International Symposium on Pacific Rim Dependable Computing

[6] 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

[7] S. Christey. Vulnerability type distributions in CVE, Oct. 2006. http://cwe.mitre.org/documents/vuln-trends.html.

[8] K. J. Higgins. Cross-site scripting: Attackers‟ new favorite flaw, September 2006. http://www.darkreading.com/ document.asp?doc_id=103774&WT.svl=news1_1.

[9] Common Vulnerabilities and Exposures, "The standard for information security vulnerability names," http://cve.mitre.org/,last accessed May 24, 2007

[10] Risk Mitigation for Cross Site Scripting Attacks Using Signature Based Model on the Server Side Jayamsakthi Shanmugam, Dr.M.Ponnavaikko

[11] XSSDS: Server-side Detection of Cross-site Scripting Attacks , Martin Johns1, Bjorn Engelmann2, and Joachim Posegga, 2008 Annual Computer Security Applications Conference

[12] CERT® Advisory CA-2200-02, "Malicious HTML Tags Embedded in Client Web Requests," http://www.cert.org/advisories/CA-2200-02.html, last accessed May 24, 2007.

[13] Battle of Benchmarkhttp://download.cnet.com/8301-2007_4-20047314-12.html

AUTHORS PROFILES:

Name: RAVI KANTH KOTHA
Obtained his Bachelor of technology degree from KITS Warangal and presently doing Master of Technology in Department of IT at NITK Surathkal. His area of interests are web security, network security, web mining.

Name: GAURAV PRASAD
Working as lecturer in department of IT NITK Surathkal. He has done his M.Tech from computer science department NITK Surathkal with specialization in information security .His area of interests are Information security , cloud security, data security ,grid security and web security.

Name: DINESH NAIK
Working as Asst Professor at Department of IT at NITK Surathkal. Obtained Btech degree from VBDTCE- Dawangar and M.Tech from NITTE .He has been in the area of teaching since 2001 His area of interests are computer network, information security, web services, web mining