Impact Analysis of Preventing Cross Site Scripting and SQL Injection Attacks on Web Application

Rathod Mahesh Pandurang
Dept. of Computer Engineering
Sardar Patel Institute of Technology
Mumbai, India
Email: m.rathod27@gmail.com

Dr. Deepak C. Karia
Dept. of Electronics Engineering
Associate Professor
Sardar Patel Institute of Technology
Mumbai, India
Email: deepak_karia@spit.ac.in

***Abstract*—Web applications provide immeasurable large facilities to the users. The usability and popularity of web applications have expanded. This has caused various types of attacks over them. SQL injection and XSS (Cross Site Scripting) attacks are very famous to exploit the web applications. To sneak into the web application database, one can use SQL injection attack that may cause database alteration or imparting vital details while XSS is one more threat in which malicious user tricks the input data given that may steer to the modification in webpage viewing or redirection of user to attacker's working space. The proposed Intrusion Detection System is a container based approach that is based on a mapping model. In this, a request to query mapping is applied to recognise and prevent such class of attacks. The container based approach to identify two different client requests have been used. The impact measurement of this container based approach on the web server is calculated using http_load and autobench tool. The web application performance measurement based on various parameters such as average page time, pages per second, memory and processing time for container based approach has been carried out and compared with the existing approach.**

Index terms– Intrusion Detection System (IDS),SQL Injection Attack, Cross Site Scripting (XSS) Attack, Mapping model, http_load, Autobench

I. Introduction

Internet services and applications are tremendously used these days. These applications use and manage the personal data of the user. To manage this increased usage, web services are designed on the basis of multi-tiered design. Because of increase in the use of such web services, various attacks on it has also increased. A class of IDS that machine are currently using can also detect unknown attacks by identifying those network traffic, which diverges from the typical way of behaving which is previously defined during the training phase of the IDS [1]. To understand the nature of attack, we have to first understand the architecture of web application.

A web application is the system provided for the user to interact with the service. A three tier web application has multiple clients in the front-end of the website, a web-server at middle level and a database server at back-end. The client sends the request to the web service that sets of the subsequent database queries. The database responds with a appropriate reply to the web server which in turn replies to the client.

Numerous types of attacks are possible on the three tier web applications. Even a simple application logic error gives opportunity to hackers to get into the system [2], [3]. Checking for various vulnerabilties and providing solution for it is an important part for any web application [4], [5], [6]. The threats under consideration are XSS (Cross Site Scripting) Attack and SQL Injection Attacks [7]. XSS is an attack which concentrates on embedding unwanted data on the valid web application. This inserted data can be a URL link to attacker's web application, using which attacker can trick the user to give away it's crucial data. SQL injection is a code injection attack in which attacker targets the database of the system to access data from it or to change the valid data with ambiguous data of his/her own [8].

For impact measurement of our Intrusion Detection system on the web server we are going to use two tools. The performance metric for measuring can be throughput of the server, response time [9]. The tools under consideration are: http_load [10] and Autobench [11].

The http_load is a useful HTTP benchmarking utility that entitles user to run numerous http fetches in parallel so that the throughput of the web server can be tested. It gives user an estimate of serving capability of a server (in bytes) in a particular time. To test, user has to save the urls before running the tool in a text file – one link per line, which is then pass to the program. The program fetches the urls as fast as it can. The speed of the net connection also affects the end result [10].

Autobench is a Perl script which automates the process of evaluation a web server (or is used for managing a relative testing of two different types of web servers). Autobench is a sheath around the tool named httperf. It runs httperf against every host many times. It increases the number of requested connections per second on every cycle and plucks out the vital information from the output of the httperf. It delivers a CSV or TSV type of file for analysis of the result [11].

In this paper, working of the Intrusion detection system is shown. Each client is given a separate container for their session so that the damage caused by the attacker is limited to that container only. The impact measurement of container based server is compared with vanilla web server. The web application performance measurement based on average page time,pages per second, memory and processing time is done.

The paper is organised as follows: Section II shows the related work of the area. Section III gives general discription of the algorithm used. Section IV talks about the mapping model used. Section V points out the implementation done and

978-1-4673-9542-7/15/$31.00 ©2015 IEEE

experiments carried out. Section VI shows the results. Finally Section VII concludes the results obtained.

II. Related Work

Meixing Le et al put forth a system that employed mapping model for detection of SQL injection attacks. The authors have used lightweight virtualization technique to contain the users. Approach defined efficiently uncovered the threats, but the overhead for lightweight virtualization was lot. This puts limit on the number of concurrent users which can use this system [12].

On the other hand Lwin Khin Shar et al put forth a technique to detect and prevent XSS attacks. They eliminated XSS vulnerabilities from the code in two parts. In first detection part, to track the flow of user data in HTML output statements, taint-based analysis was done and any malicious statements or comments was seen. In second part pattern matching mechanism and data dependency was examined to avert the injection in code due to XSS [13].

One of the other way to recognise threat was presented by R.Priyadarshini and Jagadiswaree. They have proposed a work which includes five phases to reduce SQL Injection and XSS to some level. In the first phase IDS (Intrusion Detection System) and APIs (Application Programming Interface) are used to detect attacks, attacks are prevented with the help of IDS in the second phase , in third phase attack database log are maintained, and in fourth phase WAPT (Web Access Pattern Tree) is used for verifying record and in the fifth phase report generation which is done using P chart techniques. This work is done in JSP, ASP.NET [14].

C. M. Frenz et al proposed a model in Perl to detect XSS attacks over web applications. This model is mostly useful for user content driven web applications like web forums as this IDS assumes content to be executable, which would be in heading tags or paragraphs. Regular expressions are used to detect XSS in web application. This approach is not that much resilient even though this is a decent approach in Perl [15].

A SNORT based approach was put forth by Alnabulsi and Islam. They suggested detection technique which employs SNORT tool by adding SNORT rules. It is Successful for SQL injection but prone to XSS [16].

The approach defined in [12] was heavily loading the server and it limited total number of concurrent users and also it focused on SQL Injection attack only. The method used in [15] was Perl based approach useful for content driven application but was not that robust. The IDS in [13] had two staged classifications of XSS attacks and it had a drawback of higher computational time. So our aim will be, to design such an Intrusion Detection system which will overcome these limitations by minimizing the server load and increasing the execution time. For that reason we are going to build a mapping model.

III. Algorithm

Two web applications: static and dynamic are developed. The static web application will have file upload and download operation while the dynamic web application will be a simple blogging site. The static web application will have a deterministic mapping model of request and query and dynamic application will have non-deterministic mapping model.

1) The client accesses the static web application.
2) The client clicks on a file download option which steers it to another page where that particular file is placed. The url points to the file id and its location.
3) This web query in the form of the url is examined whether it is a valid one or not.
4) If valid, the file is allowed to be downloaded else alert is generated.
5) The client uses the dynamic web application in the form of blog by registering. The client is given separate container as soon as it browses the blog.
6) The login field and the comment field can be used to attack the front end or it can be used to attack the back-end of the application. Input to these field is examined whether it contains any malicious data. If the input data is valid then request is allowed to proceed else alert is generated.
7) The record of the user activity is saved in the database which can be used for the examining purpose.

IV. Mapping model

A. Static Mapping model

Static web application (SWA) employs the deterministic approach for mapping model as the replies by the web application will be same considering any particular request each time. For example the download link for one file remains same, once that file is uploaded to any web application. The case of images,any audio files such as mp3 and other non-changing data is similar.Check whether for particular download request, query generated is anticipated one or not. If so then that query is authorised to be processed else admin is alerted for malicious activity. Fig. 1 shows deterministic and non-deterministic approach for mapping model.

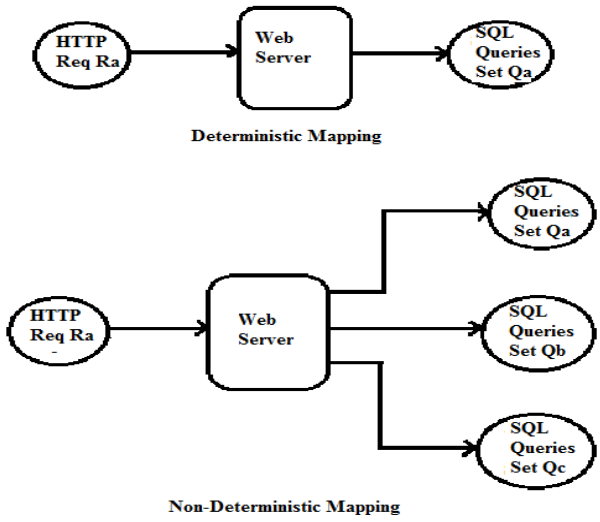

Fig. 1: Mapping Approach

B. Dynamic Mapping model

The Dynamic Web Application follows non-deterministic approach. In Dynamic Web Application, applying mapping model is somewhat difficult, as we cannot anticipate or store the exact input request and subsequent queries for each user. The DWA mapping model is a mixture of deterministic as well as non-deterministic generation of query sets.The dynamic request is checked whether it is prior processed one if so then request is executed which saves a lot of computation time. If not then input data given by user is removed from SQL query and if any malicious content is found then it is removed and query set is generated. This generated query set is checked with permissible query set and if it matches request is allowed to go through else notification is given to admin and that user is blocked.

V. Implementation

The hardware and software requirements of the system are as follows:

- OS: Windows 7 RAM: 2GB Processor: I3
- Visual studio 4.0
- Microsoft IIS Server 7.0
- Performance monitoring tool: http_load and Autobench

The client accesses the web application via a web server. Each client will be given a web service separately. The client will send a web request using the UI of web application. Database query will be generated and the server will respond back to the particular client based on the web request.

The attack on the system will always be in the form of web request. It may contain complicated tags like <script>, <iframe>etc. that will be targeted to attack the client's web browser as a form of XSS attack or a well-structured SQL query that will attack over the database of the web application forming a SQL injection attack. The system is differentiated based on request and query mapping model.

- Two web applications: static web application and dynamic web application are developed using ASP.NET and C#. The static web application will have file upload and download operation while the dynamic web application will be a blog with user account, adding articles, commenting on an article etc features. The static web application will have a deterministic mapping model of request and query and dynamic application will have non-deterministic mapping model.
- Web application in which the query generated is same for each web request, regardless the time or parameters such applications are defined here as static. For example, once the file is uploaded to any web application the download link for that file remains same. Similar is the case of images,mp3 files, video files and other non-changing data.
- But web applications like blog sites, social networking sites etc where queries generated may change based on time or data being passed through the same web request, such web applications are defined as dynamic web applications.
- The working environment for dynamic web applications as well as static web application is set up in such a way that every time the client accesses a web application, a dedicated web service (container) is provided to that client whether he/she is a valid user or attacker. This dedicated web service implements the mapping model for static and dynamic web applications. As soon as the user leaves the web application by logging out, the web service is returned back to the pool of such web services and can be used by some other client. Because of these damages caused by the attacker is limited to that web service only.
- Once the system is ready, the impact measurement by using separate container (web service) for each user on the web server against vanilla web server is done. For that we have used two tools: http_load and autobench. The tool named http_load gave us the throughput and autobench gave us the response time.
- After that performance of web application is measured using inbuilt functionalities of Visual Studio. The parameters used were average page time, memory, pages per second and processing time.

VI. Results

A. Web Server Performance Measurement

1) http_load: In Fig. 2, the overhead of container-based server against a vanilla webserver is compared. For the http_load evaluation, we used the rate of five concurrent users and under the parameters of 100, 200, and 400 total fetches, we also measured for 3 and 10 seconds of fetches. For example, http_load fetches the URLs as fast as it can 100 times, in the 100-fetches benchmark. Similarly, http_load fetches the URLs as fast as it can during the last 10 seconds, in the 10-seconds benchmark . We picked 5 major URLs of the website and tested them against both servers. The figure shows our experiment result. The most crucial indicator to reflect the throughput performance of the web server is the value of fetches per second in the http_load results.

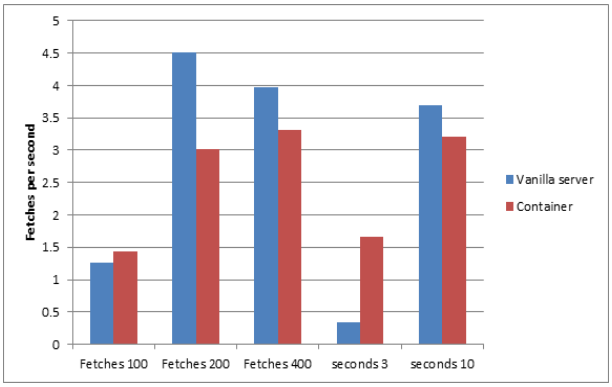

Fig. 2: Performance Evaluation using http_load

From the Fig 2, we can observe that the overhead was less for container based server for 100 fetches but as we increased the fetches to 200 and 400, the overhead increased for container based server. When we put the parameters at 3 seconds, the overhead was less for container based server in comparison with vanilla web server but for 10 seconds, it increased.

2) Autobench: In Fig. 3, we measured the response time of the server using autobench. We tested demanding rate ranging from 10 to 190, which means that a series of tests started at 10 requests per second and increased by 10 requests per second to 190 requests per second were being requested and any responses which took more time than 10 seconds to arrive were counted as errors. The requests rates of both the server and the replay rates for both servers were compared.

Fig. 3 shows that when the rate was less than 150 concurrent sessions per second, requests were fairly handled by both the server. Beyond that, the container-based server showed a drop in the rates.

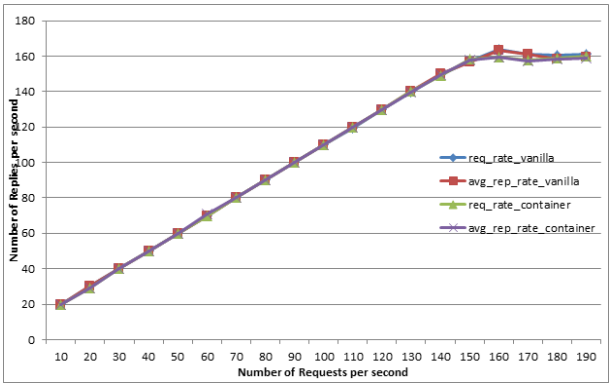

Fig. 3: Performance Evaluation using Autobench

B. Web Application Performance measurement

1) Average page time: The average page time for existing system and proposed system is compared in Fig 4. We have measured the average page time for 50,100 and 200 concurrent users.Fig 4 shows that our approach has better average page time.

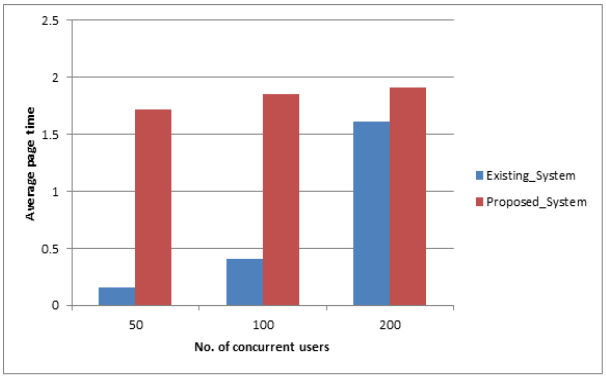

Fig. 4: Average page time

2) Pages per second: The pages per second for for existing system and proposed system is compared in Fig 5. Our approach fetches more number of pages per second than existing approach. The pages per second fetched increases as the number of concurrent users increases.

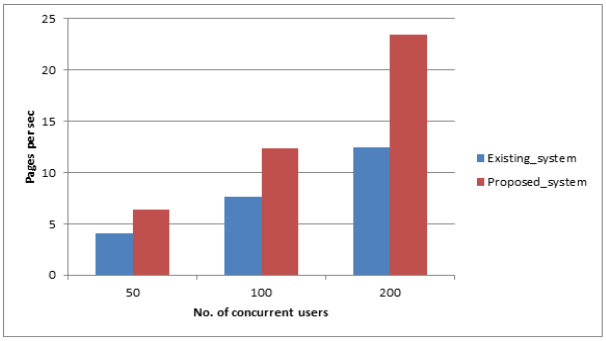

Fig. 5: Pages per second

3) Memory: In Fig. 6 available Mbytes of memory for proposed approach and existing approach is shown. The memory requirement for our approach is slightly more than existing system since we are using container based approach which requires some memory.

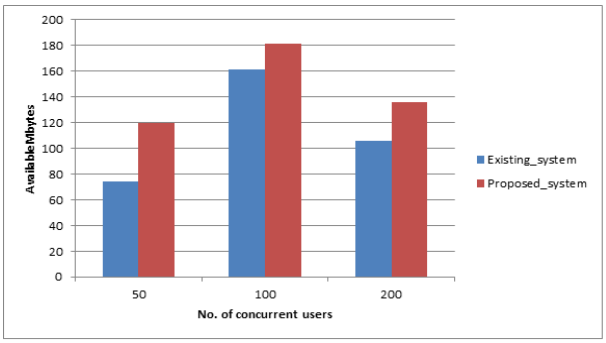

Fig. 6: Memory (Available Mbytes)

4) Processing time: Fig. 7 shows that processing time for our approach is a little bit less than existing approach which means that we have successfully reduced processing time for 50 and 100 concurrent users.

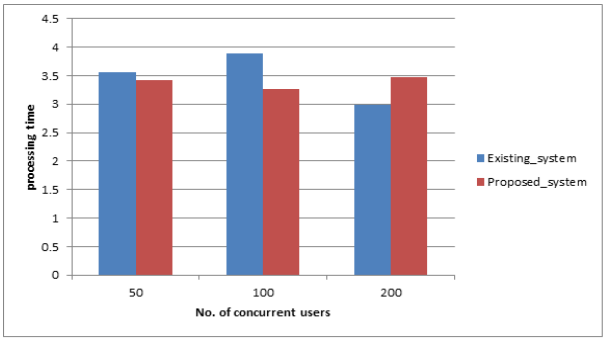

Fig. 7: Processing Time

Beyond that the processing time of our container based approach degrades.

Table 1 shows performance analysis of these various parameters.

	Existing System	Proposed System	Performance (In %)
Average page time	0.72	1.82	152 ↑
Pages per second	8.03	14.05	75 ↑
Memory	113.83	129.5	13 ↑
% processing time	3.48	3.38	2.64 ↓

TABLE I: Performance Analysis

Table 2 shows comparison between existing approach and proposed approach. It shows that our approach detected and prevented more attacks than the existing one.

Research Paper	SQL Injection Attack	Cross Site Scripting Attack	URL Manipulation Attack
[11]	Yes	No	No
[12]	No	Yes	No
[13]	Yes	Yes	No
[15]	Yes	No	No
Proposed System	Yes	Yes	Yes

TABLE II: Comparison of existing and proposed system

VII. Conclusion

In this paper, Intrusion Detection System based on a mapping model is created to detect and prevent SQL Injection and Cross Site Scripting Attack. Each client has its own container which will restrict the damage caused by any attacker to that container only. The impact measurement of container on the web server is done using http_load and autobench and it was found that up to 100 fetches container-based approach has better throughput after that it degrades. Up to 150 concurrent session container based approach runs in level with vanilla web server. The web application performance was measured based on average page time, memory, pages per second and processing time for existing system and proposed one and it was observed that our approach performed better than existing one.

References

[1] A. M. Chandrasekhar, K. Raghuveer, "Intrusion Detection Technique by using K-means, Fuzzy Neural Network and SVM classifiers", *2013 International Conference on Computer and Informatics(ICCCI)*, Coimbatore, INDIA, Jan04-06,2013.

[2] Johari, R.; Sharma, P., "A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection," Communication Systems and Network Technologies (CSNT), 2012 International Conference on , vol., no., pp.453,458, 11-13 May 2012.

[3] Atashzar, H.; Torkaman, A.; Bahrololum, M.; Tadayon, M.H., "A survey on web application vulnerabilities and countermeasures," Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on , vol., no., pp.647,652, Nov. 29 2011-Dec. 1 2011.

[4] Thankachan, A.; Ramakrishnan, R.; Kalaiarasi, M., "A survey and vital analysis of various state of the art solutions for web application security," Information Communication and Embedded Systems (ICICES), 2014 International Conference on , vol., no., pp.1,9, 27-28 Feb. 2014.

[5] Dukes, L.; Xiaohong Yuan; Akowuah, F., "A case study on web application security testing with tools and manual testing," Southeastcon, 2013 Proceedings of IEEE , vol., no., pp.1,6, 4-7 April 2013.

[6] Teodoro, N.; Serrao, C., "Web application security: Improving critical web-based applications quality through in-depth security analysis," Information Society (i-Society), 2011 International Conference on , vol., no., pp.457,462, 27-29 June 2011.

[7] OWASP, *"https://www.owasp.org/index.php/Top_10_2013-Top10"*.

[8] A. Tajpour, M. Massrum, "Comparison of SQL Injection Detection and Prevention Techniques", *2nd International Conference on Education Technology and Computer (ICETC), 2010)*.

[9] Pandurang, R.M.; Karia, D.C., *"Performance measurement of WEP and WPA2 on WLAN using OpenVPN,"* Nascent Technologies in the Engineering Field (ICNTE), 2015 International Conference on , vol., no., pp.1,4, 9-10 Jan. 2015.

[10] http_load, *"http://www.codediesel.com/tools/benchmarking-server-performance-with-http_load/"*.

[11] Autobench, *"http://www.xenoclast.org/autobench/"*.

[12] Meixing Le, Brent ByungHoon Kang, "DoubleGuard: Detecting Intrusions in Multi-tier Web Applications", *IEEE Transaction on Dependable and Secure Computing* , Vol. 9, No. 4, July/August 2012.

[13] Lwin Khin Shar, Hee Beng Kuan Tan, "Automated removal of cross site scripting vulnerabilities in web applications", *Information and Software Technology* ,54,467478, 2012.

[14] R.Priyadarshini, Jagadiswaree.D, Fareedha.A, Janarthanan.M, "A Cross Platform Intrusion Detection System using Inter Server Communication Technique", *IEEE-International Conference on Recent Trends in Information Technology, ICRTIT IEEE MIT*, Anna University, Chennai. June 3-5, 2011.

[15] C. M. Frenz,J. P. Yoon, "XSSmon: A Perl Based IDS for the Detection of Potential XSS Attacks", *Systems, Applications and Technology Conference (LISAT), IEEE*, Long Island, 2012.

[16] Alnabulsi, H.; Islam, M.R.; Mamun, Q., "Detecting SQL injection attacks using SNORT IDS," Computer Science and Engineering (APWC on CSE), 2014 Asia-Pacific World Congress on , vol., no., pp.1,7, 4-5 Nov. 2014.