High Secure Web Service to Resolve Different Web Vulnerabilities

Girisan E K
Assistant Professor, Department of Computer Science, Sree Narayana Guru College, K.G Chavadi, Coimbatore, Tamil Nadu, India

Savitha T
M.Phil Scholar, Department of Computer Science, Sree Narayana Guru College,K.G Chavadi, Coimbatore, Tamil Nadu, India

Abstract – Web applications are one of the most widespread platforms for information and services delivery over Internet today. As they are increasingly used for critical services, web applications become a popular and valuable target for security attacks. Although a large body of techniques has been developed to fortify web applications and mitigate the attacks toward web applications, there is little effort devoted to drawing connections among these techniques and building a big picture of web application security research. In this paper a security scheme is proposed to protect the web application from Cross-Site scripting and injection attacks. The proposal also developed with the aim of regulating the existing techniques after detailed analysis. The system specifically considered four types of attack such as Cross-Site scripting, Code, data and SQL injection attacks. To improve the web application security, the proposed system utilizes the improved honey pot mechanism, Attack Threshold calculation-token generation and web service based authentication mechanisms. The proposed work detects attacks using policy rules and expressions.

Index Terms – Web Security, Cross-Site scripting, SQL Injection Attacks, web service, Authentication.

1. INTRODUCTION

Web application has become one of the most important communication channels between various kinds of service providers and clients .The web application vulnerabilities found in different web applications has increased tremendously in the recent trend[1]. To understand web application security, the threats, the vulnerabilities, and attacks have to be defined. A threat is any potential malicious occurrence that could harm an asset. Vulnerability is a security weakness which makes a threat to spread several types of attacks. This may happen due to poor designing, inappropriate and insecure coding techniques [2][3]. Securing a web application is very difficult. Because some attacks rely on user input that is not validated by the web application perfectly. The authentication attacks, Cross Site Scripting and SQL injection vulnerabilities are in the top list [4]. The earlier work on Cross-site Scripting (XSS) and some sql injection detection need to processed at the server side, and these vulnerabilities cause different types of other attacks in the web applications[5]. So, the web vulnerabilities should be verified priory and effectively. Hence, Web application security is the need of the hour for today's web medium to provide secure and seamless services in an enterprise web application environment. There are many reasons for security flaws to work their way into web applications, such as security is very rarely considered during the functional requirements phase [6].

2. PROBLEM DEFINITION

As technology development increases, equally web vulnerabilities also increases. Many works introduced and provided vulnerability detection mechanisms, while some others provide both detection and prevention mechanisms [7]. However, the application has several issues like performance oriented, more vulnerability are not handled together etc., and Websites affects the cross site scripting, injection attacks which is more vulnerable. Web security system should effectively find, track the vulnerabilities without any performance issues [8]. Several web authentication systems help to track the unauthorized access in the web. However, the high interaction and network resources are need to deploy. Input validation mechanisms are widely used in the earlier work against injection and XSS attacks. Because those attacks occurs due to lack of inspection or insufficient inspection on the input provided by the clients. The two most common input validation based attacks are SQL injection attacks and Cross site scripting attacks [9]. So this server side validation and Attack detection may create server overload problem and which decrease the performance of the server [10].

3. PROPOSED SYSTEM

Web applications are one of the most widespread platforms for information and services delivery over Internet today. In the proposed system a security scheme is proposed to protect the web application from cross site scripting and injection attacks, with the aim of regulating the existing techniques into detailed analysis that promotes future investigation. The proposed system specifically considered multiple attacks such as Cross

Site scripting, Injection, Broken authentication and session management attacks. To improve the web application security, the system has the following contributions.

A. Contributions of the Proposed System

The proposed system developed a high security email server with the three types of research contributions. Using the following techniques and tools, the proposed system overcomes the web application vulnerabilities. To improve the web application security, the system has the following contributions.

- To effectively detect, defend and track multiple types of attacks, a new Security Service Architecture (SSA) is proposed and deployed in a separate service oriented method.
- The SSA includes the Attack Threshold calculation and authorized token generation schemes to verify the client. These verification and authentication schemes are deployed at the web service to reduce the load of the server.
- The proposed work utilizes the Improved Honey Pot (IHP) Mechanism to track the attacker activity; additionally the proposed work performs user input validation to defend the unexpected input, which is more vulnerable for many attacks. This has been performed using security policy and expressions matching.
- The proposed system is explained with various sub sections in this chapter, this initially explains the architecture of the proposed system and then attack scenario is explained with the proposed SSA scheme.

B. Proposed System Architecture

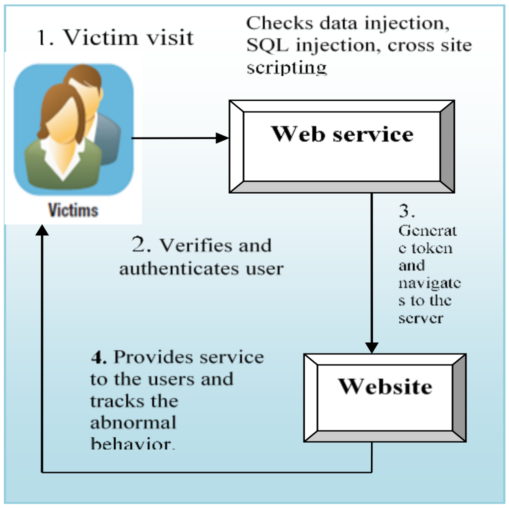

1. Victim visit
Checks data injection, SQL injection, cross site scripting
Victims
Web service
2. Verifies and authenticates user
3. Generate token and navigates to the server
4. Provides service to the users and tracks the abnormal behavior.
Website

Figure 1.0 the overall process involved with the Proposed System

The solution proposed is modularized, configured, and developed in .Net, XML and web services. This approach is evaluated in a web application developed in ASP.net deployed in email server application and is found effective as it provides the flexibility to be used across languages with a very minimal configuration to prevent XSS, injection and password authentication.

The overall architecture of the proposed system is shown in Figure 1.0. The proposed system performed in an email application, which has been created with multi-tier architecture. The proposed system has segmented into several phases such as service initialization with converter, input validation, request passing, user validation, token generation, session initialization, authentication, authentication monitoring and IHP implementation for reporting the malicious behavior.

4. PROPOSED METHODOLOGY

Applications are constantly probed for vulnerability and when found to be vulnerable, are attacked with sustained belligerence. Literature shows that the attacks on web applications are increased, since the attacks are launched on port 8080, which remains open in every situation. Secure Socket Layer (SSL) and firewalls are incompetent against application level attacks as it cannot prevent the port 80 attacks. These attacks can bring down the web application server and can also provide access to the internal databases containing sensitive information like their passwords, phone numbers, financial information's and personal information. Email applications are developed using number of languages and deployed in different operating systems. This is due to the different features that web application provides to its users. If the application is very simple and does not require up time of the server, then this will be developed using HTML. But email like web applications need to consider various interfaces that it need to interact, security and availability of various features in the application.

A) Service Initialization and Validation

The main part of the solution is the application that generates the XML Schema based on the input parameters and constraints at the web application to the web service call. The solution comprises of three major components such as web service call, converter, validates and policy generator application. The converter is the interface between the web application and users. This can be an executable binary or the interface can also be developed in the same language.

B) Converter Section

The converter section is the interface section between the application and the user. The "https" requests are configured to send the requests to the converter that converts the request object to a name value XML pair. Then this XML object is passed on to the validate section. The outcome of the process is the status of the vulnerability of the input that decides the next action for the converter. If the input is found valid, converter passes this input to the web application. Otherwise it throws an exception and takes the programmed action, if the request is found invalid.

C) Validate section

In the proposed system, the XSS and SQL verification are performed by validation sections. Here the input controls are generated by the schema generator and stored in the repository. The schema generator functionality is explained in the following schema generator application section. Validate section receives the XML request object from the converter and retrieves the corresponding schema for the request. The Validate function verifies the input mentioned as the name value pair in the XML object and checks for its vulnerability by mapping the schema constraints. The outcome of the Validate section is sent as the input to the converter section that decides further action for the input.

i. Input Data Form

Input data form gets all input of the web page to generate policy document for that web page. The input parameters of the web page are captured and then it will be passed to the web service

ii. Input Data Element class

Each input control in the web page, which contains the data type, length; input format, special characters allowed and mark up allowed attributes are different. Hence, the regular expressions and the constraints generated by the policy generator for each row are also different. Each row and its associated attributes like data type, length, etc for each input control is represented as an element in the policy language. Hence the input data element class mentioned here is used to generate the elements in a policy document. Once the input is given and done button is clicked in the input data from each row in the data view grid is mapped to an Input Data Element class instance in a loop and this InputDataElement is passed to the Scheme Generator class instance for generation of policy element in a policy document.

iii. Policy Generator

The next step in the proposed system is generating the polices using Policy generator approach is based on regular expression and hence while generating policy, the constraints are generated automatically and included in the policy that is used by the validator function to validate the input for malicious patterns. There are 7 methods included in the CreatePolicyComponentForRootElement(), CreatePolicyComponentForMessageElement (), Save policy, Create Type for String With markup (), CreateBaseTypeFor string (), Remove Spaces (), CreateTypeForNumeric ()

iv. Policy and Schema generation section

The XML schema document is created using the .NET 4.5 System.Xml.Schema namespace sections. This is the core part of the proposed architecture and schema generator application generates the schema for each page based on the input provided by the developer; the generated schema is stored in a file system or in a database. When the Validate receives the XML object for which the XSS vulnerability is to be assessed, it retrieves the corresponding schema of the web page from the repository and validates the input based on the rules stated in the schema. The schema generation process comprises of an input data form, input data element class and a schema generator.

```
Algorithm: SSA
Input: Login details and URL
Output: Container and token ID.
Step 1: Parse the user login Li data and pass to web service Ws.
Step 2: Ws receive user login request Li and moves to a container C.
Step 3: Create token Tk for Li in  C
Step 4: for every Tk in C do verify input and validateif Li reserved SQL Keyword Move Tk to Rejected List RL;
Step 5: Parse the user input  Li data into Tk; While (not empty of token) Check if token • reserved JavaScript keyword Set the flag1 to continue; Else Set the flag1 to deny;
Step 6: Extract the URL from HTTP; Parse the URL into Tk; While (not empty of Tk) Check if (URL = Benign using the signature check) Set the flag to continue; Else Set the flag to deny;
Step 7: Send the Tk, flag1 and original Web request to Web application Server;
Step 8: return Tk and validation report
```

The algorithm initially gets the user requested URL and parse for the validation. Finally returns the generated authentication token and flag value.

The IHP technique is deployed to handle the broken authentication issues, which can be performed using the session hijacking and various types of password authentication vulnerabilities.

The proposed SSA architecture contains the tracking portion, which handles the authentication related issues in the email server application. This navigates the user who tried wrong password more than a specific threshold. The threshold limit can be detected from the user log, which is stored in a web log. Based on that the user's remote screens are captured and that will be sending to the legitimate users.

Algorithm: Improved Honey Pot(IHP)

```
Input: user authentication log, threshold T
Output: determines the type of user (attacker or legitimate) and gives appropriate page and monitors if the user as attacker.
Step 1: Parse the user login Li data and pass to web service Ws. And perform SAA
Step2: if the Li and flag value is 0, then add the count to the Failure Count (FC).
Step3: if ( FC>T) then
Step4: move to the fake page Fp
Step 5: validate and provide appropriate page Vp. Session generation and signature processing for the selected token and session.
Step 6: update token and session
Step 7: end
Dynamic fake session creation for the attackers will be performed by the SSA architecture, and that will detect the attacker activity passively.
```

5. IMPLEMENTATION AND RESULTS

The experiments are performed on an Intel Dual Core with a RAM capacity 2GB. The algorithms are implemented in ASP.Net for email server site creation and C#.NET as coding language and are run under Windows family. The system has successfully implemented using Visual studio.Net and C#.net as code program. The security architecture of the proposed system is carried out on the email server application, which needs more security in the current scenario.

Table1.0Description about implementation parameters

Attacks	Count	Description
Total XSS attack samples used	10	The system collects 10 user information's along with their fake XSS related scripts.
Total SQL samples	5	The proposed system gathered 5 sql queries to evaluate the proposed system performance.

The implementation was performed by the real world XSS, injection and authentication based attack datasets. Due to the security and insufficient dataset of real email server website a new website has been created for the evaluation. And then the system constructed a new scheme to perform the SSA process. The followings are the details about the implementation process and parameters. The proposed system experimented with several numbers of users to evaluate the proposed system outcome. All the above parameters have been created from the email server website, which is developed for the implementation. The system implements the session feature type and its description, based on this features the system will authenticate users. This also restricts user if any feature mismatches.

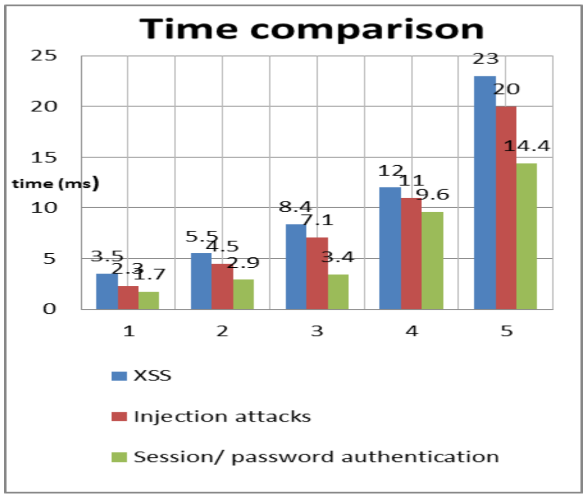
5.1 Performance of proposed system

In this section measure the performance of the each iteration then measure the results the overall time of the SSA. Policy matching accuracy is evaluated by comparing with different set of policies. The table 2.0 shows the policy extraction time at the time of different inputs and different attack policies

samples	XSS	Injection attacks	Session/ password authentication
1	3.5	2.3	1.7
2	5.5	4.5	2.9
3	8.4	7.1	3.4
4	12	11	9.6
5	23	20	14.4

Table 2.0: Performance Measurement

In this section, through statistic analysis, the system finds the average time taken for each process in SSA.



Figur 2.0 Time Comparison to Perform Different Security Process

This initially calculates total time consumption for XSS verification, SQL injection validation and session generation. This shows the impact of the new security system, which needs less time. The accuracy of the proposed model is analyzed with the accuracy measures. From the above figure 2.0 the time consumption at each stage of SSA is compared. The user list is specified from 1 to 5. For each process, the time calculated and finally the system provides the overall time taken by the SSA. This shows for 5 users the system takes 57.4 mille second time.

6. CONCLUSION

A new improved SSA architecture based on web service model for email server application proposed takes care of the all possible risks of hacking. Redefining the proposed security architecture and including Password log history in attack detection have reduced the risks encountered by the earlier authorization techniques. And in case of any attacker tries to login, then the fake page will be navigated. The model is not prone to contour analysis since parameter analysis takes place in a secured internal environment of mail server application. The proposed authorization model will save the email server application from the hands of hackers.

The Proposed system developed a security service Architecture to prevent SQL Injection, XSS vulnerabilities and authentication attacks for the web applications. This has deployed a web service to reduce the load of the server. The proposed system achieved high accuracy in attack detection and prevented the application from various attacks. The proposed system not only prevents the attack, it also tracks the authentication failures and other vulnerabilities using the low interactive honey pot mechanisms.

REFERENCES

[1] Shrivastava, Ankit, SantoshChoudhary, and Ashish Kumar. "XSS vulnerability assessment and prevention in web application." Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on. IEEE, 2016.

[2] Sonewar, Piyush A., and Nalini A. Mhetre. "A novel approach for web application threats, the vulnerabilities security." Pervasive Computing (ICPC), 2015 International Conference on. IEEE, 2015.

[3] Gupta, Aditi, JavidHabibi, Michael S. Kirkpatrick, and Elisa Bertino. " Exploitation of cross-site scripting (XSS) vulnerability on real world web applications and its defense." IEEE Transactions on Dependable and Secure Computing 12, no. 3 (2015): 326-337.

[4] Halfond, William G., Jeremy Viegas, and Alessandro Orso. "A classification of SQL-injection attacks and countermeasures." Proceedings of the IEEE International Symposium on Secure Software Engineering. Vol. 1. IEEE, 2006.

[5] Sonewar, Piyush A., and Sonali D. Thosar. "Detection of SQL injection and XSS attacks in three tier web applications." Computing Communication Control and automation (ICCUBEA), 2016 International Conference on. IEEE, 2016.

[6] Narayanan, Sandeep Nair, AlwynRoshanPais, and Radhesh Mohandas. "Detection and Prevention of security threats In Web application using Semantic Equivalence." Computer Networks and Intelligent Computing. Springer, Berlin, Heidelberg, 2011. 103-112.

[7] Sonewar, Piyush A., and Sonali D. Thosar. "Detection of SQL injection and XSS attacks in three tier web applications." Computing Communication Control and automation (ICCUBEA), 2016 International Conference on. IEEE, 2016.

[8] Huang, Yao-Wen, et al. "Securing web application code by static analysis and runtime protection." Proceedings of the 13th international conference on World Wide Web. ACM, 2004.

[9] Sonewar, Piyush A., and Nalini A. Mhetre. "A novel approach for detection of SQL injection and cross site scripting attacks." Pervasive Computing (ICPC), 2015 International Conference on. IEEE, 2015.

[10] Mobasher, Bamshad, et al. "Effective server overload problem for shilling item-based collaborative filtering systems." Proceedings of the 2005 WebKDDWorkshop, held in conjuction with ACM SIGKDD. Vol. 2005. 2005.