**IOSR Journal of Engineering (IOSRJEN)** **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 22-25

# Securing Web Applications Against Cross Sitescripting, Injection Attacks And Authentication Attacks

M.Newlin Rajkumar, Susithra. D, Preetha Rexlin P.L, Vickma. S
*(Master Of Engineering in Computer Science, Anna University Regional Campus, 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 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 4 types of attack such as Cross Site scripting, Code, data and SQL injection attacks. In addition, the existing research works on the three attack categories. 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 traffic detects attacks using policy rules and expressions*

***Keywords****: Script injection attacks, process site scripting, web application attacks.*

## I. Introduction

Web application has become one of the most important communication channels between various kinds of service providers and clients. It has evolved from a static medium, with user interaction limited to navigation between web pages, to a highly interactive dynamic medium performing concurrent transactions and serving up personalized content. This dynamic medium application offers a wide range of services, such as on-line stores, e-commerce, social network services, etc. As more and more services are provided via the World Wide Web, efforts from both academia and industry are striving to create technologies and standards that meet the sophisticated requirements of today's enterprise Web applications and users. A web application can be characterized by the number of layers that information will pass through on its journey from the data tier (where it is stored in a database typically) to the presentation tier (where it is displayed to the client). Each layer generally runs on a different system or in a different process space on the same system. Client and server are separated by logics. This 2-tier client-server scenario works very well for a small business that only uses, or needs, a single data source. However, the goal of most businesses is to grow, and this needs additional logics to meet the business demands. Unfortunately, the 2-tier approach does not scale very well. If the business rules change then the application needs to be rebuilt and redeployed. Due to the limitations of the 2-tier client-server architecture, distributed applications are often divided into three or more tiers Components in each of these perform a specific type of process.

### 1.1 Web Functionality

Web applications employ numerous different technologies to deliver their functionality. Reasonable functional application may employ dozens of distinct technologies with in its server and client components.

#### 1.1.1 Server-Side Functionality

Whenever a user requests a resource, the server's response is created on the fly, and each user may receive content that is uniquely customized for him. When a user's browser makes a request for a dynamic resource, it does not normally ask for a copy of that resource. In general, it will also submit various parameters along with its request. These parameters enable the server-side application to generate content that is tailored to the individual user. There are three main ways in which HTTP requests can be used to send parameters to the application, namely, the URL query string, HTTP cookies and the body of requests using the GET / POST method.

#### 1.1.2 Client-side functionality

In order to make the server-side application receive user input and actions, and present the results of these back to the user, it needs to provide a client-side user interface. Since all web applications are accessed via the web browser, all these interfaces share a common core of technologies. HTML forms are the usual mechanism for allowing users to enter arbitrary input via the browser. Hyperlinks and form elements can be used to create a rich user interface capable of easily gathering most kinds of input which web applications require. However, most applications employ a more distributed model, in which the client side is used not simply to submit the user data and actions but also to perform the actual processing of data. A significant development in the use of JavaScript has been the appearance of Asynchronous JavaScript and XML (AJAX) techniques for creating a smoother user experience, which is closer to that provided by the traditional desktop applications (Zepeda and Chapa 2007). AJAX involves issuing dynamic HTTP requests from within the HTML page, to exchange data with the server and update the current web page accordingly, without loading a new page altogether. These techniques can provide very rich and satisfying user interfaces.

**1.2 Sql Injection**

Web applications in the World Wide Web (WWW) have user interfaces where a user can input data to interact with the application's underlying relational database management system. This input becomes a part of an SQL statement, which is then executed on the RDBMS. A code injection attack that exploits the vulnerabilities of these interfaces is called an ''SQL Injection Attack'' (SQLIA) (CERT 2007; Su and Wassermann 2006; Howard and LeBlanc 2003; Viega and McGraw 2001).Through SQL injection attacks, an attacker may extract undisclosed data, bypass authentication, escalate privileges, modify the content of the database, execute a denial-of-service attack, or execute remote commands to transfer and install software.SQL Injection could be possible through redirecting and reshaping the query, error based query, injection through stored procedure and blind SQL injection.

**1.3 Cross-Site Scripting Attack**

Web developers may use the Client Script, in particular, the JavaScript to enhance their web applications. If the attacker is able to embed a script in a web page that a valid user requests from sites, he can take control of the entire context. Such a maliciously embedded script is called a Cross-Site Scripting (XSS) attack (Kerschbaum 2007), since the script actually originated from a different site. In order to mount this attack, the attacker exploits the vulnerable web applications. Interactive web applications require a user input. If this input is used by the web application to build the response web page, it might be exploited by the attacker. In particular, if the inputs are not sanitized well enough by the application, then they can be exploited. This script originating in the input part of the URL of the request would be executed in the attacked user's security context.The consequences can be severe like Cookie (and session) theft, Browser hijacking, including but not limited to, User monitoring and data theft, and Request forgery and fake transactions.

**1.4 Broken Authentication Attack**

Session hijacking takes effect directly within the web application into which it was injected by the XSS exploit. Therefore, the attacker possesses the same control over the application as the attacked user. By creating HTTP requests to the exploited application, the attacker is able to execute actions on the application using the victim's current authentication state.From the application's point of view, all actions by the attacker executed through a session hijacking attack are indistinguishable from legitimate actions of the attack's victim (i.e., by the authenticated user which accesses the exploited application). Thus, a session hijacking attack empowers the attacker to temporarily overtake the victim's identity in respect of the exploited application. Session hijacking attacks may either require real- time interaction by the attacker, or be fully pre-scripted for automatic execution. The latter case is, for instance, used by XSS worms (Livshits and Lam 2008). All currently known XSS session hijacking attack methods can be assigned to one of the following different classes "Session fixation", "browser hijacking" and "background XSS propagation".

**1.4 The Common Language Runtime (Clr):**

The common language runtime is the foundation of the DOTNET Framework. It manages code at execution time, providing important services such as memory management, thread management, and removing and also ensures more security and robustness. The concept of code management is a fundamental principle of the runtime. Code that target the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code.

**1.5 The Dot Net Frame Work Class Library:**

It is a comprehensive; object-oriented collection of reusable types used to develop applications ranging from traditional command-line or graphical user interface (GUI) applications based on the latest innovations provided by ASPDOT NET, such as Web Forms and XML Webs services.The Dot Net Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features.

**1.6 Features Of The Common Language Runtime:**

The common language runtime manages memory; thread execution, code execution, code safety verification, compilation, and other system services these are all run on CLR.

- Security.
- Robustness.
- Performance.

**1.7 About Asp Dotnet**

ASPDOT NET is the next version of Active Server Pages (ASP); it is a unified Web development platform that provides the services necessary for developers to build enterprise-class Webs applications. While ASPDOT NET is largely syntax compatible, it also provides a new programming model and infrastructure for more secure, scalable, and stable applications. ASPDOT NET is a compiled, NET-based environment; user can author applications in any DOT NET compatible language, including Visual Basic DOT NET, C#, and JScript DOT NET.Additionally, the entire DOT NET Framework is available to any ASPDOT NET application. Developers can easily access the benefits of these technologies, which include the managed common language runtime environment (CLR), type safety, inheritance, and so on. ASPDOT NET has been designed to work.

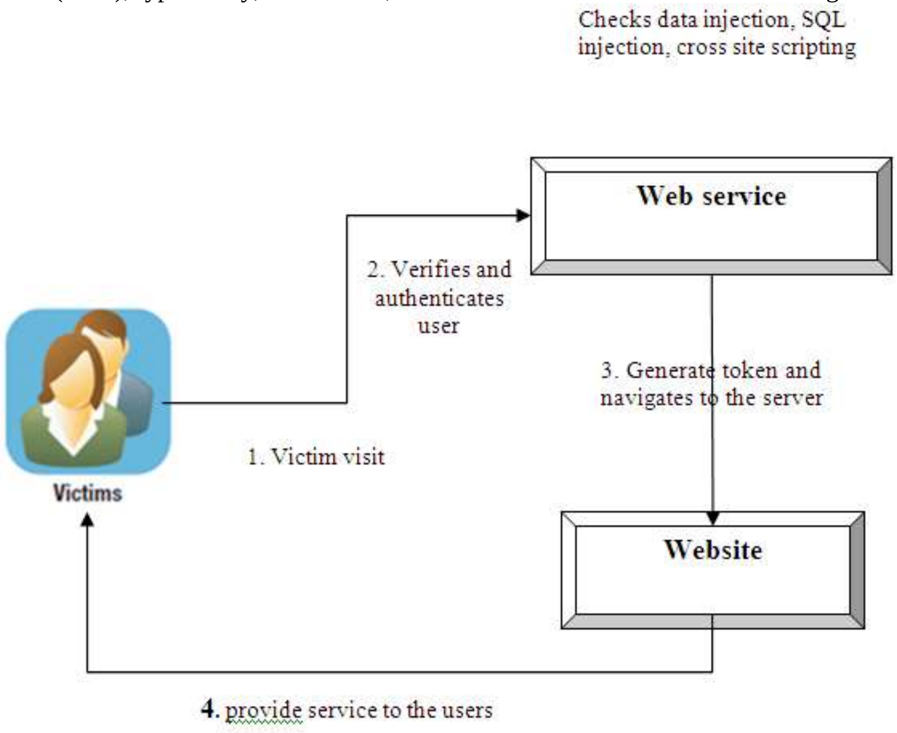
Checks data injection, SQL injection, cross site scripting
Web service
2. Verifies and authenticates user
3. Generate token and navigates to the server
Victims
1. Victim visit
Website
4. provide service to the users

**Figure 1.1**

| Attacks | Count | Description |
|---|---|---|
| Total XSS attack samples used | 2 | The system collects 10 user information along with their fake XSS related scripts. |
| Total SQL samples | 5 | The proposed system gathered 5 more sql queries to evaluate the proposed system performance. |

**Table 1.1**

## II. Conclusion

A new improved Security architecture based on web service model for email server application proposed takes care of the all possible risks of hacking. Redefining the location trust metric and including Password log history in trust metric over comes the risks encountered by the earlier trust metrics based 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 hackersntly used data, or customizes your application's configuration, to name only a few possibilities. ASPDOT NET provides a simple model that enables Webs developers to write logic that runs at the application level. Developers can write this code in the global.aspx text file or in a compiled class deployed as an assembly.

This logic can include application-level events, but developers can easily extend this model to suit the needs of their Web application. ASPDOT NET provides easy-to-use application and session-state facilities that are familiar to ASP developers and are readily compatible with all other DOT NET Framework APIs. ASPDOT NET offers the IHttp Handler and Http Module interfaces. ASPDOT NET takes advantage of performance enhancements found in the DOT NET Framework and common language runtime. Additionally, it has been designed to offer significant performance improvements over ASP and other Web development platforms. All ASPDOT NET code is compiled, rather than interpreted, which allows early binding, strong typing, and just-in-time (JIT) compilation to native code, to name only a few of its benefits. ASPDOT NET is also easily factorable, meaning that developers can remove modules (a session module, for instance) that are not relevant to the application they are developing. Web Forms allows us to build powerful forms-based Web pages. When building these pages that use ASPDOT NET server controls to create common UI elements, and program them for common tasks. These controls allow to rapidly building a Web Form out of reusable built-in or custom components, simplifying the code of a page. An XML Web service provides the means to access server functionality remotely. Using Web services, businesses can expose programmatic interfaces to their data or business logic, which in turn can be obtained and manipulated by client and server applications.

## Acknowledgement

I would like to express my gratitude to my guide Dr.M.NEWLIN RAJKUMAR.,M.E.,Ph.D, for guiding me properly in my project work and for helping to solve the project work difficulties. I would also like to thanks all the staff members of computer science and engineering department for supporting me and guiding me in my project work whenever required.

## References

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

[2]. Gupta, Aditi, Javid Habibi, Michael S. Kirkpatrick, and Elisa Bertino. "Marlin: Mitigating code reuse attacks using code randomization." *IEEE Transactions on Dependable and Secure Computing* 12, no. 3 (2015): 326-337.

[3]. Yang, Xinyu, Jie Lin, Wei Yu, Paul-Marie Moulema, Xinwen Fu, and Wei Zhao. "A novel en-route filtering scheme against false data injection attacks in cyber-physical networked systems." *IEEE Transactions on Computers* 64, no. 1 (2015): 4-18.

[4]. Miao, Fei, Quanyan Zhu, Miroslav Pajic, and George J. Pappas. "Coding schemes for securing cyber-physical systems against stealthy data injection attacks." *IEEE Transactions on Control of Network Systems* 4, no. 1 (2017): 106-117.

[5]. 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.

[6]. 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.

[7]. Majorczyk, Frédéric, and Jonathan-Christofer Demay. "Automated Instruction-Set Randomization for Web Applications in Diversified Redundant Systems." Availability, Reliability and Security, 2009. ARES'09. International Conference on. IEEE, 2009.

[8]. Narayanan, Sandeep Nair, Alwyn Roshan Pais, and Radhesh Mohandas. "Detection and Prevention of SQL Injection Attacks using Semantic Equivalence." Computer Networks and Intelligent Computing. Springer, Berlin, Heidelberg, 2011. 103-112.

[9]. 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.