International Journal of Advanced Research in

Computer Science Engineering and Information Technology

Volume: 4 Issue: 1 19-Dec-2014,ISSN_NO: 2321-3337

A SURVEY ON ROUTINE DETECTION OF WEB APPLICATION DEFENCE FLAWS

[1] M.S.THARA DEVI, [2]S.SELVANAYAKI

[1, 2] Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering College, Chennai, Tamil Nadu, India.

ABSTRACT− *The security of web applications becomes a major concern and it is receiving more and more attention from governments, corporation and research community. As the web application play a preponderant role, one can realize the importance of finding ways to reduce the number of vulnerabilities. The main research goal is to understand the typical software faults that are behind the majority of web application vulnerabilities. The focus is mainly on the two most critical web applications, which are SQL Injection and XSS and the goal is to identify leaks and recover them with help of algorithms. The main aim of the project is to propose an algorithm to detect the security vulnerabilities that performs a scanning process for all website/application files. These train software developers and code injectors that can be used to assess security mechanisms such as intrusion detection system, vulnerability scanners and static code analyser's a result, the number of reported attacks that exploit web application vulnerabilities are still increasing.*

Key terms: XSS, SQL Injection, Web application vulnerabilities.

1. Introduction

New threats emerge every day. Some hackers are not satisfied with penetrating your network, they seek information that resides in your applications or databases. Attackers also followed the move to the web and as such more than half of current computer security threats and vulnerabilities affect web applications [10]. Not surprisingly, the number of reported attacks that exploit web application vulnerabilities is increasing. Given the preponderant role of web applications in many organizations, one can realize the importance of finding ways to reduce the number of vulnerabilities. Applications are often plagued by poor designs, software bugs and poor programming practices.

Applications contain and process your most critical information. Web application and web users are targets of many attacks like Cross Site Scripting, SQL Injection, Cross Site Request Forgery etc., Generally Web application security is "The Securing of Web Applications". Regarding the programming language perspective, we considered some of the most relevant in the context of web applications.

First, we focused on the most widely used weak typed language, PHP. Then, we analyzed strong typed languages, namely Java, C#, and VB [1]. Recall that our goal is not to analyze each programming language in what concerns their ability to prevent security vulnerabilities, but to analyze the vulnerabilities and their relation with some

language characteristics, like the type system.[10] Most information systems and business applications built nowadays have a web front end and they need to be universally available to clients, employees, and partners around the world, as the digital economy is becoming more and more prevalent in the global economy.

These web applications, which can be accessed from anywhere, become so widely exposed that any existing security vulnerability will most probably be uncovered and exploited by hackers. The vulnerabilities analyzed in our field study were cross-site scripting (XSS) and SQL injection (SQLi), as these are two of the most common and critical vulnerabilities found in web applications.

To characterize the types of software faults that are most likely to create these vulnerabilities, we classified the pieces of code used to fix them in a set of real web applications. Each patch was inspected in depth to gather the precise characteristics of the code that was responsible for the security problem and classified them according to an adaptation of the orthogonal defect classification (ODC) .

Many programming languages are currently used to develop web applications. Ranging from proprietary languages (e.g., C#, VB) to open source languages (e.g., PHP, CGI, Perl, Java), the spectrum of languages available for web development is immense.
Programming languages can be classified using taxonomies, such as the programming paradigm, the type system, the execution mode, and so on.

The type system, particularly important in the context of the present work, specifies how data types and data structures are managed and constructed by the language, namely how the language maps values and expressions into types, how it manipulates these types, and how these types correlate. Regarding the type system, they can be typed versus untyped, static versus dynamic typed, and weak versus strong typed [10]. In particular, strong typed languages provide the means to produce more robust software, since a value of one type cannot be treated as another type (e.g., a string cannot be treated as a number), as in weak typed languages.

1.1 Objective

The objective of this paper is to identify one of the vulnerability and recover them automatically.

1.2 Scope

The scope of the paper is to identify the data leaks then and there and produce suggestions on specific attacks.

2. System Analysis

System Analysis is a combined process dissecting the system responsibilities that are based on the problem domain characteristics and user requirements.

2.1 Existing System

Two of the most critical web application vulnerabilities analysed are Cross Site Scripting and SQL Injection [6]. I f input controls are not validated against the malicious input or script that destroy the database or steal website files. In the existing system, they didn't focus on the programming language perspective [10].They classified pieces of code which is used to fix them in a set of real web application, where each patch was inspected in depth to gather precise characteristics of code, which was responsible for security problems. Hence they observed a single fault type MFCE, Missing Function Call Extended which is a new addition based on a missing function in situations where the return value is used in the code for most of the security problems analysed. According to the language perspective, they show that the number of vulnerabilities analysed using specific language is not a guarantee of success in preventing vulnerabilities and it is just one of many factors that contribute to the building of safer application.

Disadvantages:

- This system is exposed to attack because of trusting client side data.
- Lack of re-authenticating the user before issuing new password or performing critical attacks.

2.2 Proposed System

In the proposed system, leaks in data are identified and suggestions for recovery is given for two of the most widespread vulnerabilities SQL Injection and Cross Site Scripting [4]. These leaks are identified by using specific syntax and rules. Another added up advantage in this paper is the SQL Injection which is identified is being recovered automatically. This is done with the help of Prepared Replaced Statement and Symbolic Execution algorithms.

Advantages:

- Preprocessing input from user before echoing it.
- Leaks are identified then and there.
- Automatic identification and recovery of one vulnerability.

3. System design

3.1 System Architecture

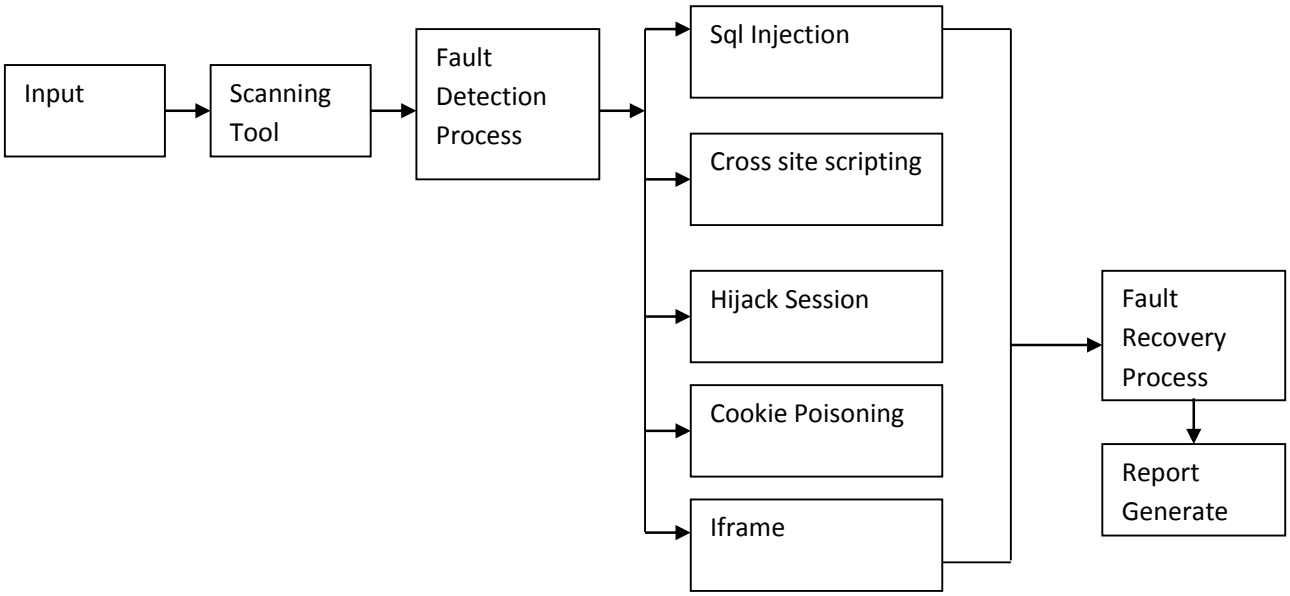

Fig:Architecture Diagram

Here the input is given by the user where a scanning tool is developed, which does fault detection process. This process identifies vulnerabilities like SQL Injection and Cross Site Scripting, which leads attackers to hijack session, cookie poisoning etc. Then occurs fault recovery process and finally reports get generated.

4. Literature Survey

1. G. _ Alvarez and S. Petrovic , suggests that**,** a number of taxonomies of computer attacks and vulnerabilities have appeared in recent years. The shortcoming of taxonomies is that they often include categories for classification of web attacks, so they fail to cover all subtleties of web attacks. This paper proposed a taxonomy of web attacks taking into account a number of important features of each attack category .this represents an effort to cover known attacks and also some attacks that might appear in future. Finally possible applications are described such as intrusion detection system & application firewalls.
2. S. Christey, explains that **,** this paper's future work is intended to spark discussion in the research and consumer communities and to encourage the development of metrics for software assurance. This paper highlights the most common of the unforgivable vulnerability, followed by a proposal that the research community establish a set of VAAL ,that can be used as indicators for relative security of software products. Vulnerability Assessment Assurance Level is originally proposed by David Litchfield as a useful way of communicating the extent to which security analysis has been performed on a product which can also be used as guideline for conducting repeatable analysis of various depths.

3. J. Christmansson and R. Chillarege, ,says that **,** in the past decade, fault injection has emerged as an attractive approach to the validation of dependable systems. Generation of error set ie.,what model, where to inject , when to detect and how to design operational file.In this paper the proposed methodology represents, the input data to the first part are chosen such that they satisfy injection condition.consequently,time is not wasted waiting for an error injection and system usage after injection will be representation of field usage.

4. W. Halfond, J. Viegas **,** focusses that , the developer have proposed a range of coding guidelines that promote defensive coding practices such as encoding user input and validation.they insisted that the future work should focus on evaluating the techniques precision and effectiveness in practice. Empirical evaluation presented in "How do I prevent Injection " would allow for comparing the performance of different techniques when they are subjected to real world attacks and legitimate input.

5. N. Jovanovic, C. Kruegel , discusses **,** that the number and the importance of web applications have increased rapidly over the last years. At the same time the quantity and impact of security vulnerability in such application have grown as well. This paper enhanced the previous work by integrating a novel alias analysis using shadow variables. This new approach is able to generate precise solutions even for difficult aliasing problem. Moreover they presented an iterative ,two stage processing step for the automatic resolution of file inclusions.

6. B. Livshits and S. Lam**,** says that, this paper proposes a static analysis technique for detecting many recently discovered application vulnerability such as SQL injection,xss & HTTP splitting attacks. This paper also proposes a tool based on static analysis for finding vulnerability caused by unchecked input. They showed how a general classification of security errors in java applications can be formulated. Still this approach has not found any techniques to recover this most critical vulnerabilities.

7. Katkar anjali S.,Kulkarni raj B, have been focussing on Web application that consist of several different interacting technologies. These interactions between different technologies can cause vast security problems. The main objective is to find out whether these security mechanisms provide significant help to the builder of a web application. They also presented that the future work can be extended by demonstrating the other various vulnerability like sql injection,xss,buffer overflow,denial of service & providing solution for these.

8. Pankaj Sharma, Rahul Johari , have been focussing on that, almost all organizations have improved their performance through allowing more information exchange within their organization as well as between their distributers, suppliers and customers using web support. In this paper the review finds that the existing techniques suffer from some weaknesses like inherent limitation, incomplete implementation & complex framework. Keeping in view, the emerging web technologies & extensive usage of highly interactive content over internet.

.

5. XSS and SQL Injection Analysis

XSS: is a web security vulnerability by which a malicious user is able to inject content into a webpage that will impact other website visitors. The injected content could then perform any number of unwarranted acts including hijacking user session, stealing private information, performing DOS attacks**.** [10] **SQL Injection**: inserting user supplied SQL statements into dynamically generated SQL query making unintended use possible or use parameterized stored procedure to prevent. Main focus is on exploit, which is a code developed by hackers to attack a specific vulnerability of target system or web application. Analysis is based on the data collected from the database of exploits found from several sites. For eg Milworm site is a hacker-related site devoted to share exploits of vulnerabilities developed by several Hacking groups and individuals.

5.1 Stored cross site scripting

The stored (or persistent) Cross Site Scripting occurs when the data provided by the attacker is saved by the server, and then displayed permanently on "normal" pages returned to other users. Stored XSS requires particular kind of vulnerability in the application where the data is placed in somewhere (ex. Data base) and later feedback is send to the user, this can be through Forum, Blog, etc. [10] The attacker can send <HTML> or <JavaScript> to the application instead of the normal input to be stored in the data base, later when the victim comes to the application web site he/she will download the
<HTML> or <JavaScript> [1] located there. The application here acts as an attack site but Works for the hacker.

5.2 Reflected Cross Site Scripting

Reflected (or non-persistent) Cross Site Scripting can occur when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions[5], is used immediately by server-side scripts to generate a page of results and reflected back for the user, without sanitizing the request. For example, if we have a user Log-In prompt (User-Id, Password) and the user has supplied his Log-In information. Suppose the user typed his Password incorrectly, he may have a message like ("Sorry, Invalid Log-In"), but sometimes we have a message like ("Sorry Ahmed, Invalid Log-In") and here's the problem, where the (user name) sends and reflected back to the output. If there is no input validation for Log-In text boxes, the attacker can exploit this vulnerability to inject his malicious input 'XSS' instead of User-Id [2]. The attacker can craft an email contains a link request from the user to click on the link to update personal data.

5.3 Hijack Session

The attacker needs the cookie form the victim to hijack the session. This is can be implemented by creating one form and make it submit to the attacker site [10].

Example:

</form><form name = 'a' action = 'attackersiteaddress'method = 'post'>
<input type = hidden value = '<script> + document.cookie +
</script>'>
</form>
<script> a.submit() </script>

5.4 Cookie Poisoning

The attacker can corrupt the value of the cookie if he detect that an application is relying on the cookie value to display specific action done by the user with "response. Write"[4]. Assume the application store the value of the last search done by the user along with the date-time in cookies.

Example:

The attacker here can update the value of the last search with a herf pointing to his site as following:
<script> document.cookie.userlastsearch = '<A herf ="attackersiteAddress"> You have won a random prize please click here to continue </A>'</script>
There he may ask the user to Log-In again to fool him with a prize. The attacker may bait him with 5$ and latter ask him to pay 50$ for some wonderful product. The amount may not the main target but the credit card number.

5.5 IFrame

The <iframe> tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document [6]. The attacker can simply fool the user by showing the UI that has size 100% in height and width to look the same as an application site through writing the following malicious code:
<iframe SRC="attacker site" height = “100%” width =”100%”>
From the previous examples, we reach to the main reasons that make an application susceptible to Cross Site Scripting attacks:
1. There is no input validation control for the inputs coming to an application.

2. There is no sanitization control for the output coming from the application.

5.6 Fault Detection

The suggested algorithm performs a scanning process for all website/ application files. Our scanner tool relies on studying the source code of the application depending on ASP.NET files and the code behind files (Visual Basic VB and C sharp C#)[1].To detect the security vulnerabilities and leaks. [5]Therefore, the scanner tool tries to detect the vulnerabilities that can help hackers from the reflected output or messages, check most of the ASP.NET server controls and the commands in the code behind that interact with the database.

5.7 Fault Recovery

After scanning process, it will generate a report list all the discovered leaks and vulnerabilities by displaying the name of the infected file, the description and its location [5]. The suggested algorithm will help organization to fix the vulnerabilities and improve the overall security. This report requires a reaction from the organization to take the necessary corrections steps

.

6. Conclusion

According to our findings from survey papers we have found many techniques in which vulnerabilities can be identified. But this paper has an advantage of identifying vulnerabilities and recovering one of the vulnerability, SQL Injection automatically once occurred. Some of the other advantages are pre-processing input from user before echoing it, leaks are identified then and there

References

[1] G. _ Alvarez and S. Petrovic, “A New Taxonomy of Web Attacks Suitable for Efficient Encoding,” Computers and Security, vol. 22, no. 5, pp. 435-449, July 2003.
[2] S. Christey, “Unforgivable Vulnerabilities,” Proc. Black Hat Briefings, 2007.
[3] J. Christmansson and R. Chillarege, “Generation of an Error Set That Emulates Software Faults,” Proc. IEEE Fault Tolerant Computing Symp., pp. 304-313, 1996.
[4] W. Halfond, J. Viegas, and A. Orso, “A Classification of SQL Injection Attacks and Countermeasures,” Proc. Black Hat Briefings, 2005.
[5] N. Jovanovic, C. Kruegel, and E. Kirda, “Precise Alias Analysis for Static Detection of Web Application Vulnerabilities,” Proc. IEEE Symp. Security and Privacy, pp. 27-36, 2006.
[6] B. Livshits and S. Lam, “Finding Security Vulnerabilities in Java Applications with Static Analysis,” Proc. USENIX Security Symp., pp. 18-18, 2005.

[7] Katkar anjali S.,Kulkarni raj B.,"Web vulnerability detection and security mechanism",sep 2012.

[8] Pankaj Sharma, Rahul Johari, "A survey on web application vulnerabilities (SQLIA,XSS) exploitation and security engine for sql injection, 2012 International Conference.

[9] OWASP Foundation, "OWASP Top 10," https://www.owasp.org/index.php/Top_10_2010-Main, July 2010.