ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

WEB SECURITY VULNERABILITY ASSESSMENT AND RECOVERY MACHANISAM

M.Madhusudhanan*, M.Saravanan, D.Durai kumar
* M.Tech-Information Technology, GanadipathyTulsi's Jain Engineering College, Vellore, Tamilnadu, India.
Associate Professor, Dept. of Information Technology, GanadipathyTulsi's Jain Engineering College, Vellore, Tamilnadu, India.
Associate Professor, Dept. of Information Technology, GanadipathyTulsi's Jain Engineering College, Vellore, Tamilnadu, India.

ABSTRACT

Nowadays web applications have critical logical holes (bug) affecting its security, Thus it makes application as vulnerable and easy to attack by hackers and organized crime. In order to prevent these security problems from occurrence of its maximum importance to understand the typical software faults. This paper contributes the knowledge of widely spread two critical web applications by presenting a field study on most of vulnerabilities like SQL Injection and XSS. By analyzing the security patches of source code which are widely used in web applications written in weak and strong typed languages. In order to understand the way in which these vulnerabilities are really exploited by hackers, and also provides an analysis of the source code of the scripts used to attack them. With the outcomes of this result and its study can be used to train code inspectors and software developers in the detection of such software faults, and also with that outcomes research for realistic vulnerability and attackers can be used to assess security mechanisms, like vulnerability scanners, intrusion detection systems, and static code analyzers. By using various number of software testing techniques tools various level of vulnerability are identified and recovery mechanisms were suggested.

KEYWORDS: Internet Applications, Security, Languages, Review and evaluation.

INTRODUCTION

In computer security, vulnerability is a weakness which allows an attacker to reduce a system's information security assurance. The Web Application Vulnerability Assessment is a method to test that assesses the security of interactive applications using web technologies such as e-banking, news and e-commerce web applications. There are number of reasons for the organizations to conduct a vulnerability assessment. It simply conducts a check-up regarding overall web security risk. If organization has number of servers and more than a standalone handful of applications and the vulnerability assessment of such a large scope could be enormous. The major thing needs to discusses is what applications need to be assessed and for what.

The scope could be the web security of a single and ready-to-be deployable application. With the identified vulnerability the recovery measures are analyzed with higher level of risk evolution procedure.

FAULT DETECTION

Error detection mechanisms form the basis of an error resilient system as any fault during operation needs to be detected first before the system can take a corrective action to tolerate it. The scanner tool relies on analyzing the source code of the application depends on ASP.NET files and the code behind files such asC sharp C# and Visual Basic VB for the detection of security vulnerabilities and leaks. Therefore, the scanner tool tries to detect the vulnerabilities that can help hackers from the reflected output or messages, it check most of the ASP.NET server controls and the commands in the code behind that interact with the database.

FAULT RECOVERY

Fault recovery is a Mechanism and technique employed to reestablish the desired level of operation of a component. After scanning process, it will generate a report list all the discovered leaks and vulnerabilities by displaying the name of the infected file with the description and its location. The suggested recovery mechanism will help organization to fix the vulnerabilities and improve the overall security. With this report the organization need to take the necessary corrections steps in order to increase the overall security.

RELATED WORK

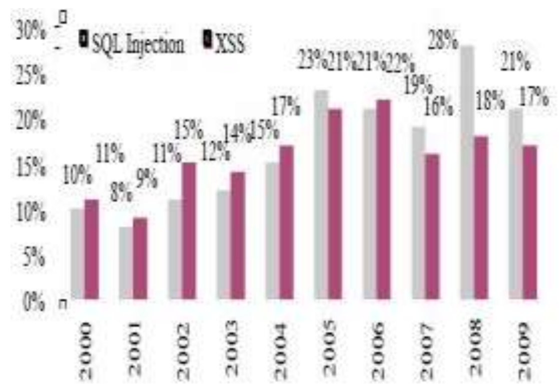
Security vulnerability is an attack against a system, including things like incorrectly configured systems, malware, passwords written on sticky pads, and so on. Thus it makes the system more vulnerable by increasing the security. However, this is a broader connotation somewhat generally used within the security community. Vulnerability gives the main definitions relating to dependability, availability, a generic concept including a special case of such attributes as reliability, maintainability, safety, Security brings in concerns for confidentiality, integrity, in addition to availability and integrity. Basic definitions of security are given first. They are discussed upon, and then added by additional definitions, which address the threats to security and its dependability (failures, faults, errors,), their attributes, and the means for their achievement like fault forecasting, fault prevention, fault tolerance and fault removal, [1]. Reference also found that a novel algorithmic improvement with GenProg allows scaling to large programs and finding repairsupto68%. By using GenProg will generate a large effective benchmark that set to use for systematic evaluations, and inherent parallelism using cloud computing resources to provide grounded, human competitive cost measurements, and.[2]With the reference of Precise Alias Analysis the problem of vulnerable web applications by means of static source code analysis with its address. With this precise alias analysis targeted at the unique reference semantics commonly found in scripting languages. Moreover, it enhances the quality and quantity of the generated vulnerability reports by employing a novel method [3].

Various references provided the automatic programing repair is the initial step toward the automatic detection of application oriented logical vulnerabilities. In this they first use dynamic analysis with that they observe the normal behavior of a web application to infer a simple set of behavioral specifications. Then, it provides the knowledge about the typical execution procedure of web applications, And filter the learned specifications to reduce false positives, by using the model checking over symbolic input for the identification if program paths that are likely to violate the specifications under specific conditions, it indicate the presence of logical fault in certain type of web application [4].

EFFECT OF PROGRAMMING LANGUAGES WITH ITS VULNERABILITY

In Web Application Security the most top Project Report listed of the ten critical web application security risks, having SQLi at the top, and XSS in second [].reference also found that XSS and SQLi as the most wide extent or occurrence; vulnerabilities []. Figure 1 depicts the yearly percentage of disclosed XSS and SQLi among all the causes of web application vulnerabilities showing that they are increasing over time [Neuhaus10]. SQLi attacks had higher advantage of invalidated input fields in the web application interface to characterize the SQL query sent to the back-end database. By exploiting XSS vulnerability, the attacker is used to send or inject the unintended client-side script code web pages, usually HTML and Javascript.SQLi and XSS allow attackers to access unauthorized data like read, insert, change or delete, gain access to privileged database accounts, impersonate other the administrator, mimic web applications, view and manipulate remote files on the server, inject and execute server side programs that allow the creation of botnets (collection of Internet-connected).

Figure:



Reports of SQLi and XSS causes of vulnerabilities

Programs communicating with other similar programs in order to perform tasks controlled by the attacker. The most common vulnerabilities, including XSS and SQLi, the reasons of their existence, attacks, best practices to avoid, detect and make them less severe can be found in many referenced works, [1].Most of the programming languages are currently used to develop web applications. Ranging from languages like C#, VB. To open source languages (PHP, CGI, Perl, Java), Programming languages can be classified using taxonomies, like the programming paradigm, the type system, the execution mode. The type system, mainly focus on the context of the present application, specifies how data structures and its data types are managed with the constructed language, in particular 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 weak vs strong typed static vs dynamic typed [2]. In particular strong typed languages is one in which each type of data (such as integer, character, hexadecimal, packed decimal, and so forth) is predefined as part of the programming language and all constants or variables defined for a given program must be described with one of the data types (e.g., a string cannot be treated as a number), as in weak typed languages. One of the contributions of this work is to help understanding the impact of the type system in the security of web applications. This is of particular significance, as critical security vulnerabilities like XSS and SQLi are strongly related to the way the language manages data types [4]. For example, it is common to find attacks that inject SQL code by taking advantage of variables that supposedly should not be strings (e.g., numbers, dates) as the type of the variable is determined based on the assigned value. On the other hand, in strong typed languages this is not possible because the type of variables is determined before run time and the attempt to store a string in a variable of another type raises an error. However, this does not prevent the occurrence of vulnerabilities in strong typed languages, but only by taking advantage of string variables. In fact, although Java is intrinsically a safe programming language [2] and it is a strong typed language, vulnerabilities can be found in Java programs due to implementation faults [3].

VULNERABILITY CLASSIFICATION AND ASSESSMENT APPROACH

CLASSIFICATION OF SOFTWARE FAULTS - SECURITY VULNERABILITIES

SQL Injection

SQL Injection is a form of attack that can occur when an application uses user input that has not been checked to see that it is valid and the hacker uses this malicious input to exploit sensitive information from the database.

For example, the user can enter the following malicious input:' OR 1=1 --

This would turn the database query into:

SELECT au_lname, au_fname FROM authors WHERE au_id = '' OR 1=1 --

Since 1=1 always evaluates to true, this query will always return more than 0 rows.

Cross Site Scripting

Cross Site Scripting occurs mainly in dynamic web pages that are mixing of browser data (HTML) with the code (<script> tag) which is embedded in the data. The script can be (JavaScript, VBScript, ActiveX, HTML, or Flash).The main objective of 'XSS' is to manipulate client-side scripts of a web application to execute in the manner desired by the malicious user.

There are two main types of Cross Site Scripting:

- Stored Cross Site Scripting
- Reflected Cross Site Scripting

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. 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> located there. The application here acts as an attack site but works for the hacker.

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, 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. The attacker can craft an email contains a link request from the user to click on the link to update personal data.

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.
Example:

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

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". 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>
```

IFrame:

The <iframe> tag specifies an inline frame. An inline frame is used to embed another document within the current HTML document. 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:

- There is no input validation control for the inputs coming to an application.
- There is no sanitization control for the output coming from the application.

Prepared statements and stored procedures

Many of the more mature databases support the concept of prepared statements. What are they? They can be thought of as a kind of compiled template for the SQL that an application wants to run, that can be customized using variable parameters. Prepared statements offer two major benefits:

The query only needs to be parsed (or prepared) once, but can be executed multiple times with the same or different parameters. When the query is prepared, the database will analyze, compile and optimize its plan for executing the query. For complex queries this process can take up enough time that it will noticeably slow down an application if there is a need to repeat the same query many times with different parameters. By using a prepared statement the application avoids repeating the analyze/compile/optimize cycle. This means that prepared statements use fewer resources and thus run faster.

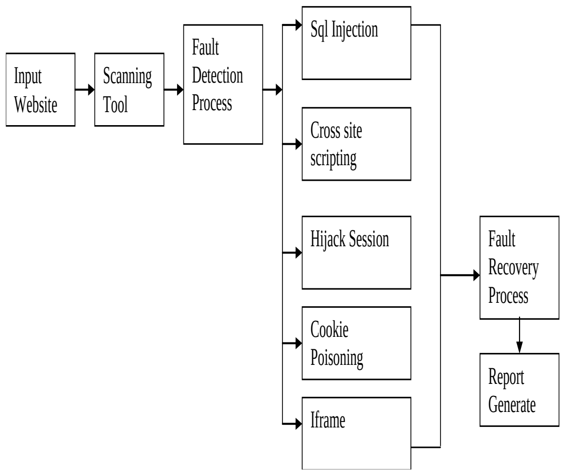
The parameters to prepared statements don't need to be quoted; the driver automatically handles this. If an application exclusively uses prepared statements, the developer can be sure that no SQL injection will occur (however, if other portions of the query are being built up with unescaped input, SQL injection is still possible).

Prepared statements are so useful that they are the only feature that PDO will emulate for drivers that don't support them. This ensures that an application will be able to use the same data access paradigm.regardless of the capabilities of the database.

SYSTEM MODEL

The web application dump files are scanned with the scanning tool and provide a report with the classified vulnerabilities. All the discovered leaks and vulnerabilities by displaying the name of the infected file, the description and its location. 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.

Figure:



System Architecture

CONCLUSION

Thus the scanning tool were used to verify a class of security properties like vulnerabilities for web applications. In particular, for the detection of SQL injection, Cross Site Request Forgery, and Cross Site Scripting. Based on the analysis of vulnerabilities fault locations were identified, based on the impact of vulnerabilities recovery mechanism where suggested and also with the work on syntactic and semantic checking of dynamically generated database queries (SQL) XSRF, XSS were analyzed.

REFERENCES

1. Avizienis.A, Member, IEEE and Vytautas Magnus Univ, Senior Member, "IEEE,taxonomy of dependable and secure computing", 2011.
2. Le Gues, C.Michael Dewey-Vogt "A Systematic Study of Automated Program Repair: Fixing 55 out of 105 Bugs for $8 each", International Conference on Software Engineering, 2012.
3. Jovanovic, N., Kruegel, C., Kirda, E., "Precise Alias Analysis for Static Detection of Web Application Vulnerabilities", IEEE Symposium on Security and Privacy, 2009.
4. ViktoriaFelmetsger, LudovicoCavedon "Automatic Detection of Web Application Security Flaws" 2011.
5. José Fonseca, NunoSeixas, Marco Vieira, Henrique Madeira "Analysis of Field Data on Web Security Vulnerabilities" 2013.
6. Fogie, S., Grossman, J., Hansen, R., Rager, A., Pektov, P., "XSS Attacks: Cross Site Scripting Exploits and Defense", Syngress, 2007.
7. José Fonseca, NunoSeixas, Marco Vieira, Henrique Madeira "Analysis of Field Data on Web Security Vulnerabilities"IEEE Transactions On Dependeble And Secure Computing ,2013