Static Analysis Approaches to Detect SQL Injection and Cross Site Scripting Vulnerabilities in Web Applications: A Survey

Mukesh Kumar Gupta
Department of Computer Engineering
Malviya National Institute of Technology, Jaipur, India
mukeshgupta@skit.ac.in

M.C. Govil
Department of Computer Engineering
Malviya National Institute of Technology, Jaipur, India
govilmc@yahoo.com

Girdhari Singh
Department of Computer Engineering
Malviya National Institute of Technology, Jaipur, India
girdharisingh@rediffmail.com

***Abstract* - Dependence on web applications is increasing very rapidly in recent time for social communications, health problem, financial transaction and many other purposes. Unfortunately, presence of security weaknesses in web applications allows malicious user's to exploit various security vulnerabilities and become the reason of their failure. Currently, SQL Injection (SQLI) and Cross-Site Scripting (XSS) vulnerabilities are most dangerous security vulnerabilities exploited in various popular web applications i.e. eBay, Google, Facebook, Twitter etc. Research on defensive programming, vulnerability detection and attack prevention techniques has been quite intensive in the past decade. Defensive programming is a set of coding guidelines to develop secure applications. But, mostly developers do not follow security guidelines and repeat same type of programming mistakes in their code. Attack prevention techniques protect the applications from attack during their execution in actual environment. The difficulties associated with accurate detection of SQLI and XSS vulnerabilities in coding phase of software development life cycle. This paper proposes a classification of software security approaches used to develop secure software in various phase of software development life cycle. It also presents a survey of static analysis based approaches to detect SQL Injection and cross-site scripting vulnerabilities in source code of web applications. The aim of these approaches is to identify the weaknesses in source code before their exploitation in actual environment. This paper would help researchers to note down future direction for securing legacy web applications in early phases of software development life cycle.**

Keywords— web applicatio; static analysi; vulnerabilitie; SQL injection; cross site scripting

I. INTRODUCTION

The popularity of web applications for social communications, health problem, and financial transaction are increasing very rapidly. Unfortunately, software vulnerabilities are becoming very critical issues in these web applications. According to most recent website security statistics report, 63 percent of assessed websites are vulnerable, each having an average of six unsolved flaws [19]. In 2013, *Open Web Application Security Project (OWASP)* [2] and *Common Vulnerabilities and Exposures* (CWE) [1] reported cross site scripting (XSS) and SQL injection (SQLI) are in top 10 most serious vulnerabilities in web based system. In December 2011, SANS Institute security experts reported a major SQL injection attack (SQLIA) that affected approximately 160,000 websites using Microsoft's Internet Information Services (IIS), ASP.NET, and SQL Server frameworks. In addition, cross-site scripting based MySpace Samy worm infected more than a million accounts within the first 20 hours of infecting the MySpace social networking website. The already deployed applications, we call legacy applications, have serious security problems. These applications are often written with insufficient knowledge of the possible security threats or employ insufficient means to prevent them. In summary, a significant majority of legacy web applications are vulnerable. The main reason of vulnerabilities is weaknesses present in source codes. These may be programming language weaknesses, improper input validations or occurred due to ignorance of security guideline by developers. Researchers investigated that the cost of detecting and correcting a software weakness increases along the phases of software development life cycle [7]. Hence, it is necessary to detect security weaknesses in development phase for saving money and avoiding software failure. This paper proposes a classification of various software security approaches in software development life cycle. In literature, various survey papers [5, 6, 7, 8, 20, 21] discussed vulnerability detection approaches that detect vulnerabilities in testing phase or in deployment phase. This paper summary the static analysis approaches that detect vulnerabilities in coding phase of software development life cycle. The aim of these approaches is to identify weaknesses in source code before their exploitation in actual environment.

The rest of paper is organized as follows. Section II gives an overview of SQL Injection and Cross Site Scripting in web applications. Section III proposes a classification of various approaches used in development of secure web application. Section IV a survey of static analysis based approaches for detection of vulnerability in coding phase of software development life cycle. Finally, Section V concludes the paper noting and mentions future research directions.

II. BACKGROUND OF SQL & XSS VULNERABILITY

Vulnerability in any system is defined as a bug, loophole, weakness or flaw existing in the system that can be exploited by an unauthorized user in order to perform some attack or gain access to the stored data. In 2013, *Open Web Application Security Project (OWASP)[2]* and *Common Vulnerabilities and Exposures* (CWE) [1] reported cross-site scripting (XSS) and SQL injection (SQLI) are in top 10 most serious vulnerabilities in web based system.

A. SQL Injection Vulnerability

SQL Injection [8] is a type of security exploit, in which SQL queries are executed without proper validation of user inputs, to access or alter data. The common variant of this is very simple in which, a malicious user inputs some crafted data, and application uses that data to build a SQL statement. This allows the malicious user to change the purpose of the SQL query. Consider given below PHP example:

```
<?php
$database = mysql_connect("localhost","user","pass");
mysql_select_db("Order",$database);
$input = $HTTP_GET_VARS["input"];
$quries = "SELECT cardnum
          FROM Customer
          WHERE input =%$input%";
$output = mysql_query($quries,$database);
if ($output)
{echo mysql_result($output,0," cardnum");}
else
{ echo "No output " . mysql_error();}
?>
```

In the above example, if malicious user give the value of input as 3 or 2==2 --, then above SQL query be like: SELECT cardnum FROM Customer WHERE input=3 or 2==2 -- , 2==2 is always true for all customer, so the query will return all customer details. Mostly databases i.e. MS SQL Server, Oracle, MySQL, Postgres, DB2, MS Access, Sybase, Informix, etc are which are accessed through applications developed using: Perl, ASP, JSP, PHP etc are potentially vulnerable.

B. Cross Site Scripting Vulnerability

Cross-site scripting (XSS) [3] is a type of code injection vulnerability that enables malicious user to send malicious scripts to web browsers. It occurs whenever a web application uses user inputs in response pages without properly validating them. When a user visits an exploited web page, the browser executes the malicious scripts sent by the application. This is known as XSS attack. XSS attack creates various security violations such as account hijacking, data theft, cookie theft, web content manipulation, and denial of service [3]. Cross Site scripting vulnerabilities are three types: stored, reflected, and DOM-based. Stored XSS vulnerability occurred whenever user inputs are stored in database by the application program. Further those stored data are accessed and used in response page. Stored XSS is generally occurred in forums, blogs, or in social networking sites. Reflected XSS vulnerability occurred whenever user input data is referred in immediate response web page without proper validation. Reflected XSS is commonly occurred error or greeting messages. These two types of vulnerabilities are occurred due to improper validation of user input by server side program. DOM-based XSS vulnerability occurs when client side program uses invalidated user input dynamically obtained from the DOM (document object model) structure. DOM based XSS attacks are occurred client side only.

III. CLASSIFICATION OF SOFTWARE SECURITY APPROACHES

Software security is a process of engineering the software, so that they can functions properly in the presence of malicious attack. Software security aims to avoid security vulnerabilities by considering security aspects in early stages of software development life cycle. These approaches can be categorized into defensive programming (secure coding guidelines), vulnerability detection approaches and attack prevention approaches.

A. Secure Coding Guidelines

Secure coding guideline is a set of practice that should be used by software developers to develop secure software. Developer requires sufficient training to learn how to use secure libraries. SQLI and XSS vulnerabilities occur in system by using invalidated inputs. Secure coding guidelines best way to validate the input and eliminate the chances of such type of vulnerabilities. But, mostly developers either do not follow security guidelines, or repeat same type of programming mistakes in their code. So, secure coding guidelines does not guarantee for weakness free software application.

B. Vulnerability Detection Approaches

Vulnerability detection approaches are used to test applications or detect weakness present in source code of application. In literature, vulnerability detection approach can be classified into static analysis, dynamic analysis and hybrid analysis based approaches. Static analysis approaches analyze the source code without running the application. However, dynamic analysis approaches require executable code to detect the vulnerabilities. These two approaches are applied in coding or testing phase of software development life cycle. Code-based input validation testing methods apply static analysis to extract valid/invalid input conditions for source code of web application. In general, the effectiveness of code-based approaches depends on the test suites used for detecting vulnerabilities in source code. Since static analysis, automatically review the source code during its development phase. So, it reduces the bug finding and removing cost because cost increases along the later stage of development life cycle. On the negative sides, static analyzers are producing a large number of false positives and false negatives. Dynamic analyzers, on the other hand, are accurate and generate less false positives. However, they suffer from run time overhead and require large test cases to ensure a certain confidence level in detecting security bugs. Therefore, Hybrid analysis approaches combine static and dynamic analysis approaches.

C. Attack Prevention Approaches

Attack prevention approaches use run time monitors in client or server side to detect real time attacks in the system. In general, these approaches interpreted incoming or outgoing traffic and checks data for illegal scripts or verifies against security policies.

TABLE 1: CLASSIFICATION OF SOFTWARE SECURITY APPROACHES

Approaches	**Manual / Automatic**	**Vulnerability Detection**	**SDLC phase**
Secure Coding Guideline			
	Manual	No	Coding Phase
	Automatic	No	Testing Phase
Static Analysis Approaches			
Vulnerability Scanner (White Box Testing)	Automatic	Yes	Testing Phase
Vulnerability Analyzer	Automatic	Yes	Coding Phase
Dynamic Analysis Approaches			
Vulnerability and Attack Injection (Black Box Testing)	Manual/ Automatic	Yes	Testing Phase
Dynamic Code Analysis	Automatic	Yes	Coding Phase
Hybrid Analysis Approaches	Automatic	Yes	Coding Phase
Attack Prevention Approaches	Automatic	No	Deployment (Execution) Phase

IV. SURVEY OF SQLI AND XSS VULNERABILITIES DETECTION APPROACHES IN CODING PHASE

In literature, various static analysis approaches are proposed to detect SQLI and XSS vulnerabilities in coding phase of SDLC. These approaches use different techniques of program analysis to trade-off precision and analysis time. Researchers considered following analysis in their vulnerability detection approaches -

1) Flow sensitive analysis: A flow-sensitive analysis uses control flow graph of the source code to provide relationship between data definition and its uses. It analyzes those data which is used after its definition. Flow sensitive analysis is time consuming than flow insensitive analysis but provides precise results.

2) Path sensitive analysis: A path-sensitive analysis takes into accounts only valid paths through the program. A path-insensitive analysis considers all possible paths. Path-insensitive analysis is more time consuming but provide higher precision.

3) Context sensitive analysis: It can be classified as Inter-procedural and Intra-procedural analysis. Inter-procedural analysis analyses functions by considering global variables and actual parameter of a function call and models relationships between various functions. Intra-procedural analysis algorithm only models information flow that does not cross function boundaries, so results in too much false positive and false negative. An inter-procedural analysis is slower than intra-procedural analysis but provide greater precision than intra-procedural analyses.

A. Static Analysis Approaches

Static analysis approaches are able to find out the basic cause of a security problem. Static analysis can find errors early in development, even before the program is run for the first time. Finding an error early not only reduces the cost of fixing the error, but the quick feedback cycle improves the developer coding approach. The attack scenarios and information about code constructs used by a static analysis tool act as a means of knowledge transfer. Researchers proposed various static analysis approaches to detect vulnerabilities from source code of software system. Vulnerability detection approaches has been strengthened by in-depth research in static analysis algorithms in recent times. Peng Li et. al. (2010) [4] and Li Bing chang et. al. [9] compared various static analysis techniques i.e. lexical analysis, type inference, data flow analysis, constraint analysis, symbolic execution etc. These techniques are used to detect different type of vulnerabilities from program source code. However, data flow analysis technique is used to collect dynamic information from the source code. Static taint analysis is a special case of data flow analysis. In this technique, data from external user is marked as tainted data. If tainted data is used in the program without any validation then it indicates the presence of vulnerability. Huang et al. (2004) [25] proposed a static analysis approaches for SQLI and XSS vulnerability detection in PHP. They used only an intra-procedural taint analysis algorithm and thus only models information flow that does not cross function boundaries, so results in too much false positive and false negative. Pietraszek et al. (2005) [11] tracks tainted data under character-level by modifying the PHP interpreter to detect SQL injection vulnerability. Jovanovic et al (2006) [23] proposed a flow-sensitive, inter-procedural and context-sensitive data flow analysis to discover vulnerable points (XSS, SQLI, command injection) in PHP programs. They

TABLE 2 - SUMMARY FOR XSS AND SQLI DETECTION APPROACHES IN CODING PHASE

Author/Approach	**Year**	**Analysis Techniques**	**Language**	**SQLI**	**XSS**
Livshits et. Al. [27]	2005	Inter-procedural Analysis + Flow Insensitive + Pointer Analysis	JAVA	Yes	Yes
Yichen Xie et al. [17]	2006	Flow-Sensitive and Inter-procedural Analysis	PHP	Yes	No
Jovanovic et al. [23] / Pixy [26]	2006	Flow Sensitive + Inter-Procedural + Alias and Literal Analysis	PHP	No	Yes
G. Wassermann et. al.[18]	2008	Tainted information flow with string analysis.	PHP	No	yes
Y. Zheng et. al [29]	2013	Path-Sensitive + Inter-Procedural Analysis	PHP	No	Yes
Binbin Qu et. al.[24]	2008	Taint Dependency Analysis	Java	Yes	Yes
Ettore Merlo et. al.[6]	2006	Flow-Sensitive Analysis+ Inter-Procedural Analysis +Fixed point Algorithm	PHP	Yes	No
H. Atashzar et. al.(7)	2011	Inter-Procedural Analysis +Dynamic Taint Analysis	PHP	Yes	Yes
Zhang Xin-hua et. al. (ASPWC) [22]	2010	Inter-Procedural Taint Analysis	ASP	Yes	Yes
Giovanni Agosta et. al. [13]	2012	Symbolic Code Execution + Inter-Procedural Taint Analysis	PHP	Yes	Yes
J. Huang et.al. [28] /TASA	2010	Path-Sensitive Analysis + Inter-Procedural Analysis	ASP	Yes	Yes

implemented first open source tool [26] for statically detect only XSS vulnerabilities in PHP4 code by means of data flow analysis. Pixy tool does not analysis PHP dynamic feature i.e. included files automatically. Author results shows false positive rate of around 50%. Wassermann et. al. (2008) [18] proposed an approach to detect XSS vulnerability by combining tainted information flow with string analysis.

Authors used string analysis to track untrusted substring value. But not able to detect DOM-based XSS vulnerabilities and cannot handle arbitrary complex and dynamic code. Xin-Hua Zhang et al (2010) [22] implemented a taint analysis based tool to detect XSS attacks and SQL injection vulnerabilities in ASP. It tracks various kinds of external input, tags taint types, construct control flow graph and taint data propagate to various kinds of vulnerability functions, but not free from false positive problem. G. Agosta et. al. (2012) [13] used symbolic execution and string analysis technique. It improve precision by approximated the string values that may appear at sensitive link. However, static analysis techniques are used in various domains to detect the vulnerabilities but not efficient as these may generate false positive and false negative results. Static program analysis reads the code and constructs and analyses an abstract model. These approaches use different techniques of program analysis to trade-off precision and analysis time. Researchers considered following analysis in their vulnerability detection approaches – flow sensitive analysis, flow insensitive analysis, inter-procedural or intra-procedural analysis, path sensitive or path-insensitive analysis. Table 2 summarizes various static analysis vulnerability detection approaches and their analysis techniques.

B. Dynamic Analysis Approaches

Static analysis techniques are used in various domains to detect the vulnerabilities but not efficient as these may generate false positive and false negative results. Dynamic analysis analyzes the information obtained during program execution to detect vulnerabilities. It is perform during testing phase of SDLC or runtime after software release. Haissam et al (2003) proposed a fault injection testing technique that introduces faults in order to test the behavior of the system. Chess et. al (2008) [24] proposed a dynamic taint based approach to detect vulnerabilities. In this method, the tainted data is inspected to check the validity of input before their use during the execution of the program. But dynamic analysis has some limitation like only vulnerabilities present in the execution paths are detected and unable to detect vulnerabilities in parts of code that were not executed. Also, results produced are not generalized for future executions. There is no certainty that the set of inputs over which the program was run is characteristic of all possible program executions.

C. Hybrid Analysis Approaches

Hybrid approaches [10, 14, 15, 16] are a combination of static and dynamic analysis approaches. In these approaches dynamic analyze techniques improve the false alarms of static analysis approaches and provide precise results.

V. CONCLUSION AND FUTURE WORK

Researchers have proposed various approaches to detect cross-site scripting and SQL injection vulnerabilities, but these vulnerabilities continue to exist in many web applications. In this paper, we have proposed a classification of software security approaches used to develop secure software in various phase of software development life cycle. This paper also summarized various static analysis approaches that detect vulnerabilities in coding phase of software development life cycle. The aim of these approaches is to identify the weaknesses in source code before their exploitation in actual environment. Static analysis approaches are able to find out the basic cause of a security problem and can find errors early in development, even before the program is run for the first time. Finding an error early not only reduces the cost of fixing the error, but the quick feedback cycle improves the developer coding approach. But, static analysis approaches still suffer from false positive and false negative results. In future, more research is needed to improve analysis technique for providing precise detection results.

REFERENCES

[1]. Common Vulnerabilities and Exposures (The Standard for Information Security Vulnerability Names) http://cwe.mitre.org/

[2]. https://www.owasp.org/index.php/Top_10_2013

[3]. Lwin Khin Shar, Hee Beng Kuan Tan,"Automated Removal of Cross Site Scripting Vulnerabilities in Web Applications" *in Information and Software Technology*, Vol. 54, Issue 5, pp 467-478, May2012

[4]. Peng Li, Baojiang Cui, "A comparative study on software vulnerability static analysis techniques and tools", *IEEE International Conference on Information Theory and Information Security (ICITIS),* 17-19 Dec. 2010

[5]. R. Johari, P. Sharma, "A Survey on Web Application Vulnerabilities (SQLIA, XSS) Exploitation and Security Engine for SQL Injection," *International Conference on Communication Systems and Network Technologies (CSNT),* pp.453,458, 2012

[6]. M. Pistoia, S. Chandra,S.J. Fink, "A survey of static analysis methods for identifying security vulnerabilities in software systems," *IBM Systems Journal,* vol.46, no.2, pp.265-, 288, 2007

[7]. H. Atashzar, A. Torkaman M. Bahrololum, M.H. Tadayon, "A survey on web application vulnerabilities and countermeasures," *6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT),* pp.647-, 652, 2011

[8]. D.A. Kindy, A. Pathan, "A survey on SQL injection: Vulnerabilities, attacks, and prevention techniques," *IEEE 15th International Symposium on Consumer Electronics (ISCE),* pp.468- 471, 2011

[9]. Li Bingchang, Liang Shi, Zhuhua Cai, "Software Vulnerability Discovery Techniques: A Survey,"*Fourth International Conference on Multimedia Information Networking and Security (MINES),* pp.152-156, 2012

[10]. Monica S. Lam and Michael Martin,"Securing web applications with static and dynamic information flow tracking", *In proceeding of the ACM Symposium on Partial Evaluation and Semantics-based Program Manipulation*, pages 3-12 , 2008

[11]. Scholte, T. Robertson, W. Balzarotti, D. Kirda, "Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis," *IEEE 36th Annual Computer Software and Applications Conference (COMPSAC),* 16-20 July 2012

[12]. Lwin Khin Shar, Hee Beng Kuan Tan, "Defending against Cross-Site Scripting Attacks," *Computer,* vol.45, no.3, pp.55-62, March 2012

[13]. G. Agosta, A. Barenghi, A. Parata, G. Pelosi, "Automated Security Analysis of Dynamic Web Applications through Symbolic Code Execution," *Ninth International Conference on Information Technology: New Generations (ITNG),* 16-18 April 2012

[14]. Ruoyu Zhang; Shiqiu Huang; Zhengwei Qi; Haibin Guan, "Combining Static and Dynamic Analysis to Discover Software Vulnerabilities," *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS),* June, 2011

[15]. Balzarotti, Cova, M. Felmetsger, Kirda, E. Kruegel, C. Vigna, Giovanni, "Saner: Composing Static and Dynamic Analysis to Validate Sanitization in Web Applications," *IEEE Symposium on Security and Privacy,* 18-22 May 2008

[16]. Lwin Khin Shar, Hee Beng Kuan Tan, and Lionel C. Briand. "Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis". In *Proceedings of International Conference on Software Engineering* (ICSE '13). IEEE Press, 2013.

[17]. Yichen Xie and Alex Aiken. "Static detection of security vulnerabilities in scripting languages." In *Proceedings of the 15th conference on USENIX Security Symposium* - vol. 15, 2006

[18]. G. Wassermann, Su Zhendong, "Static detection of cross-site scripting vulnerabilities," *ACM/IEEE 30th International Conference on Software Engineering, ICSE '08,* 10-18 May 2008

[19]. www.whitehatsec.com/home/resource/stats.html

[20]. I.A. Elia, J. Fonseca, M. Vieira, "Comparing SQL Injection Detection Tools Using Attack Injection: An Experimental Study" *IEEE 21st International Symposium on Software Reliability Engineering (ISSRE), 2010,* vol., no., pp.289, 298, 1-4 Nov. 2010

[21]. P. Kumar, R.K. Pateriya, "A survey on SQL injection attacks, detection and prevention techniques" *Third International Conference on Computing Communication & Networking Technologies (ICCCNT),* 26-28 July 2012

[22]. Xin-Hua Zhang; Zhi-jian Wang, "A Static Analysis Tool for Detecting Web Application Injection Vulnerabilities for ASP Program," *2nd International Conference on e-Business and Information System Security (EBISS),* 22-23 May 2010

[23]. N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities," *Proc. of the IEEE Symposium on Security and Privacy, May 2006*

[24]. B. Chess, J. West. Dynamic Taint Propagation: "Finding Vulnerabilities without Attacking" Information Security Technical Report. Volume 13, Issue 1, 2008, 33-39.

[25]. Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D. Lee, and S.-Y. Kuo. "Securing web application code by static analysis and runtime protection". In *Proceedings of the 13th International World Wide Web Conference*, 2004.

[26]. N.Jovanovic, C. Kruegel, E. Kirda. "Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities". In SP'06: Proceedings of the *27th IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2006.

[27]. Livshits, V. Benjamin and Lam, Monica S.: "Finding Security Vulnerabilities in Java Applications with Static Analysis". In SS'05: *Proceedings of the 14th USENIX Security Symposium, Baltimore*, MD, USA, 2005.

[28]. Jianjun Huang, Bin Liang, Jiagui Zhong, Qianqian Wang, Jingjing Cai, "Vulnerabilities static detection for Web applications with false positive suppression," *Information Theory and Information Security (ICITIS), IEEE International Conference on* , vol., no., pp.574,577, 17-19 Dec. 2010

[29]. Yunhui Zheng and Xiangyu Zhang." Path sensitive static analysis of web applications for remote code execution vulnerability detection". In *Proceedings of the International Conference on Software Engineering* (ICSE '13). IEEE Press, 652-661, 2013.