Australian Journal of Basic and Applied Sciences, 9(6) Special 2015, Pages: 47-50

ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

Prediction Model for Input Validation Vulnerabilities in Cloud Based SaaS Web Applications

R.Dhanalakshmi and Riju Thomas

Department of Computer Science and Engineering, KCG College of Technology,Chennai-97.

ARTICLE INFO

Article history:
Received 12 November 2014
Received in revised form 26 December 2014
Accepted 29 January 2015
Available online 10 February 2015

Keywords:
Computing Security; SaaS Web Applications; Input Validation Attack; Vulnerability Prediction; Data mining.
.

ABSTRACT

Cloud computing is the most rapidly growing technology nowadays, without investing in new infrastructure the capacity of computing can be dynamically increasable utilizing this innovation. Notwithstanding of these preferences some enterprise customers still hesitant to have their web provisions in cloud servers, the prime explanations behind that is the inconveniences of information security and vulnerabilities in this domain. Vulnerabilities include injection vulnerabilities and weak authentication. As per OWASP (Open Web Application Security Project) the most prevailing web application injection vulnerabilities are SQL injection (SQLI) and cross site scripting (XSS). This paper proposing to build a prediction model by collecting attributes which characterize input validation routines used in applications to defend these attacks. In this paper 22 code attributes have been proposed and our test study demonstrates the best model of proposed framework is dependent upon MLP classifier on the grounds that it accomplished pd=72% and pf=12% and predicting SQLI vulnerabilities attained pd=82%,pf =19% for predicting XSS vulnerabilities.

To Cite This Article: R.Dhanalakshmi and Riju Thomas., Prediction Model for Input Validation Vulnerabilities in Cloud Based SaaS Web Applications. ***Aust. J. Basic & Appl. Sci.,*** *9(6): 47-50, 2015*

INTRODUCTION

In recent times cloud computing have witnessed a magnificent growth and now it becomes most dynamic technology in IT industry. The advantages of using cloud computing are a) It reduces significant cost because without investing in infrastructure the capability of computing can be increased b) its accessibility, flexibility and scalability. According to Heiser (2009) cloud computing is defined as "a style of computing where massively scalable IT-enabled capabilities are delivered 'as a service' to external customers using Internet technologies''. The various services offered by cloud computing architecture are layered as the Infrastructure as a Service (IaaS) providing infrastructure support services. Platform as a Service (PaaS) layer provides platform oriented services such as the environment for hosting the customer applications. Software as a Service (SaaS) layer which have a characteristic that it offers a entire application as service based on the demand of the client.

In SaaS deployment model since the data is reserved outside the boundary of enterprises, so the service provider itself provide and adopt security checks for ensuring data security either from attack of hackers and malicious users to gain access to unauthorized data to them. Based on Subashini *et al.* (2012) the most prevalent vulnerabilities in this layer are input injection vulnerabilities such as SQLI and XSS. Unvalidated or not encoded user input is the reason for these two attacks. Static and dynamic based vulnerability detection techniques are already proposed to mitigate these attacks. Static approaches usually check heuristic rules to detect code that could be vulnerable but are inefficient in practice due to high false alarm rate. Dynamic approaches are more efficient but it requires analysis techniques like concolic execution which is difficult to implement because of complexity, so the need to find alternate solutions is inevitable. Recent studies show that vulnerability and defect prediction models based on classification algorithms gives promising results.

Web developers include input sanitization as well as input validation routines to mitigate SQLI, XSS, buffer overflow, path reversal in their web application. Hence characteristics of such functions could be more appropriate in building prediction models. Input validation and sanitization routines includes both language inbuilt routines and user defined routines so static and analysis of execution trace of

Corresponding Author: R.Dhanalakshmi, Department of Computer Science and Engineering, KCG College of Technology,Chennai-97.

Australian Journal of Basic and Applied Sciences, 9(6) Special 2015, Pages: 47-50

program need to be performed to obtain code attributes.

In this paper, we are proposing to categorize input sanitization and validation routines used in a web application into a set of attributes and using that attribute dataset we train and create prediction model for predicting vulnerabilities in new web applications.

Related Works:

1.1 Security Measures in Cloud Computing:

In order to defend security threats like SQLI, XSS cloud service providers adopt several techniques, some of them are avoiding dynamically generated SQL statements in code that validate and sanitize user entered values. According to 'Symantec MessageLabs Web Security.cloud' architecture the security of cloud system lies in two components Bhadauria *et al.*(2012) a) Multi Layer Security b) URL filtering.Multi Layer Security aims to block malwares and URL filtering reduces aims to retract from harmful web pages and to protect bandwidth.

The study (2013), uncovers that SQLI is most discovered weakness in SaaS and suggests some generic solutions to avoid SQL Injection like use parameterized queries, validate user inputs by using inbuilt functions and user defined function and by using stored procedures.

Vulnerability Prediction:

Vulnerability prediction based on analyzing the code or by collecting attributes characterizes the vulnerability and building prediction models comes under this area.

In this domain researchers relate software attributes such as static code code attributes with defects Tosun *et al*. (2009) or vulnerabilities Shin *et al*. (2011). The commonly used static code attributes are Halstead complexity measures and cyclomatic, complexity attributes, Lines of Code etc.

2. Proposed Method:

Developers implement input sanitization and validation routines to prevent vulnerabilities such as SQLI and XSS. An application is vulnerable if the input validation routines are inadequate or there is no such are implemented. From this observation it can be able to say that distinctiveness of this sanitization and validation code routines than other attributes in a program could be more suitable for building a prediction model for predicting program's vulnerability. From the study of related work it is known that attributes characterizing input validation routines can be statically and dynamically collected. With help of those studies this paper proposing to use only input sanitization code attributes that target vulnerable code at statement level for building prediction model. The static analysis is performed based on Shar *et al.* (2012) proposed methodology. The proposed system architecture is shown if Fig 1.

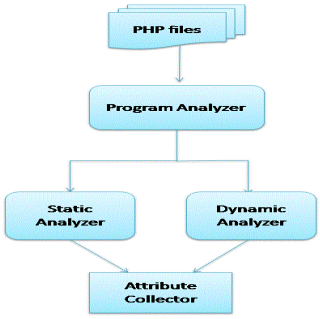

Fig. 1: Proposed System Architecture.

The proposed system has two phases; first one is for collecting static code attributes and second is for collecting dynamic code attributes. Static Analysis phase is implemented based on the tool pixy (2006), which act as program analyzer. Dynamic analysis phase uses a type of pattern mining approach by analyzing their execution traces of input validation routines in program. Based on the above static and dynamic analyzers the attribute vector of each sinks in PHP files can be collected as the output of the system.

Result and Analysis:

There is a possibility of different classification algorithms gives variant performances. So that in our work, we built C4.5 and Multi Layer Perceptron (MLP) models for the experiment. These classifiers were benchmarked as among the top classifiers in recent studies.These classifiers allocate a sensitive sink to a target classes that is either 'assailable' or '*not-assailable*', the classifiers are implemented using data mining tool *WEKA*. We have used 10 fold cross validations on the training data.

According to confusion matrix true positives (tp), true negatives (tn), false positives (fp), and false

Australian Journal of Basic and Applied Sciences, 9(6) Special 2015, Pages: 47-50

negatives (fn), are the possible results of a prediction with target classes "true" and "false". The performance of learned predictor is then evaluated using the performance measures probability of detection (pd),probability of false alarm(pf),. precision (*pr*), and accuracy (*acc*).

Using the APIs provided by WEKA, the proposed system was enabled to run each of 2 classifiers on dataset collected. Each classifier was run twice, for finding both SQLI and XSS attacks. The dataset are composed from four php applications, they are geccbblite0.1, webchess0.9.0, and eve1.0, yapig .The dataset is shown in Table 2.

Table 3 depicts the performance of SQLI vulnerability predictors and performance of XSS predictors, the overall results were averaged. Based on the average results we can say that MLP was the best predictor. For predicting SQLI vulnerabilities, on average MLP achieved pd=72% and pf=12% and for predicting XSS vulnerabilities MLP achieved pd=82%,pf =19%.

Table 1: Proposed Attributes.

Att. ID	Attribute Name	Description
Static Attributes		
1	User Input	The count of nodes in CFG that obtain input directly from users.
2	FileNodes	The count of nodes in CFG that obtain input from files.
3	DBNodes	The count of nodes in CFG that access input from database.
4	PersistentDataObject	The count of nodes in CFG which takes input from persistent data objects.
5	Uninititialized	The count of nodes which is not initialized.
6	SQLISinks	The count of SQLI sinks nodes in CFG.
7	XSSSinks	The count of XSS sinks nodes in CFG.
8	Size	The count of nodes that invoke functions which return the size or length of an argument.
9	Match	The count of nodes that invoke functions that compare two arguments.
10	RegularExpression Match	The count of nodes in CFG that perform regular expression based matching functions.
11	Num-Check	The count of nodes that perform check for numeric data type is given in the argument.
12	SQLI-Purifier	The count of nodes that perform check for numeric data type is given in the argument.
13	XSS-Purifier	The count of nodes that uses inbuilt routines for preventing SQL Injection attacks.
14	Encoded-Node	The count of nodes that uses inbuilt routines for preventing Cross Site Scripting attacks.
15	Encrypted-Node	The count of nodes in CFG that performing encoding of data
16	String-Replace	The count of nodes that that do regular expression based string replacement.
17	RegularExpression Replacement.	The count of nodes that convert other data into numeric data.
18	Num-Convert	The count of nodes that invoke function which return already defined information.
19	Defined	The count of user specified methods used in the program.
20	User specified	The count of nodes that belongs to other categories than from above mentioned.
Dynamic Attributes		
21	User defined Sanitizing and Validation Function.	The count of nodes that invoke user defined sanitizing and validation functions.
Target Attribute		
22	Assailable?	Class attribute which specify whether node is assailable to vulnerability attack.

Table 2: Data Set.

Test Sub	#ScriptSink	%Vul XSS	#DBSink	%Vul SQLI
geccbblite	20	50	9	44
webchess	133	28	127	67
eve	17	83	59	14
yapig	20	40	-	14

Table 3: XSS and SQLI Prediction Results.

Data & Classifier		Pd	Pf	Pr	Acc	Pd	Pf	Pr	Acc
		XSS Prediction Result				SQLI Prediction Result			
geccbblite	C4.5	60	40	60	60	75	20	75	78
	MLP	70	20	58	60	100	20	80	89
webchess	C4.5	97	1	97	99	50	12	75	85
	MLP	98	3	95	98	60	8	50	97
eve	C4.5	78	32	80	94	50	12	75	85
	MLP	93	20	50	82	57	8	50	88
yapig	C4.5	75	16	75	81	-	-	-	-
	MLP	67	17	85	76	-	-	-	-
Average	C4.5	78	23	79	84	58	15	75	83
	MLP	82	19	72	79	72	72	12	91

Conclusion:

The objective of this paper is to propose a framework that aid security auditing in cloud based SaaS web applications and testing by providing probabilistic alerts about potentially vulnerable code statements. For that purpose attributes based on static and dynamic code analysis which characterizes input validation and sanitization code patterns for preventing vulnerabilities are used.

By conducting experiments on four test subjects we found out that the best predictor is MLP which achieved a pd=72%, pf=12% for predicting SQL Injection and pd=82%, pf=19% for predicting Cross Site Scripting. The outcomes of our study delineates that our framework is an effective technique to predict vulnerability.

REFERENCES

Bhadauria, R., S. Sanyal, 2012.“Survey on Security Issues in Cloud Computing and Associated Mitigation Techniques”,in: International Journal of Computer Applications, 47(18): 47-66.

CloudComputingSeries, 2013. <https://cloudcomputing.syscon.com/node/1624391> .

Heiser, J., 2009. What you need to know about cloud computing security and compliance, Gartner, Research, ID Number: G00168345.

Jovanovic, N., C. Kruegel, E. Kirda, 2006. “Pixy: a static analysis tool for detecting web application vulnerabilities”, in: Proceedings of the IEEE Symposium on Security and Privacy, 258-263.

Shar, L.K., H.B.K. Tan, 2013. ”Predicting SQL injection and cross site scripting vulnerabilities through mining inputsanitization patterns”,in:Information and Software Technology, 55(10): 1767-1780.

Shar, L.K., H.B.K. Tan, 2012. "Mining input sanitization patterns for predicting SQL injection and cross site scripting vulnerabilities", 34th International Conference onSoftware Engineering (ICSE).

Subashini, S., V. Kavitha, 2012. “A survey on security issues in service delivery models of cloud computing. J Network ComputerApplication”in:Journal of Network and Computer Applications, 34: 1-11.

Shin, Y., A. Meneel, L. Williams, J. Osbourne, *2011.* Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities, IEEE Transactions in Software Engineering, 37(6): 772-787.

Tosun, A., B. Turhan, A.Bener, 2009. “Validation of network measures as indicators of defective modules in software systems”, in: PROMISE '09 Proceedings of the 5th International Conference on Predictor Models in Software Engineering, 1-12.

Witten, I.H., E. Frank, Morgan Kaufmann, 2005. Data Mining, second ed.