I J C T A, 10(8), 2017, pp. 159-165 ISSN: 0974-5572

Detection and Elimination of Vulnerabilities in Web Applications using Data Mining and Static Analysis

D.D. Pukale[1], Neha Gaikwad[2], Pranali Dhole[3], Kajal Chauhan[4] and Komal Kumari[5]

ABSTRACT

Security in web application is an important issue in present web. A noteworthy reason for this problem is insufficient knowledge of software engineer about secure coding, which leads to vulnerabilities in web application. This issue can be handled by the use of static analysis on source code which will detect input vulnerabilities. But this approach can result in detection of numerous false positives. Hence our paper approaches a combination of techniques that will detect input vulnerabilities but with lesser false positives. The problem of false positives will be solved by the usage of data mining. Our approach will use two inverse methodologies: firstly, people will code the information regarding input vulnerabilities (for taint analysis), secondly automatic detection of these vulnerabilities in source code will be done (with machine learning, for data mining). Here we will do programmed code amendment by embedding fixes in the source code. Afterwards diverse testing techniques like regression testing will be used to ensure if the code after rectification acts of course and each vulnerability is evacuated. The approach will be actualized in the WAP instrument and a trial assessment will be performed with numerous web applications with PHP source code to guarantee the accuracy of hardware.

Keywords: Data mining, False Positives, Input Vulnerabilities, Source code.

1. INTRODUCTION

In today's world one of the most important concerns regarding World Wide Web is 'security'. Everyday newspapers are flooded with various hacking related news of different web pages. So this paper explores an approach that will detect input vulnerabilities but will also build programmer's ability to develop better and secure web pages in future. Here the input vulnerabilities will be detected by examining the source code. The vulnerabilities will be fixed afterwards and the programmer will be informed about these vulnerabilities. This will help programmer to improve their coding practise. After fixing the source code, the programmer will be provided a testing module to check if the code has been fixed properly or there is still the presence of input vulnerabilities [5]. To find the vulnerabilities, taint analysis (one of the form of static analysis) will be used, but it have one major problem associated with it. It shows the presence of vulnerabilities when it is not actually present in the source code, this is also known as false positives [9]. This problem becomes more difficult with PHP language which is weakly coded by the programmers and hence leads to security threats. In 2013, 9% of all vulnerabilities recorded by the National Vulnerability Database were connected to PHP. In this manner, we use a combination of taint analysis and data mining to find vulnerabilities and to detect false positives respectively. This approach will use two inverse methodologies: firstly, people will code the information regarding input vulnerabilities (for taint analysis), secondly automatic detection of these vulnerabilities in source code will be done (with machine learning, for data mining). Many efforts have been taken in this area to solve this problem, but much success has not been achieved till now. Mostly two approaches have been used separately. First approach is intrusion

[1-5] Bharati Vidhyapeeth's College of Engineering for Women, Pune (Department of Computer Engineering), *Emails: pukaledd@gmail.com, ngaikwad1052@gmail.com, pranali.dhole74@gmail.com, kajalchauhan440@gmail.com, kkm42921@gmail.com*

detection. In this approach, detection totally relies on the information provided by the programmers whereas in the second approach i.e anomaly based detection, detection is done with the help of machine learning. In our approach, we have used the combination of these two approaches. Human will code the information regarding the input vulnerabilities and machine learning will help to detect these vulnerabilities.

Here the concern that comes along with taint analysis is the presence of false positives. So to solve this problem we take help of data mining techniques. The data mining techniques will be performed after the implementation of taint analysis and it will only measure the parts of the code that we doubt to have false positives present in them, and will use a combination of classifiers to label each vulnerability as true or false. Classifiers used for this purpose will be: ID3, Random Forest, Random Tree, Logistic Regression , C4.5/J48, Naive Bayes, K-NN, Bayes Net, SVM and MLP. We explore the various induction rules like PART, JRip, Prism and Ridor to present the attributes associated with false positives[9].

2. EXISTING SYSTEM

The WAP tool is one of the recent developments that detect vulnerabilities of eight classes in PHP web application. WAP can be improved to detect different types of input vulnerabilities. It is able to handle the vulnerability classes like SQLI, XSS, remote file inclusion (RFI), local file inclusion, OSCI, PHP command injection, Source Code Disclosure. The tool has knowledge that is imparted manually about the vulnerabilities[6]. It verifies if the input can reach the sensitive sinks without proper sanitization or validation.

Table 1. Shows the example of XSS, its sensitive sinks, entry points, and sanitization functions.

Table 1
Entry points, Sensitive sinks and sanitization functions for XSS vulnerabilities.

Entry Points	*Sensitive Sinks*	*Sanitisation Functions*
$_POST	Print	Htmlentities
$_GET	echo	htmlspecialcha
$_REQUEST	die	rs
$_COOKIE	printf	strip_tags
HTTP_POST_VA	exit	urlencode
RS	error	
HTTP_GET_VA	file_put_contents	
RS	fprintf	
HTTP_COOKIE_	file_get_contents	
VARS	fgets	
$_FILES	fgetc	
$_SERVERS	fscanf	

The three main components of WAP are: 1) Taint Analyzer, 2) Data Mining Component, 3) Code Corrector.

The Taint Analyzer is divided into three different parts of data about each vulnerability class: sanitization functions, entry points, sensitive sinks. But taint analyzer tends to generate many false positives. A false positive is a term used when taint analyzer identifies the vulnerabilities that are not true. The entry points are the functions that reads input parameters, such as, $_GET, $_POST are always a different of the same set, where as the rest tend to be simple to recognize the vulnerability type. The data mining segment has to be trained with new knowledge set about false positives that may occur while processing the code for the new class. Initially training might be skipped, and upgraded incrementally when more data becomes available. For the training of the data set, we need data about candidate vulnerabilities which are found by the taint analyzer,

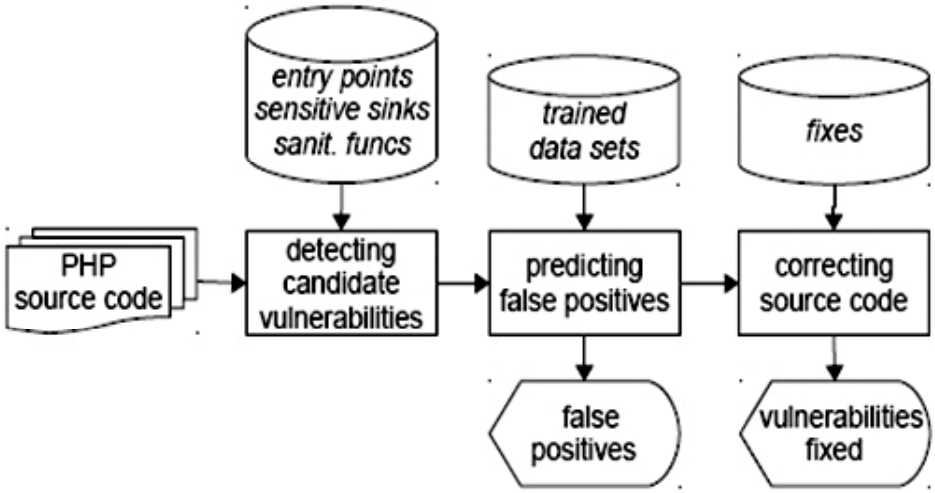

Figure 1: Overview of the WAP tool

which have to be labeled as true positive or false positives. False positive predictor module classifies the vulnerability as false positive or false negative. Then, the attributes associated to the false positives have to be used to configure the classifier[6]. The code corrector needs information about what sanitization function need to be used at a particular point to handle that class of vulnerability, and where it shall be embedded. Again, getting this information is possible once the new class is known and understood. Tainted model is being used while performing static and dynamic analysis for finding the security analysis. This model is implemented by means of static analysis in PHP for WAP tool. Lack of formal specification of PHP is one of the limitations of WAP tool. One of the disadvantages is that amid the testing of the tool with many open source applications, a few times WAP was not able to parse the source code for absence of a grammar rule to deal with complicated constructions. With time, these rules were added, and these problems are solved. Pixy uses taint analysis for verifying PHP code, but expands it with *alias analysis* that considers the existence of aliases, i.e., of two or more variable names that are utilized to denominate the similar variable. Furthermore, Pixy does only module-level analysis, whereas WAP does global analysis (i.e., the analysis is not limited to a module or file, but can involve several). Shar and Tan created tools named PhpMinerI, and PhpMinerII, which are two tools that use data mining for detecting the presence of vulnerabilities in PHP programs. The attribute sets are extracted from the code slices and data mining algorithms are applied to these attributes, leading to detecting vulnerabilities. WEKA tool performs the process of data mining. Some of the applications have also used Penetration Testing approach which is based on stimulation of attacks against web application. As this approach is implemented as black box testing so the scope is limited to HTTP responses.

The following is the techniques used for vulnerability scanning: Static Analysis- This technique is known as one of the fast, reliable and efficient method to detect the vulnerabilities. It performs the analysis of the program structure by using various methods. To detect the flaws in the code, techniques used are: lexical analysis, constraint analysis etc. Many of the applications are implementing this technique so as to detect the flaws along with the additional modules.

3. LITERATURE SURVEY

1. *Sonam Panda, Ramani S* proposed a static examination algorithm to determine SQLCIVs. It determines the arrangements of conceivable database queries that a web application may create utilizing situation free grammars and tracks flow of data through untrusted sources into those grammars [8]. By making use of a general meaning of SQLCIVs based on the background of unreliable substrings, we can dodge the requirement for manually inscribed policies. It was precise, notable unknown liabilities in the web applications of real world with multiple negatives, showing the feasibility of our method [8].

2. *L. K. Shar and H. B. K. Tan* put forward the discovery of only two major input liabilities comprising of SQL Injection and XSS by the combination of static and dynamic analysis of the code [3]. Wherein the hybrid characteristics of the code are collectively classified from the graph of data dependency. Nodes that have definite problems related to security are then classified using static analysis [3].

3. *Mounika B, A. Krishna Chaitanya* projected an inevitable system that provided an option to produce an output sanitization which is robotized input approval (IPAAS) and it is displayed for avoiding XSS and SQL assaults [5]. This method advances the safe improvement of web applications by accomplishment of parameter extraction and different type learning techniques by applying commanding information validators at runtime [5].

4. *N. L. de Poel*, proposed SAFERPHP, which is a static investigation system used for detecting the liabilities in PHP source code. This agenda employs various algorithms including: 1) To implement new type of symbolic performance to check denial-of-service liabilities. 2) A new type of infer procedural analysis to verify the application sanction policy and find misplaced checks before any complex database operations [2].

5. *Y.W. Huang* anticipated an approach which gave quick protection at a much lower cost than others, since approval is confined to potentially weak segments of code [6]. If Web SSARI classifies the use of untrusted data taking after right treatment, the code is left as it is. As per several experiments, Web SSARI carried only 0.02 percent of all the statements to be inspected with inappropriate sanitization agendas [6]. Interestingly, Sharp and Scott implements global justification for each data submitted by the user without any complaint and without even concerning that the same validation process may be implemented by web application as well. This finally outcomes in pointless upstairs [6].

4. OUTLINE OF THE PAPER

The paper basically focuses on following points:

1) Improvement in security of web applications by detecting and removing input vulnerabilities in source code of web applications.
2) Usage of taint analysis and data mining techniques for detection of vulnerabilities with fewer false positives.
3) Automatic correction in the source code and informing programmer about it.
4) Finally experimenting the tool with various web applications to see the correctness of tool.

5. PROPOSED SYSTEM

This paper proposes a system to check for vulnerabilities and detect them and also eliminate those detected vulnerabilities in Web Application. It also proposes a tool that will scan PHP applications to detect and remove vulnerabilities and input validation vulnerabilities. The tools for implementation of the problem in proposed system is made by combination of two techniques : 1) Data mining 2) Static source code analysis. To identify the false positives data mining will be used along with the top three machine learning classifiers. Induction rule classifier will be used to confirm the presence of the false positives. The selection of classifiers is done on basis of comparison between the available appropriate alternatives. The single detection technique fails to provide correct results so different detection techniques are combined. But they also fail to provide entirely correct results. After detection of candidate vulnerabilities, it will be checking for false positives .The proposed system comprises of the tool which will replace the vulnerable code by fixes .Fixes are nothing but the correct code. After replacing the code by fixes, testing will be performed to check for the correct working of the system. It will check the applications behavior after replacing vulnerable code with

fixes. The proposed system is designed for PHP applications[9]. The system has been experimented with number of synthetic code in which vulnerabilities have been induced purposely.

6. IMPLEMENTATION

The approach can be implemented as a sequence of steps.

1. Taint analysis: The job of this module is to parse the source code. This also gives the trees which describe more about candidate vulnerable control flow paths. If the variables are not checked properly they may lead to development of vulnerabilities. So the variables need to check before reaching the sensitive sinks. Input to the module of taint analysis is a PHP source code and the output of this module will be the

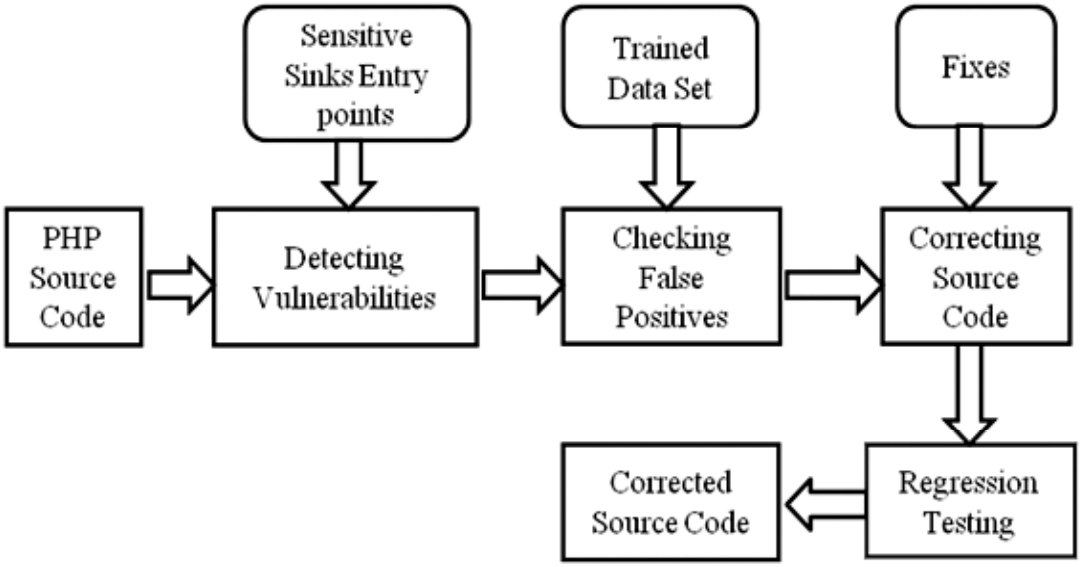

Figure 2: Architecture of Proposed System.

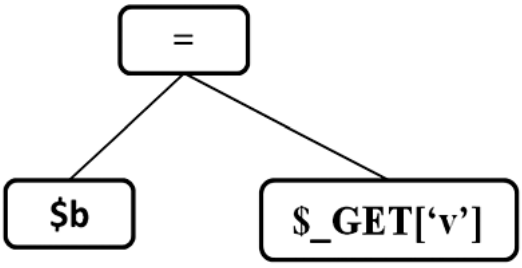

Figure 3:Abstract Syntax Tree

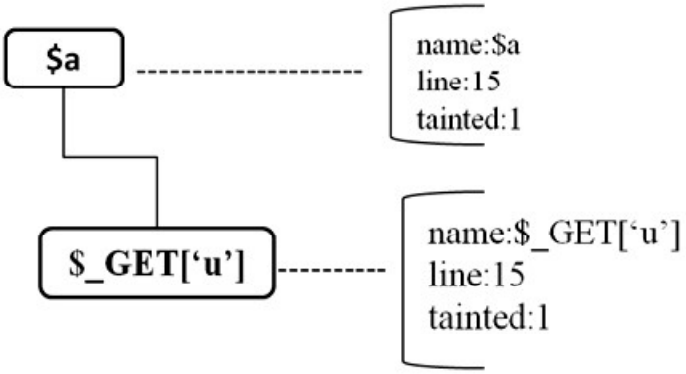

Figure 4:Tainted Symbol Table and Taint analysis on variables

candidate vulnerabilities. The first step will be parsing of the source code which will generate an Abstract Syntax Tree i.e. AST. Lexer and Parser will do the job of creating AST. In the Fig 3 shows the Abstract Syntax Tree for $b=$_GET['v']. All the variables that act as entry points are marked tainted in the beginning. A symbol table will be generated which will have tainted variables. Taint analysis travels through this Tainted Symbol Table. If a variable is marked as tainted then the symbols depending on this variable are checked. Fig 4 shows the Tainted Symbol Table for specific symbols having name, line number, and tainted flag as its variables. In this way, all the candidate vulnerabilities are marked which will be checked by a false positive predictor to make sure about the real vulnerability.

2. *Data mining*: Using data mining we will obtain the attributes from the candidate vulnerable control-flow paths. Here data mining will be actually used to confirm the presence of false positives. This will be done with the help of classifiers. If the presence of false positives is confirmed, further processing will be done with the help of induction rules.

3. *Code correction*: After detecting vulnerabilities and checking it for false positives each real vulnerability is removed by correction of its source code. This module for the type of vulnerability selects the fix that removes the vulnerability and signalizes the places in the source code where the fix will be inserted. Then, the code is corrected with the insertion of the fixes and new files are created. Fixes are small pieces of the code (small PHP functions developed to the effect) that performing sanitization or validation of the user inputs, depending of the vulnerability type. Our approach involves automatic code correction using K-NN algorithm after the detection of the vulnerabilities by taint analyzer and the data mining component.

4. *Feedback*: Feedback will be provided by the programmer at the end of testing module.It will be based on experience of the client.Experience will be based on data collected from vulnerable paths , vulnerabilities , fixes , false positive probability and the attributes that classified it as a false positive.Feedback will help programmer to avoid the same mistakes.

5. *Testing*: In this module testing will be performed on the corrected code to check for more bugs.We will perform manual testing on the source code.In manual testing ,testing is done without using automation tools.Test cases are executed manually.Different manual testing tools are Selenium ,QTP, Jmeter, Loadrunner.

7. ALGORITHMS USED

The algorithms which will be used in implementing the proposed system are Logistic Regression, Naïve Bayes (NB) and K-Nearest Neighbor (KNN). The other algorithms that will be used are random tree and random forest classifier.

8. CONCLUSION

The following paper put forwards an approach for finding and amending vulnerabilities within the web applications, and a tool that accomplishes the approach for PHP programs and input validation vulnerabilities. A blend of two systems: static source code examination and data mining is utilized for the approach and the device search for vulnerabilities. To recognize false positives, Data mining is incorporated which utilizes the main three machine learning classifiers, and an induction rule classifier, to legitimize their occurrence. After an intensive comparison between several alternatives all classifiers were chosen. The static examination issue is undecidable and depending on data mining can't evade this undecidability, however it just gives probabilistic consequences. The tool is used to correct the code by embedding fixes, comprising sanitization and validation functions. Testing is utilized to check if the fixes surely evacuate the vulnerabilities and do not compromise the (correct) conduct of the applications. The tool became explored by way of the usage of synthetic code with vulnerabilities embedded on motive, and with a widespread variety of open source PHP applications. It turned into additionally contrasted with source code analysis tools: Pixy, and PHP

MinerII. This evaluation proposes that the device can discover and correct the vulnerabilities of the classes it is programmed to deal with. it may find out 388 vulnerabilities in 1.4 million strains of code. Its exactness and accuracy were round 5% advanced to PHP MinerII's, and 45% superior to Pixy's.

REFERENCES

[1] L. K. Shar "Predicting common web application vulnerabilities from input validation and sanitization code patterns", Automated Software Engineering (ASE), 27th IEEE/ACM International Conference, 2012.

[2] N. L. de Poel, "Automated security review of PHP web applications with static code analysis," ACM 23rd international conference on World wide web, 2014.

[3] L. K. Shar and H. B. K. Tan, "Automated removal of cross site scripting vulnerabilities in web applications," Journal of Information and Software technology, Vol 54, Issue.5, May 2012.

[4] Y.W. Huang , "Web application security assessment by fault injection and behavior monitoring,", ACM 12th international conference on World Wide Web, 2003.

[5] Mounika B, A. Krishna Chaitanya, "Survey on Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis", International journal of Mathematics and Computer Research,Volume 1, Issue 2 March 2013.

[6] Y.W. Huang , "Securing web application code by static analysis and runtime protection,", ACM 1-58113-844-X/04/ 0005, WWW 2004.

[7] L. K. Shar, H. B. K. Tan, "Mining SQL injection and cross site scripting vulnerabilities" in International Conference on Software Engineering, 2012.

[8] Sonam Panda, Ramani S, "Protection of Web Application against SQL Injection Attacks", IJMER Vol 3, Issue.1, Jan-Feb 2013

[9] Iberia Medeiros, Numo Neves, "Detection of web application vulnerabilites using static analysis.", IEEE transaction on relaibility.

[10] Symantec, Internet threat report. 2012 trends, vol. 18, Apr. 2013.

[11] Ashwani Garg, Shekhar Singh, "A Review on Web Application Security Vulnerabilities" IJARSCE, Volume 3, Issue 1, January 2013.