Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 3, March 2016, pg.173 – 178

Collaborative Framework for Testing Web Application Vulnerabilities Using STOWS

R. Meena, Dr. R. Kalpana

Department of Computer Science and Engineering, IFET College of Engineering, Villupuram

Abstract- A major reason for vulnerabilities in web application is lack of adequate knowledge about secure coding. An approach to find these vulnerabilities is to use source code static analysis, but these tools tends to report many false positive. These problems are addressed by using a framework of collaborative testing in which test tasks are completed through the collaboration of various test services that are registered, discovered, and invoked at runtime using the ontology of software testing STOWS. The composition of test services is realized by using test brokers, which are also test services, but specialized in the coordination of other test services for vulnerabilities. The ontology can be extended and updated through an ontology management service so that it can support a wide range of test activities, methods, techniques, and types of software artifacts. This approach presents a prototype implementation of the framework in semantic WS and demonstrates the feasibility of the framework by running examples of building a testing tool as a test service, developing a service for test executions of a WS.

Keywords: Vulnerabilities, Secure coding, false positives, STOWS, framework, software security, static analysis, web applications.

I. INTRODUCTION

A research effort on security in web applications has been going on for many years, its security continues to be a problem. A major reason for this is that many programmers do not have adequate knowledge about secure coding, so they leave applications with vulnerabilities[2]. An approach to solve this problem is to use source code static analysis[1] to find these vulnerabilities, but these tools are known to report many false positives that make the hard task of correcting the application[1]. We explore the use of a combination of methods to discover vulnerabilities in source code with fewer false positives[1]. Addressing these problems, this paper proposes a framework of collaborative testing[3] in which test tasks are completed through the collaboration of various test services that are registered, discovered, and invoked at runtime using the ontology of software testing STOWS[3].

The composition of test services is realized by using test brokers, which are also test services and it is specialized in the coordination of other test services. The ontology can be extended[3] and updated through an ontology management service. So that it can support a wide open range of test activities, methods, techniques, and types of software artifacts[3]. The paper presents a prototype implementation of the framework in semantic WS and demonstrates the feasibility of the framework by running examples of building a testing tool as a test service, developing a service for test executions of a WS[9], and composing existing test services for more complicated testing tasks. The existing system combines the static to report analysis with data mining. Static analysis is an effective mechanism to find vulnerabilities[2] in source code[1], but tends many false positives (non-vulnerabilities) due to its uncertainity.

These problems are addressed by using a framework of collaborative testing in which test tasks are completed through the collaboration of various test services that are registered, discovered, and invoked at runtime using the ontology of software testing STOWS[3]. The composition of test services is realized by using test brokers, which are also test services, but specialized in the coordination of other test services for vulnerabilities.

II. RELATED WORKS

Web Services: Beyond Component-Based Computing[5]

In this paper, the different facets of Web services are identified and a flexible approach to engineering complex Web services is adopted in the form of a proposed framework for the development of Web services. After the examination of its main constituent parts, it is argued that its full potential and that of Web service engineering in general, is realized through the gradual formation of a rich service grid offering value-added supporting functionality and therefore the main desirable properties of such a service grid are highlighted. Furthermore, it introduces the concept of service grids as the necessary infrastructure that will enable Web services to transform the Web from a collection of information into a distributed computational entity, and identifies open matters and current technical challenges for the Web services co`mmunity in association with the proposed framework.

Testing Web Services: A Survey[4]

The Service-Oriented Computing (SOC) paradigm is allowing computer systems to interact with each other in new ways. According to the literature, SOC allows composition of distributed applications free from their platform and thus reduces the cost of such compositions and makes them easier and faster to develop. Currently web services are the most widely accepted service technology due to the level of autonomy and platform-independency provided by them. However, web services also bring challenges. For example, testing web services at the client side is not as straightforward as testing traditional software due to the complex nature of web services and the absence of source code. This paper surveys the previous work undertaken on web service testing[10], showing the strengths and weaknesses of current web service testing strategies and identifying issues for future work.

Generating Test Cases for Web Services Using Data Perturbation[7]

Web services have the potential to dramatically reduce the complexities and costs of software integration projects. The most obvious and perhaps most significant difference between Web services and traditional applications is that Web services use a common communication infrastructure, XML and SOAP, to communicate through the Internet. The method of communication introduces complexities to the problems of verifying and validating Web services that do not exist in traditional software. This paper presents a new approach to testing Web services based on data perturbation. Existing XML messages are modified based on rules defined on the message grammars, and then used as tests. Data perturbation uses two methods to test WS: data value perturbation and interaction perturbation. Data value perturbation modifies values according to the datatype.

Interaction perturbation classifies the communication messages into two categories: RP Communication and data communication. At present, this method is restricted to peer-to-peer interactions.

Testing Web Services by XML Perturbation[6]

The extensible Markup Language (XML) is widely used to transmit data across the Internet. XML schemas are used to define the syntax of XML messages. A receiving application must either validates XML messages, process the data in the XML message without validation, or modify the XML message to ensure that it conforms to the XML schema. A problem for developers is how well the application performs the validation, data processing, and, when necessary, transformation. This paper describes and gives examples of a method to generate tests for XML-based communication by modifying and then instantiating XML schemas.

III. PROPOSED SYSTEM

Finding and correcting vulnerabilities in web applications, and a tool that implements the approach for PHP programs and input validation vulnerabilities. We proposed the approach and the tool search for vulnerabilities using a combination of two techniques: static source code analysis, and data mining. Static analysis tools automate the auditing of code, either source, binary or intermediate. We propose an approach for improving the security of web applications by combining detection and automatic correction of vulnerabilities in web applications. In this paper, we present a framework for collaborative testing in which testing activities are accomplished through interactions among multiple participants. We employ the ontology of software testing STOWS to describe the capabilities of T-services and test tasks for the registration, discovery, and invocation of T-services. The knowledge intensive composition of T-services are realized by the development and employment of test brokers, which are also T-services. Our results suggest that the tool is capable of finding and correcting the vulnerabilities from the classes it was programmed to handle

IV. SYSTEM IMPLEMENTATION

Taint analysis

Parsing the source code, generating an abstract syntax tree (AST), doing taint analysis based on the AST, and generating trees describing candidate vulnerable control-flow paths (from an entry point to a sensitive sink). For every sensitive sink that was found to be reached by tainted input, it tracks the path from that sink to the entry point using the tables and trees just mentioned. Along the track paths, the vectors of attribute are collected and classified by the data mining algorithm as true positive, or false positive. Note that we use the terms true positive and false positive to express that an alarm raised by the taint analyzer is correct or incorrect. These terms do not mean the true and false positive rates resulting from the data mining algorithm, which measure its precision and accuracy. If we could track the user inputs and verify if they reach some functions, then we could detect the vulnerability.

Data Mining

Obtaining attributes from the candidate vulnerable control-flow paths, and using the top 3 classifiers to predict if each candidate vulnerability is a false positive or not. In the presence of a false positive, use induction rules to present the relation between the attributes that classified it. A set of attributes that characterize the positive-string manipulation, validation, manipulation in SQL queries. We define a process to evaluate machine learning classifiers to choose the best that classify our instances with high accuracy and precision.

Code Correction:

Given the control-flow path trees of vulnerabilities (predicted not to be false positives), identifying the vulnerabilities, the fixes to insert, and the places where they have to be inserted; assessing the probabilities of the vulnerabilities being false positives; and modifying the source code with the fixes. The code correct or picks the paths classified as true positives to signal the tainted inputs to be sanitized using the tables and trees mentioned above. For that purpose, our machine learning approach allows us to identify combinations of attributes that are correlated with the presence of false positives, i.e., what attributes justify the classification of false positives. To show this correlation, we use induction or coverage rules for classifying the instances, and for presenting the attributes combination to that classification.

Feed Back & Predictor

Providing feedback to the programmer based on the data collected in the previous steps Predicts if a candidate vulnerability is a false positive or a true vulnerability. It reports the real vulnerabilities detected and how they were corrected. Finally, outputs a corrected version of the web application and reports the fewer false positives predicted.

Testing

Higher assurance can be obtained with two forms of testing, specifically program mutation to verify if the fixes do their function, and regression testing to verify if the behavior of the application remains the same with benign inputs. This verification is done by checking if these test cases produce incorrect or unexpected behavior or outputs. We employ the ontology of software testing STOWS to describe the capabilities of T-services and test tasks for the registration, discovery, and invocation of T-services. The knowledge intensive composition of T-services are realized by the development and employment of test brokers, which are also T-services.

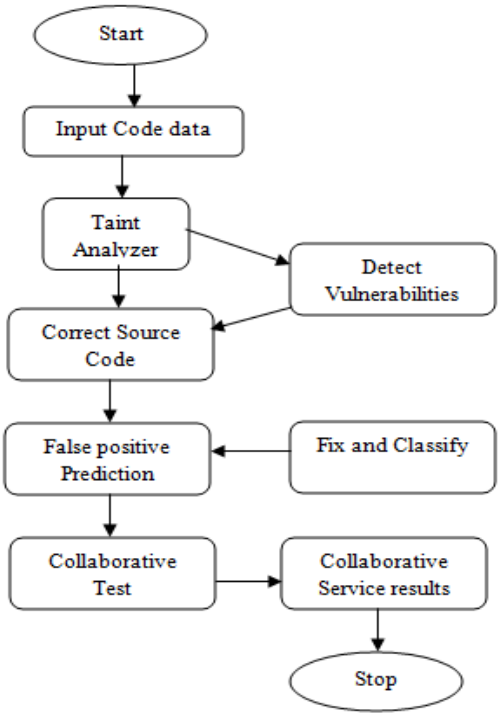

Figure A

V. RESULTS

The following figure B represents the page which contains the login for the user. The data about the user are stored in the database which is connected to the login code. Once the user had logged in, the user is authorized and redirected to the corresponding request page.
Figure C denotes one of request pages.

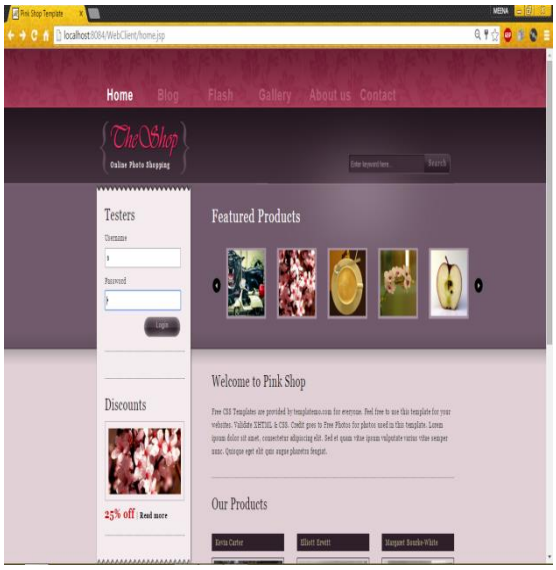

Figure B

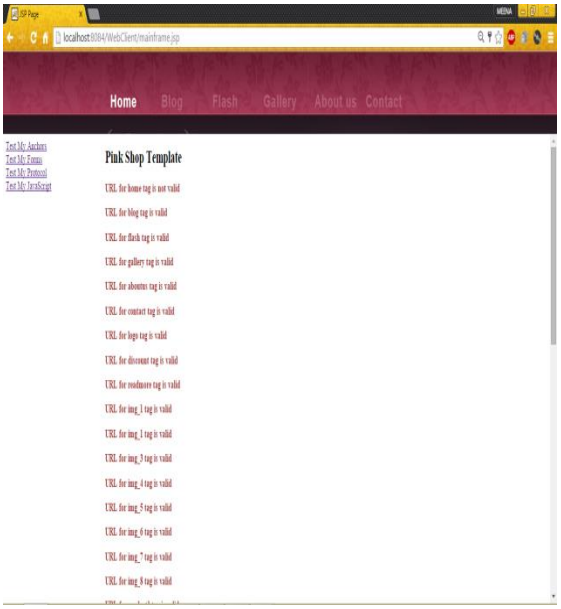

Figure C

VI. CONCLUSION

This paper presents an approach for finding and correcting vulnerabilities in web applications. The tool search for vulnerabilities using collaborative framework for testing web applications. Data mining is also used to identify false positives using the top 3 machine learning classifiers, and to justify their presence using an induction rule classifier. All classifiers were selected after a thorough

comparison of several alternatives. It is important to note that this combination of detection techniques cannot provide entirely correct results. This problem is undecidable, and resorting to data mining cannot circumvent this undecidability, but only provide probabilistic results. The tool corrects the code by using a framework of collaborative testing in which test tasks are completed through the collaboration of various test services that are registered, discovered, and invoked at runtime using the ontology of software testing STOWS. It presents a prototype implementation of the framework in semantic WS and demonstrates the feasibility of the framework by running examples of building a testing tool as a test service, developing a service for test executions of a WS, and composing existing test services for more complicated testing tasks.

REFERENCES

1. Ibéria Medeiros, Nuno Neves and Miguel Correia, "Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining" vol. 65,pp/ 54 – 69, 2016.
2. Kalpana.R and J.Sowmiya "Detecting Malware by Signature Distribution Algorithm in MANET Using Heterogeneous Devices" International Journal of Computer Science and Mobile Computing ISSN:2320-088x, volume:4, 3 March 2015.
3. Hong Zhu and Yufeng Zhang ,"A Test Automation Framework for Collaborative Testing of Web Service Dynamic Compositions",Vol. 5, no. 1, pp: 116 - 130, 2012.
4. M. Bozkurt, M. Harman, and Y. Hassoun, "Testing Web Services: A Survey," Technical Report TR-10-01, Dept. of Computer Science, King's College London, Jan. 2010.
5. M. Stal, "Web Services: Beyond Component-Based Computing," Comm. ACM,vol. 45, no. 10, pp. 71-76, Oct. 2002.
6. W. Xu, J. Offutt, and J. Luo, "Testing Web Services by XML Perturbation, "Proc. IEEE 16th Int'l Symp. Software Reliability Eng. (ISSRE '05),pp. 257-266, Nov. 2005.
7. J. Offutt and W. Xu, "Generating Test Cases for Web Services Using Data Perturbation", SIGSOFT Software Eng. Notes, vol. 29, no. 5, pp. 1-10, Sept. 2004.
8. G. Canfora and M. Penta, "Service-Oriented Architectures Testing: A Survey, "Software Eng.: Int'l Summer Schools (ISSSE 2006-2008),Revised Tutorial Lectures, A. Lucia and F. Ferrucci, eds., pp. 78-105, Springer-Verlag, 2009.
9. X. Bai, W. Dong, W. Tsai, and Y. Chen, "WSDL-Based Automatic Test Case Generation for Web Services Testing, "Proc. IEEE Int'l Workshop Service Oriented System Eng. (SOSE '05),pp. 215-220, Oct.2005.
10. F. McCabe, E. Newcomer, M. Champion, C.Ferris, and D.Orchard, Web Services Architecture, W3C Working Group Note, http://www.w3.org/TR/ws-arch, 2004.

