Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey on Ajax based Web Application Testing

Pallavi P. Patil*, Prof. P. D. Lambhate
Computer Department,
SPPU, India

Abstract— AJAX enables web browser to send asynchronous requests to the web server and generate the response in the background, without reloading the client page. Ajax based web applications are stateful, event-based. Dom tree manipulation is performed dynamically at the client-side. These features make AJAX based applications different from traditional web applications. Such web applications introduce new challenges with new security vulnerabilities, and their nature makes it very difficult or impossible to test them automatically with the current web-application security tools. This paper discusses various techniques used to test modern web applications. It also focuses on the security testing of the applications.

Keywords— AJAX, DOM, Testing, Security, Vulnerability

I. INTRODUCTION

Today web-based applications are becoming more ubiquitous, and among these applications, a new trend is emerging: rich Internet applications (RIAs), using technologies such as Ajax, Flex, or Silverlight, break away from the traditional approach of Web applications having server-side computation and synchronous communications between the web client and servers. Such web applications introduce new challenges with new security vulnerabilities, and their nature makes it very challenging to test them.

Ajax is one of the key technologies used to develop today's web applications. AJAX is "Asynchronous JAVASCRIPT and XML". AJAX, facilitates user navigation through a sequence of HTML pages as well as dynamic rich interaction using graphical user interface (UI) components. Ajax provides better user-friendliness and interactiveness to the web applications. However ajax makes the applications error-prone and vulnerable due to their some features like asynchronous, stateful and event-based nature, the use of JAVASCRIPT, manipulation of the browser's Document-Object Model (DOM) at the client-side and delta communication between client and web server. Use of various testing techniques can improve the dependability of ajax applications. However traditional techniques are not enough to recognize the dynamic dependencies found in today's web applications. Furthermore, Testing modern web applications for security vulnerabilities is far from trivial and challenging [1]. Detecting and eliminating the vulnerabilities needs the deep understanding of these vulnerabilities. It is important to know what makes the application vulnerable, what loop holes need to be corrected to make the application free from these vulnerabilities, what alternatives can be further used so that the risk of being vulnerable to attacks can be reduced and many more. Different techniques have been deployed to detect these vulnerabilities and appropriate steps are taken. Strategies such as static analysis, attack graph generation and its analysis, usage of vulnerability scanners are some of them. However, these are not adequate for Ajax applications.

II. WEB APPLICATION TESTING TECHNIQUES

Several techniques have been presented in the literature to support testing of Web applications. It includes static analysis, model based approach, capture and replay.

A. Static Analysis

Static analysis focuses on the analysis of program structure using various techniques. The flaws are detected by analyzing the code of the program. Static analysis is done using techniques like lexical analysis, type inference, constraint analysis and many more. Lexical analysis emphasize on the semantics of the program structure; each program module is compared with the library of loophole to detect flaws present in the system. Type inference is related to the data type rules for the variable. It determines whether the variables used in the program are in sync with the type to which they relate. Constraint analysis is a two-step process. It involves- constraint generation and constraint solution [2].

Avancini and M. Ceccato [3] proposed an approach to generate test cases that cover Cross-Site Scripting vulnerabilities on web application. The approach focuses on PHP. This kind of security threats is very general, as it affects a large number of web applications, written in any programming language. As generation of test cases relies on static analysis, it suffers its limitations.

Static analysis techniques have some demerits associated with them. If an unknown vulnerability is detected, then it is not possible to compare it with the predefined loophole library for its validation [2]. Moreover Static analysis techniques are not able to reveal many of the dynamic dependencies present in today's web applications hence not suitable to test Ajax applications [1].

B. Model Based Approach

Title In this approach is based on reverse engineering techniques. Web crawlers are used to infer the Web model of the application. The web model is a graph in which each Web page is represented as a node and each link (e.g., HTML links, automatic redirections, submits) is represented by an edge. Test cases are generated by navigating the navigating the Web model of the application. Ricca and Tonella [4] proposed a model-based testing approach for web applications. ReWeb: a tool for creating a model of the web application in UML is introduced by them. Andrews et al. [5], proposed another approach using finite state machine based on constraints defined by the tester.

Security vulnerabilities are related to security functionalities at the application level and sensitive to implementation details as well. Thus traditional model-based approaches which elide implementation details are by themselves inadequate for testing security vulnerabilities. Krishnan, K. Ross and P.Salas [6] proposed a framework that retains the advantages of model based testing that exposes only the necessary details relevant for vulnerability testing. Tools such as WAVES [7] and SecuBat [8] have been proposed to automatically assess security of web application. The server response is analyzed to determine vulnerable parts of the web application. Session-based testing techniques are merely focused on synchronous requests to the server and lack the complete state information required in AJAX testing.

However, all these model-based testing techniques are targeted towards classical multipage web applications and use traditional crawlers. The traditional web crawlers are unable to crawl AJAX applications [24].

C. Capture and replay

The server side of AJAX applications can be tested with any conventional testing technique. On the client side, testing can be performed at different levels. Unit testing tools such as JsUnit can be used to test JAVASCRIPT on a functional level [1]. The most commonly used testing tools support the capture and replay of particular user actions. In this approach, user interactions with the application interface of the Web application under testing are recorded and then repeated during the testing phase. The most commonly used AJAX testing tools are currentlycapture/replay tools such as Selenium IDE, WebKing, and Sahi, which allow DOM-based testing by capturing events fired by user interaction. Capture and replay can be used to test AJAX-based Web applications, since it runs the application from a user point-of-view using the GUI. Xelenium is a tool developed to test security vulnerabilities of Ajax web applications. It is based on Selenium. The Capture and Replay tools requires manual efforts at tester side to exercise client-side functionality and manage test suites.Several implementations of this technique would need to be improved to be effectively used to test AJAX applications.

III. STATE OF THE ART

A. State Based

A state based testing approach is proposed by Marchetto [9]. To test an AJAX application effectively, the states of client side components must be considered during testing phase. State based testing technique for AJAX starts with the analysis of all the states that can be dynamically generated by the client-side user interactions during execution of the application. AJAX allows changing of HTML elements dynamically according to the user interactions. The different states of a AJAX web application are characterized by HTML elements of a web page. The finite state model of the web application is built using values of corresponding HTML elements. State based testing is a powerful technique and can reveal faults which can not be detected using traditional techniques. Marchetto proposed use of traces of the application to generate a finite state machine. In this technique finite state machine for a given AJAX application is dynamically extracted. However the dynamic analysis was partial as it requires manual refinement efforts for model extraction. It is quite challenging to explore dynamic extraction of states. Thus a proper approach is required to construct a model using automatic dynamic analysis. Execution traces can be traced using log files generated by real user interaction. Traces contain information about Dom states and call back. Marchetto proposed use of state abstraction function to avoid state explosion problem which may occur due to huge no of concrete states. Figure 1 shows FSM obtained from the traces using the state abstraction function [9].

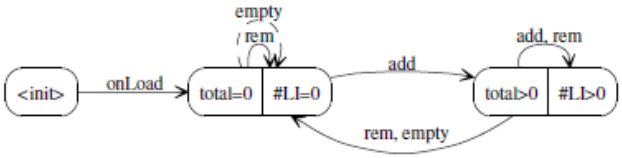

Fig 1: FSM Generated for Cart Application

In this work, search-based algorithm is used to generate event sequence of various lengths. A FSM based testing technique may result in State Explosion Problem due to generation of high number of test cases. To avoid this problem Marchetto's work used notion of semantically interacting Events and abstraction function. However, the work is mainly focused on minimizing test cases by means of semantically interacting events. It minimizes only few test cases of asynchronous communication. It is not effective for test case minimization for other AJAX features. FSM recovery needs improvement, in order to automatically infer proper abstraction function [7].

B. Invariant Based

Static analysis techniques are unable to test modern rich web application effectively due to their dynamic behaviour. It is very challenging to generate test cases for dynamic run time interaction of a web application. Mesbah proposed

an ―Invariant based Automatic testing of AJAX user interface. In this approach a model of the user interface (UI) states of an A JAX application are automatically derived. The model is derived by "crawling" an AJAX application automatically using CRAWLJAX1 tool. All the client-side UI functionality like clicking buttons and other UI elements are automated. . It simulates the real user events on the user in-terface and generates the abstract model in the form of state flow graph [1]. The paths discovered during crawling process are used to generate test cases automatically. In this work the failures in the executions, are recognized by means of invariants : properties of either the client-side DOM-tree or the derived state machine that should hold for any execution. The work defines two types of invariants: generic (applicable to any web application) or application-specific (constraints specific to the application). Fig. 2 shows a state flow graph generated using crawl ajax [1]

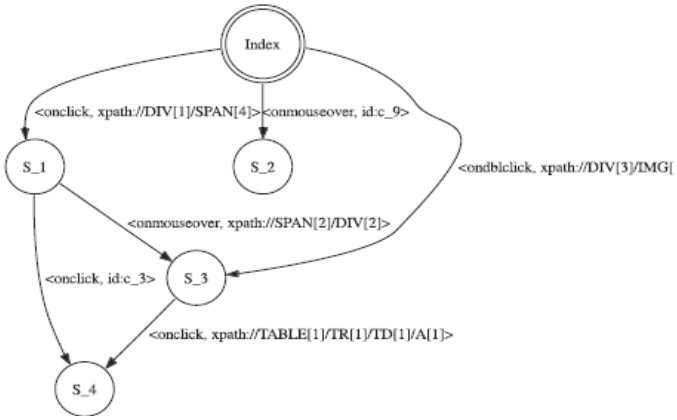

Fig 2: A state flow graph generated using Crawlajax

IV. CONCLUSIONS

The New technologies used in modern Web applications, such as AJAX, improve the interactiveness and usability of a Web Application. However ajax makes the applications error-prone and more vulnerable due to its some features like asynchronous, stateful and event-based nature, the use of JAVASCRIPT, manipulation of the browser's Document-Object Model (DOM) at the client-side and delta communication between client and web server. Ajax poses new challenges in testing of modern web applications. In this paper we have discussed the techniques used to test ajax based web applications for functionality as well as security. All the approaches have their own benefits and drawbacks. But there is no approach which tests Ajax application perfectly and checks them for security vulnerabilities. Here we have basically surveyed the existing approaches which will help us in future to design a new approach for testing the modern Ajax based Web applications.

REFERENCES

[1] Mesbah, A.,Van Deursen, *Invariant-based automatic testing of Ajax user interfaces*, 2nd ed., In Proceedings of the 31st International Conference on Software Engineering (ICSE'09), IEEE Computer Society, 2009, pages 210-22.

[2] Peng Li and Baojiang Cui, December, 2010, *A Comparative Study on Software Vulnerability Static Analysis Techniques and Tools*, in Proceedings of the IEEE International Conference on Information Theory and Information Security (ICITIS), pp.521-524.

[3] A. Avancini and M. Ceccato, *Security Testing of Web Applications: A Search-Based Approach for Cross-Site Scripting Vulnerabilities*, in 2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation, 2011, pp. 85–94.

[4] F. Ricca and P. Tonella, *Analysis and Testing of Web Applications* , Proc. 23rd Int'l Conf. Software Eng., pp. 25-34, 2001R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[5] A. Andrews, J. Offutt, and R. Alexander, *Testing Web Applicationsby Modeling with FSMs* , Software and Systems Modeling, vol. 4, no. 3, pp. 326-345, July 2005.

[6] P. Krishnan, K. Ross and P.Salas, *Model-based Security Vulnerability Testing*, ASWEC 2007. 18th Australian Conf. Software Eng., pp. 284-296, 2007.

[7] Y.W. Huang, C.H. Tsai, T.P. Lin, S.K. Huang, D.T. Lee, and S.Y. Kuo, *A Testing Framework for Web Application Security Assessment*, J. Computer Networks, vol. 48, no. 5, pp. 739-761, 2005.

[8] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, *Secubat: A Web Vulnerability Scanner*, Proc. 15th Int'l Conf. World Wide Web, pp. 247-256, 2006..

[9] Marchetto, A., Tonella, P., and Ricca, F. (2008b). *State-based testing of Ajax web applications*. In Proc. 1st IEEE Int. Conference on Sw. Testing Verification and Validation (ICST'08), pages 121-130. IEEE Computer Proceedings of the 23rd International Conference on Software Engineering, pages 25-34,Society.

