International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Crawler based Ajax Web Application Testing

Pallavi Patil†* and Poonam Lambhate‡

†Computer Department, SPPU, Pune, India
‡IT Department, SPPU, Pune, India

Accepted 27 July 2015, Available online 28 July 2015, **Vol.5, No.4 (Aug 2015)**

Abstract

New technologies like AJAX are used today to develop Web applications which support asynchronous interactions with the server, going beyond the traditional submit/wait-for-response method. AJAX improves the responsiveness and usability of a Web application but give rise to new challenges for testing. Ajax makes web applications stateful, event-based. Ajax allows browser to make asynchronous request to the web server. Dynamic manipulation of a DOM tree is performed at the client side. This differentiates Ajax based applications from traditional web sites. Thus modern web applications pose new challenges with new security vulnerabilities, and their behaviour makes it difficult or impossible to test them automatically using the existing web-application testing tools. We propose a method to automatically test AJAX applications.

Keywords: *Testing, Web application, ajax, security*

1. Introduction

Rich Internet application (RIA) is the new trend in web technology. The modern web applications built using technologies such as Ajax, Flex, or Silverlight are different from the traditional 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 features make it very challenging to test them.

Ajax is one of the important technologies used to develop today's web applications. AJAX is "Asynchronous JAVASCRIPT and XML". AJAX, provides user navigation through a sequence of HTML pages and dynamic rich interactive graphical user interface (UI) components. Ajax provides better user-friendliness and interactiveness to the web applications. However Ajax bases web applications are vulnerable and error-prone due to some features of Ajax like asynchronous and event-based stateful nature, use of JAVASCRIPT, manipulation of the Document-Object Model (DOM) by browser at the client-side and delta communication between client and server. Use of various testing techniques can make the ajax applications more safe and dependable. However traditional web application testing techniques are not enough to capture the dynamic dependencies found in modern web applications. Furthermore, Testing modern web applications for security vulnerabilities is challenging and far from trivial (Mesbah *et al,* 2009). Detecting and removing the vulnerabilities needs the good understanding of these vulnerabilities. It is important to know what makes the application vulnerable, what flaws need to be corrected to make the application secure from these vulnerabilities, what alternatives can be further used so that the risk of being vulnerable to attacks can be minimized and many more. Many techniques have been deployed to detect these vulnerabilities. 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.

2. Literature Survey

Static Analysis

Static analysis is based on analysis of program structure using various techniques. The flaws are detected using analysis of the code. Static analysis is performed using techniques like type inference, lexical analysis and constraint analysis. Lexical analysis focuses on semantics of the program structure; each program module is compared with the loophole library to detect flaws present in the system. Type inference is based on the data type rules for the variable. It checks whether the variables used in the code are in sync with their related type. Constraint analysis involves two-steps. It includes constraint generation and constraint solution (Peng *et al*, 2010).

*Corresponding author: **Pallavi Patil**

Avancini (Avancini *et al*, 2011). proposed an approach to generate test cases that cover Cross-Site Scripting vulnerabilities on web application. The approach is targeted to PHP. XSS is very general security threat, as it affects a large number of web applications, written in any programming language. However generation of test cases is based on static analysis, it suffers through its limitations.

Static analysis techniques have some drawbacks associated with them. If a new vulnerability is detected, then it is not possible to compare it with the existing predefined library of loophole for its validation (Peng *et al*, 2010). . Moreover Static analysis techniques are unable to capture many of the dynamic dependencies present in modern web applications hence unable to test Ajax applications (Mesbah *et al*, 2009). .

Model Based Approach

This approach use reverse engineering techniques. Web crawlers are used to build the Web model of the application. The model is a graph which consists of nodes and edges representing web pages and transitions (e.g., HTML links, automatic redirections, submits) among them respectively. Test cases are generated by traversing the Web model of the application. Ricca and Tonella (Ricca *et al*, 1997). proposed ReWeb: a tool for creating a model of web application in UML using a model-based testing approach. Another approach is proposed by Andrews *et al*. (Andrews *et al*, 2005). , using finite state machine based on constraints defined by the tester.

Security vulnerabilities are related to security flaws at the application level. It is sensitive to implementation details as well. Thus traditional model-based approaches are inadequate for testing security vulnerabilities as they elide implementation details. Krishnan, K. Ross (Krishnan *et al*, 2007). proposed a framework for web application testing that retains the advantages of model based testing that reveals only the necessary details requred for vulnerability testing. Tools like WAVES (Huang *et al*, 2005). and SecuBat (Kals *et al*, 2006). automatically assess web application security. The server response is analyzed to determine vulnerable area of the web application. Session-based testing techniques lack the complete state information required in AJAX application testing.as it merely focus on synchronous requests to the server

Furthermore, all the model-based testing techniques are designed for classical multipage web applications and incorporate traditional crawlers. However the traditional web crawlers are inadequate to crawl AJAX applications (Peng *et al*, 2010). .

Capture and replay

The server side of AJAX applications can be tested with any conventional testing technique. At the client side, testing can be done at various levels. Unit testing tools e.g. JsUnit are capable of testing JAVASCRIPT on a functional level (Mesbah *et al*, 2009). . The most common testing tools support the capturing and replaying of user's actions. In this technique, the interactions of user with the application interface of the target Web application are recorded and then replayed during the testing phase. Today the most popular AJAX testing tools are capture/replay tools like Sahi, Selenium IDE and WebKing, which allow testing by capturing events generated by user interaction. Capture and replay enables testing AJAX-based Web applications, as it executes the application from a user point-of-view through the GUI. Another tool Xelenium is developed to check security vulnerabilities of web applications using Ajax. It is based on Selenium. The Capture and Replay tools need manual efforts at tester side to execute client-side functionality and manage test suites. Several implementations based on this technique need improvement to effectively used for testing modern AJAX applications.

State Based

A state based approach is proposed by Marchetto (Marchetto *et al*, 2008). for testing. Testing an AJAX application effectively requires consideration of the states of client side components during testing phase. State based testing technique for AJAX begins with the analysis of all the dynamically generated states resulting from client-side user interactions during execution of the application. AJAX allows to change HTML elements dynamically based on the user interactions. The various states of an AJAX web application are specified by HTML elements of a web page. States in the finite state model of the web application is built with different values of corresponding HTML elements. State based testing is a powerful approach and can detect faults which traditional techniques are unable to reveal. Marchetto used traces of the application to infer a finite state machine. In this approach, FSM for a given AJAX application is dynamically extracted. However, the dynamic analysis is partial as the model extraction needs manual refinement efforts. It is far from trivial to explore dynamic extraction of states. Thus a proper method is required to build a model by automatic dynamic analysis. Execution traces can be traced with the help of log files generated using real user interaction. Traces include information regarding DOM and call back. Marchetto used state abstraction function to avoid state explosion which may result from large number of concrete states.

However, the work mainly aims at minimizing test cases with the help of semantically interacting events. It is not very useful for test case minimization of other AJAX features. FSM recovery needs to be improved, in order to automatically infer effective abstraction function (Huang *et al*, 2005). .

Invariant based

Static analysis techniques are inadequate to test modern RIA effectively due to their dynamic behaviour.

It is very promising to derive test cases for dynamic interaction of a web application. Mesbah proposed use of Invariants to support automatic testing of AJAX user interface (UI). Invariants are used to detect failures in the executions. Two types of invariants are proposed: generic (which are true for any web application) or application-specific (constraints specific to the target web application). In this technique a model of the UI states of an A JAX application is automatically generated. The model is inferred by traversing an AJAX application automatically using CRAWLJAX tool. Web application's UI functionality at client side like button clicks, entering text in text fields and other UI elements are is performed automatically. It simulates the real user events of the UI and builds the abstract model of web application in the form of state flow graph (Mesbah *et al*, 2009). . The paths extracted during crawling phase are exploited to automatically generate test cases.

3. Proposed Work

The proposed approach is based on automatic crawling Ajax application to perform testing of the web application. The crawler exercises the client side UI functionality by automatically clicking buttons and other UI-elements. Invariants are used to recognize failures in the executions. Test cases check the invariants and conditions for security vulnerabilities. Invariants are properties of either the derived state machine or the client-side DOM tree that should hold during any execution. Algorithm 1 and 2 show the overall process. Crawler generates stateflow graph of the web application. To infer such a graph automatically, we open the web application in a web browser, we examine the DOM-tree looking for candidate elements to fire events on, and we detect user interface state changes. We conduct the analysis and navigation part recursively for all possible states. For input fields, we provide random data if no custom data is available. Our approach detect Sql injection vulnerability using secured database checking. It detects potential input points of SQL injection. Test cases are run to explore the vulnerabilities by making attacks.

The following sections discuss these steps in more detail.

Step 1: Begin
Step 2: Enter Ajax application URL for testing
Step 3: Initialize Browser, webdriver, state machine
Step 4: Find all candidate clickables
Step 5 : Generate event for every clickable
Step 6 : If new state found, update state flow graph
Step 7: Check dead click using invariant
Step 8: Check Expired click using invariant
Step 9 : Apply Sql injection detection algorithm
Step 10: Generate test case report

Sql Injection Detection Algorithm:

Step 1: list conditions C[]= {"Or1=1","1024",'anything' OR 'x'='x',"UNION ALL SELECT 1,null,null --" ,etc}
Step 2: Find all input text fields Inp[Step 3: for i ∈ Inp do
Step 4: for Cn ∈ C do
Step 5: input Sendkeys(Cn)
Step 6: Webdiver click on submit
Step 7: value = return value submit;
Step 8: if(value)
Step 9: Vulnerable to Sql injection attack
Step 10: else
Step 11: Not vulnerable to Sql injection attack
Step 12: end for
Step 13: end for

4. Results

The Fig1. shows UI to enter an url of an target Ajax web Application to test. Crawling starts with the entered Url.

Final result generated after execution of test cases is as shown in Fig 2. It shows the vulnerabilities found in the web application under test. The results shown here are obtained by applying the proposed method to an demo ajax web application. It detects deadclicks and expired links in the application. Sql vulnerability in the application is also revealed.

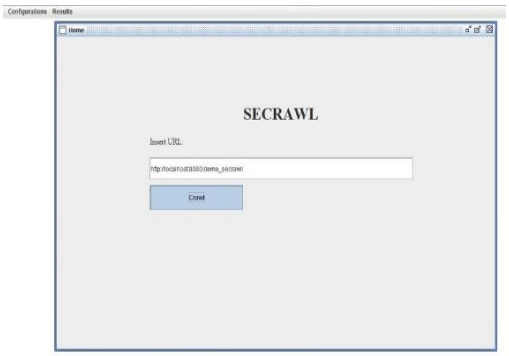

Figure 1 Form for entering URL

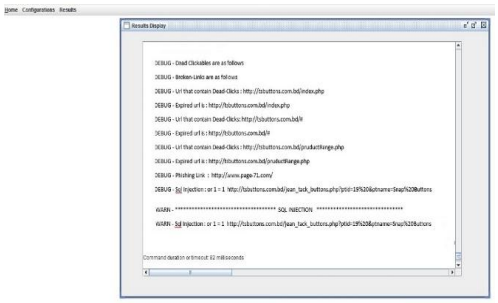

Figure 2 Test case results showing detected vulnerabilities

Conclusion

In this paper, we have proposed a method for automatically testing AJAX based web applications. Our work is based on an ajax application crawler: Crawljax.

DOM-tree invariants are used to detect AJAX-specific faults in the application. We extend the crawler to test Ajax web application for security vulnerabilities. The proposed approach checks target web application for SQL injection vulnerability. We see many opportunities of using the proposed tool in practice considering the growing popularity of AJAX applications and extending it to reveal more security vulnerabilities.

References

Mesbah, A.,Van Deursen, (2009) 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, pages 210-22.

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 onInformation Theory and Information Security (ICITIS), pp.521-524.

A. Avancini and M. Ceccato,(2011) 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, pp. 85–94.

F. Ricca and P. Tonella,(2001) Analysis and Testing of Web Applications , Proc. 23rd Int'l Conf. Software Eng., pp . 25-34,

A. Andrews, J. Offutt, and R. Alexander,(2005) Testing Web Applicationsby Modeling with FSMs , Software and Systems Modeling, vol. 4, no. 3, pp. 326-345.

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

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

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

Marchetto, A., Tonella, P., and Ricca, F. (2008). 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.