International Journal of Information and Computation Technology.
ISSN 0974-2239 Volume 4, Number 5 (2014), pp. 445-452

http://www. irphouse.com /ijict.htm

Finding Execution Faults in Dynamic Web Application

C.P. Shabariram

DPG Scholar, Department of CSE, Kathir College of Engineering, Wisdom Tree, Neelambur, Coimbatore, INDIA

Abstract

Pin down is a framework for root cause analysis on dynamic web application. Fault localization in dynamic web application is the problem of decisive where in the source code modifies has to be completed in order to fix the perceived failures. The cause of the failure is called as execution bug that also called as fault. The dynamic execution nature of the web application isolate the source cause of execution bug by various fault localization techniques. To identify execution bugs proficiently in web applications, some algorithms can be improved by using a comprehensive domain for conditional and function-call statements and using a source mapping but did not focus on server side vulnerability. This study use randomized input generation technique for dynamic web application to check whether the web page has vulnerable to SQL injection. An automated random input generation is constructed for each executable statement and to determine the execution failure such as a missing included file, an incorrect SQL query, or by an uncaught exception of the corresponding statement. In addition, determine HTML failures involve situations in which the generated HTML page is not syntactically correct according to an HTML validator to find HTML failures through checking appropriate tags with closing by parsing DOM tree and Less serious execution failures, like those caused by the employment of deprecated language constructs (like include & require function), produce obtrusive error messages but do not halt execution. In this study the result shows that code coverage improved from 90% to 100%. However the result indicates 100% coverage is a reliable indicator of the effectiveness of a test set.

Keywords: Pin down; Fault Localization; Execution bug; Random input generation; dynamic web application; SQL Injection; Execution failure; HTML failure; HTML validator; DOM tree.

1. Introduction

A specious declaration of data and functional steps in a program causes an unexpected behavior to execution. It is an intrinsic weakness of the design or implementation which might result in a failure. In program debugging, Localization is major method. It can be divided into two main components. The initial half is to spot apprehensive code by using various techniques. It may contain program bugs. The next part is for programmers to actually examine the identified code to decide whether it certainly contains bugs. In first part apprehensive code is prioritized based on its probability of containing bugs for reference all the Statistical localization techniques. Code have a lower precedence must be examined after code have a higher precedence. The next part, we assume that bug detection is perfect. So the programmers can forever correctly classify faulty code as faulty and non-faulty code like non-faulty. The amount of code desires to be inspected may increase when perfect bug discovery does not hold. There is a high insist for mechanical Execution bug identification techniques which can direct programmers to the locations of bugs with least human involvement. This insists has led to the suggestion and development of a variety of techniques over current years. While these techniques split similar goals, they can be moderately diverse from one another, and often stalk from ideas which themselves initiate from some diverse disciplines.

2. Fault Localization

Web applications are written in a mixture of some programming languages, like JavaScript in client side, PHP and Structure Query Language (SQL) in server side. In web application domain, pages are not displayed properly due to the deformed HTML or any embedded language errors. Such HTML and execution failures may be difficult to find because theses codes are generated dynamically. In this paper, execution fault locations are identified by using automated random input generation technique. Random input generation technique is constructed for each executable statement and to determine the execution failure such as a missing included file, an incorrect SQL query, or by an uncaught exception of the corresponding statement. In previous work, determined the failures based on the identifying inputs which cause an application hurtle. It did not address the problem. Statistical analysis between passing and failing tests for determine the faulty location. Further many statistical localization algorithms are used to find the locations. Using Tarantula [11], [12], Ochiai [13], Jaccard algorithms for finding the proportion of passing and failing tests in the execution of statements. Suspiciousness rating is calculated for each executed statement for forecast the location of the bug. The enhancements are presented in the statistical error identification techniques for improving efficiency of execution bug identification.

Source mapping method and extended domain are enhanced in previous work [14]. Extended domain technique applies to conditional statements helps execution bug identification. Suppose any statements are missing in HTML code or any default case not mention in switch case that time suspiciousness rating cannot be calculate. Condition modeling technique is used in this situation.

2.1 Source mapping and Extended Domain

Each statements of web application and output of every part are recorded. HTML Validator analyses these report and indicate which ingredient of the HTML output are improper. Present localization methodologies assume the existence of a test case. However, developers are often attempted with state of affairs where a failure takes place, but where no test suite is accessible that can be used for bug identification. To address such situations, we present an approach for generating test suites that can be used to localize bugs effectively. This approach is a variation on combined concrete and symbolic execution [9], [10] that is parameterized by a similarity criterion. Path constraint similarity and input similarity are increased statement coverage and path coverage. The quantity of similarity among the path constraints related with two executions. Based on the number of inputs the similarity among two execution is figured and it identical for both execution. Automated tool Apollo used for implementing these techniques. It increases the effectiveness of fault localization by direct test suites generation.

3. Pin Dowm Framework

3.1 Objective

The Fault localization algorithms explored in this paper attempt to predict the location of a bug based on randomized input values for server side execution in case of validation with empty inputs.

3.2 Overview of Pin down framework

In pin down framework architecture execution failures may be caused by a missing included file, an incorrect MySQL query, or by an uncaught exception. Such failures are easily identified as the PHP interpreter generates an error message and halts execution. The less serious execution failures, like those caused by the employment of deprecated language constructs, create prominent error messages but do not halt execution.HTML failures involve situations in which the generated HTML page is not syntactically correct according to an HTML validator to find HTML failures through checking appropriate tags with closing by parsing DOM tree. In addition, check whether the web page has vulnerable to SQL Injection. If it has vulnerable give some suggestion. Finally fix htmlentities (), magic_quotes_gpc (), stripslashes (), addslashes(), autoload() problems. Pin down framework include limited tracking in native methods, tracking of input parameter through the database Fixing Malformed SQL queries, missing reference (function, class), incorrect buffer cleaning. The final

result shows that code coverage improved from 90% to 100%. (i.e.) 100% coverage alone is not a reliable indicator of the effectiveness of a test set.

3.3 System Architecture

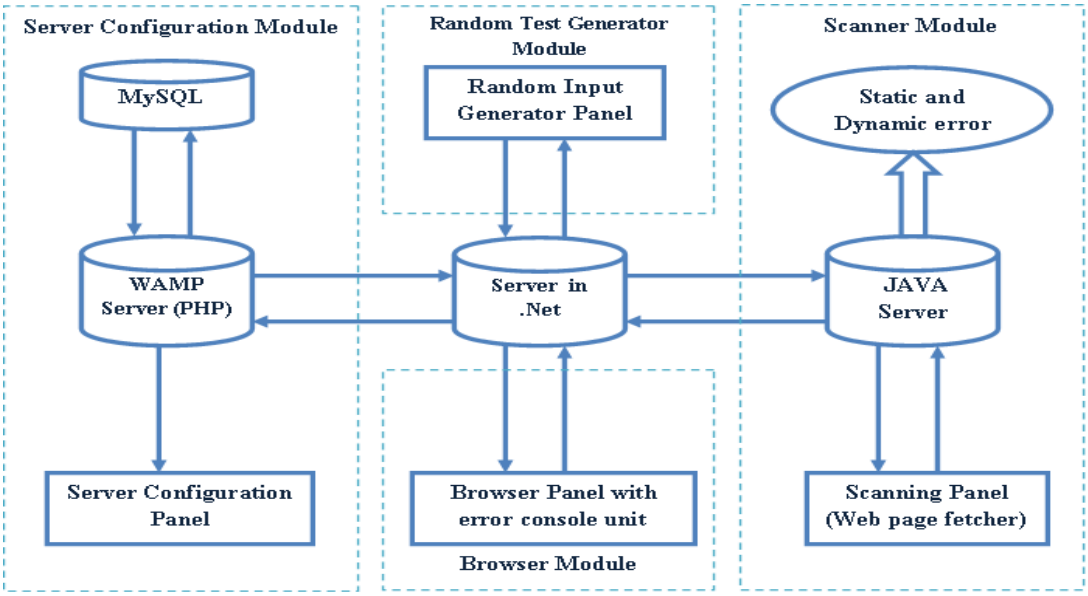


Fig. 1: Pin down framework architecture.

Fig. 1 shows the proposed system architecture in which the function of fault localization process. PHP web application taken as input for fault identification process that produce what are the possible locations for occurring bug. Fault localization process using some algorithms such as Ochiai, Tarantula that determine the suspiciousness ratings for executed statement. Executed statements are arranged in the order which one having high suspiciousness rating placed in the top of order. Generate direct test case based on combined concrete and symbolic execution and similarity criteria. Finally compare the coverage of different algorithms.

4. Implimentation

We have implemented Randomized input generation for server side and client side input fields. Dynamic web application is combination of various scripting language. In this paper, pin down framework implements server side scripting language and predict the fault. Then compare the performance of the fault localization. To increase the efficiency of fault localization process, use CREST tool. Create a web application as an input for determine the bug identification. Finally implement pin down framework and execution bug identification algorithms such as Ochiai and Tarantula for compare results derived from this system.

Major panels of pin down framework:

- Scanner
- Server Configuration
- Application Browser
- Random Input Generator
- Setting

4.1 Scanner

Either html or PHP files can be scanned when it is placed in the scanner module and error will also be detected. This panel will detect browser compatibility issue (blink tag work only in internet explorer browser) resource missing, syntax errors. Pin down framework includes the scanner tool which provides static analyzer that checks the static bugs on given Input files like HTML as well as PHP. And also many files can be added or removed to scanning for detecting errors. And also it will display how many files selected.

4.2 Server Configuration

Any Server Configuration shows the configuration details that present in the server. Current tools for webpage validation cannot handle the dynamically generated pages that are global web applications.

4.3 Application Browser

Application Browser panel can show the web page for the given URL. If the user present in online the requested webpage of the URL is showed on the screen. Otherwise default webpage is showed on the screen.

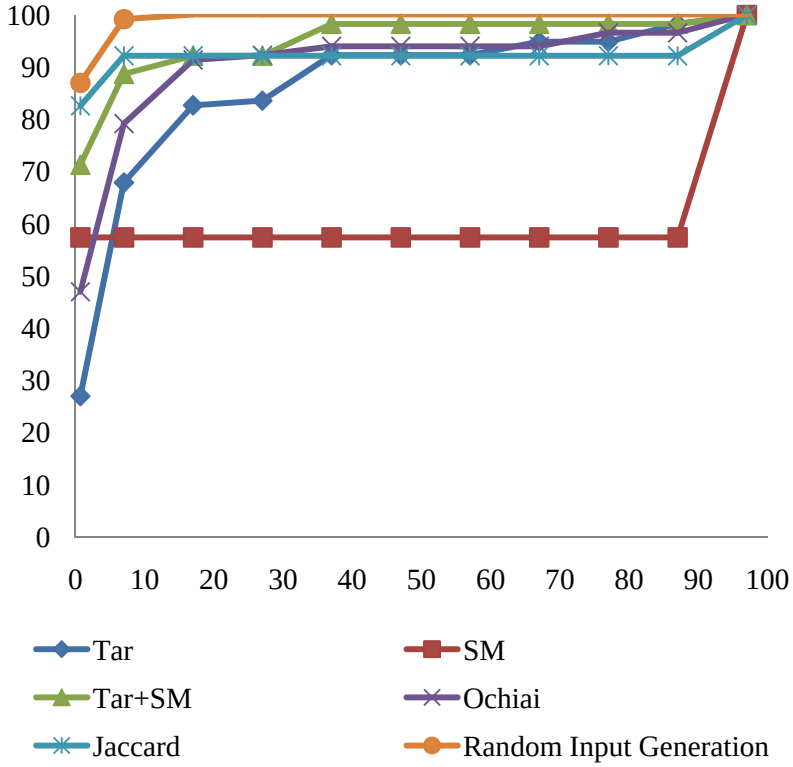
4.4 Random Input Generator

Random Input Generator is the most important tool in pin down framework which provides different number of forms, image content and hyperlinks of web page. And also shows the bugs present in the web page for random input generated by the Random Input Generator. The results show that pin down architecture could effectively optimize the time and cost involved when compared to the previous systems. However it will shows the number of iteration to detect bugs in random input generation screen.

4.5 Setting

From the server settings many options should be appeared. The option of the local host address must set to http://localhost and the option host root directory must set to c: /xampp/htdocs and the option PHP root directory must be set to c: /xampp/php and the option PHP configuration file must be set to c:xampp/php/php.ini and the option MySQL root directory must be set to c: /xampp/Mysql and the option MySQL configuration file must be set to c: /xampp/Mysqlbinimy.ini.

5. Result and Discussion



In this section, discuses about the evaluation result of our concepts. X axis represents number of execution of each algorithm and Y axis mentions statement coverage of these three algorithms. The top most increment represents the efficiency of fault localization of the application. So, automated pin down framework for testing increasing the efficiency of fault localization process. Speed of the process is increased so reduced the time period. X axis mention three algorithms and Y axis represent time period of the process. Compared to the manual bug identification process, it reduced the time period of identification process. It also reduced finding wrong locations. Pin down framework shows better performance when compared to that of the existing system, the analysis is carried out in PHP web application and the results are such as error ratio, statement coverage, time and speed. Error rate measure how the algorithm find the wrong location. Compare three algorithms our method having minimum error rate. Using this method create test suites dynamically, this process reduce testing time and increase speed of the process.

6. Conclusion and Future Enhancement

In this paper, we introduce pin down framework for fault localization. This process is achieved by using Random Input generation technique. This method determined the situation of source code changes from the executed statement in the server side

scripting language on web application. In future, the following methods are implemented with its increased code coverage. Support for PEAR packages, finding .htaccess vulnerabilities, Identify SQL injection vulnerabilities in PHP applications, Debugging against XSS attacks.

References

[1] C.P.Shabariram, "Fault Localization for Dynamic Web Application : A Survey on Recent Developements." International conference on Knowledge Copllabration in Engineering [2014].

[2] S. Artzi, "Fault localization for dynamic web applications," J. Dolby, F.Tip, M. Pistoia., IEEE Transl, vol. 38, no 2, [March or April 2012], pp.314

[3] R. Abreu, "An evaluation of Similarity Coefficient for software fault localization," P. Zoeteweij, A. J. C. Vangemund., International symposium on dependable computing[2006], pp.39–46.

[4] R. Abreu, "On the accuracy of spectrum based fault localization," P. Zoeteweij, A. J. C. Vangemund., Conference, [sept 2007], pp89–98.

[5] H.Agrawal et al, "Fault localization using execuion slice and dataflow tests," International symposium on software reliability engineering[1995]

[6] C. Cadar et al , "EXE: Automatically generating the inputs of death,"Conference on computer and communication[2006].

[7] P.Godefroid et al, "DART: Directed automated Random testing," Conference on programming language design and Implementation[2005].

[8] S.Horwitz, "Interprocedural slicing using dependence graph," T. Reps, D. Binkly., ACM Trans on programming languages and system [1990].

[9] J. Lyle, "Automated Bug localization by program slicing," M. Weiser., Second International conference on computer and applications[1987].

[10] S. Artzi et al, "Directed test generation for effective fault localization," International symposium on software testing and analysis[2010].

[11] S. Artzi et al, "Practical fault localization for dynamic web applications," International conference on software engineering[2010].

[12] S. Artzi et al, "Finding bugs in web application using dynamic test generation and explicit state model checking," J. Dolby, F.Tip, M. D. Ernst, A. Kiezun, D. Dig, A. Paradkar., IEEE Transl on software engineering, vol. 38, no 2, [march or april 2010], pp.274–294.

[13] P. Arumuga nainar et al, "Statistical debugging using compund boolean predicates," International symposium on software testing and analysis[2007]

[14] S. Artzi et al, "A framework for automated testing java script web applications," International conference on software engineering[2010].

[15] B. Baudry, " Improving test suites for efficient fault localization," International conference on software engineering[2006].

[16] Y. Yu, "An emprical study of the effects of test suite reduction on fault localization," International conference on software engineering[2008].

[17] C.P.Shabariram , “Novel Dynamic Fault Localization for Server side Vulnerabilities.” International conference on Global Innovations in computing Technology [2014].

[18] C.P.Shabariram et al, “Pin Down : Fault Localization in Large Dynamic Web Application.” International conference on Knowledge Copllabration in Engineering [2014].