Finding Bugs In Web Applications Using Dynamic Test Generation

Prashant Belhekar
Pimpri Chinchwad College of Engineering, Nigdi

Alka Londhe
Pimpri Chinchwad College of Engineering, Nigdi

Bhavana Lucy
Pimpri Chinchwad College of Engineering, Nigdi

Santosh Kumar
Pimpri Chinchwad College of Engineering, Nigdi

Abstract

The common errors that are usually found in web application are crashing of web scripts and dynamically generated deformed web pages which have an influence over the usability of web application. Dynamically generated web pages that are common on today's internet are not being handled by current tool for web page validation. This project presents dynamic test generation technique for the domain of dynamic web applications. The technique utilizes dynamic and static testing. The explicit state modelling will be a part of future scope. The techniques manages test cases and runs the test cases capturing logically constraints on inputs and minimizes the conditions on input to failing test so the resulting bug reports are small and useful. The tool implements the techniques for PHP programming language, Servlets and web services. The tool helps to generates test inputs for web application, monitors web application for clatters and corroborates that the output conforms to HTML specification.

Keywords: ***Web applications, dynamic analysis, dynamic test generation tools, Bug finder, input generation, Bug report repository.***

1. Introduction

i. Introduction

This paper broader us the way of dynamic test generation to the domain of web application that generates HTML pages during execution. The tool applies these techniques with the perspective of PHP scripting language, Servlets and web services. The popularity of server side web programming has been increased to a greater extent. According to the online research services, PHP, Servlets and other scripting languages has empowered millions of domain including the well-known websites such as Wikipedia, Word press. Besides dynamic content, tool may also generate noteworthy applications. Typically in the form of JAVA script code that is executed on the client side. The techniques used over here are primarily focused on PHP code, servlets. Although we do some minimal analysis of client to verify how it calls upon additional server code through these interface mechanism.

ii. Problem Definition

To present an assistive toolkit which possess an automated technique for finding bugs in deformed HTML pages that leads to web application crashes and to identify minimal part of inputs which is responsible for triggering failures with the help of Static Testing and Dynamic Testing.

This paper will tell us about the ways of testing i.e. static testing and dynamic testing and how it will hunt the bugs present in web application to enhance the performance of the web application.

2. Proposed System

The current paper extends by providing more extensive evaluation in the form of static and dynamic testing which well exhibit detailed classification of bugs. The proposed system creates large number of test cases on a set of URL's and applies these test cases on that URL's and tries to find bugs in web applications. Once the bugs are found a separate log is maintained for it.

The tools previously built to find the loopholes or bugs in web application could only perform static Testing efficiently but in case of Dynamic testing it was quite difficult to find the bugs in dynamically generated web pages, very few tools were developed which could find the bugs in dynamically generated web pages as web script crashes and malformed pages were the common errors found in dynamically created web pages. Tools like DART, CUTE generate tests by executing an application on concrete input values, and then creating additional input values by solving symbolic constraints derived from exercised control-flow paths, such approaches have not been practical in the domain of Web applications, because it posed special challenges due to the dynamism of the programming languages.

Earlier many tools were built, one of them was ”Web vulnerability Scanner” which could scan only CMS (Content Management System) Websites, so we thought to emphasize more on making generalized tool which will scan as well as parse and no such efficient tool was built that could find bugs not only in websites made with the help of PHP, ASP.NET and many more but also could find bugs in web services and web portals.

3. System Architecture

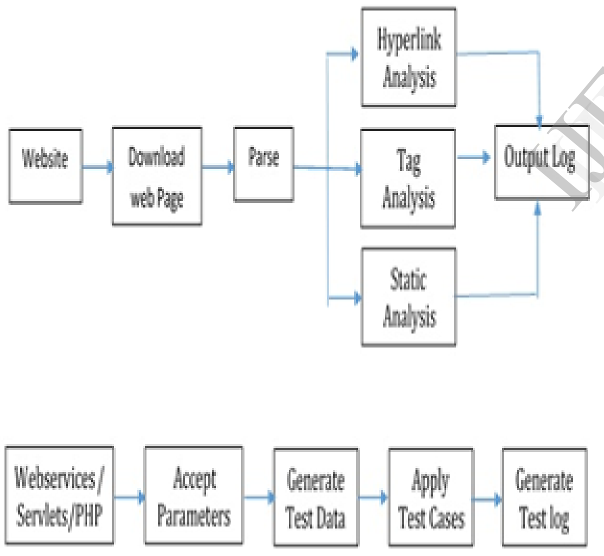


Figure 1. System Architecture

Java I/O file and Serialization:

In computer science, in the context of data storage and transmission, serialization is the process of converting a data structure or object state into a format that can be stored (for example, in a file or memory buffer, or transmitted across a network connection link) and "resurrected" later in the same or another computer environment.

In Our project for storing the Generated Test case profiles we used Serialization technique. Test Profile Generator facilitates to make all probable test cases for given inputs. One can load previously generated test profiles and is able to start auto testing.

DFS Search for Link/Tag Evaluation

For Link extractor module HTML parser is useful in extracting all links in a website. Formally, DFS is an uninformed search that progresses by expanding the first child node of the search tree that appears and thus going deeper and deeper until a goal node is found, or until it hits a node that has no children. Then the search backtracks, returning to the most recent node it hasn't finished exploring. In a non-recursive implementation, all freshly expanded nodes are added to a stack for exploration.

4. Implementation and Algorithms

```
Static Testing:

1. Enter the URL.
2. Download the HTML page.
3. Apply DOM tree parsing Technique
to the Abovementioned HTML page.
4. Find the links.
       i) Apply DFS/BFS technique to
       find the depth of the Link.
       ii) Create a link Database.
       iii) Now Call each link
       Separately.
5. Generate Error or Bug Repository
log.
```

Example:
Let us now consider, we having a website named www.pccoepune.com and it contains n number of links.

Iteration 1: The first input to program is the URL of given website. Now the system will parse the website, generates a tree by using DOM parsing model and applies DFS technique to find out all anchor tags (links) in website. All links get concatenated in a list as follow:

Links	Evaluation
http://www.pccoepune.com/	N.Y.E.
http://www.pccoepune.com/?pageid=9	N.Y.E.
http://www.pccoepune.com/?page_id=1174	N.Y.E.

http://www.pccoepune.com/?page_id=1181	N.Y.E.
http://www.pccoepune.com/?page_id=5	N.Y.E.
http://www.pccoepune.com/?page_id=2556	N.Y.E.
http://www.pccoepune.com/?page_id=2616	N.Y.E.
http://www.pccoepune.com/?page_id=2625	N.Y.E.
http://www.pccoepune.com/?page_id=2631	N.Y.E.
http://www.pccoepune.com/?page_id=2636	N.Y.E.
http://www.pccoepune.com/?page_id=1237	N.Y.E.
http://www.pccoepune.com/?page_id=2642	N.Y.E.
http://www.pccoepune.com/?page_id=1244	N.Y.E.
http://www.pccoepune.com/?page_id=404	N.Y.E.
http://www.pccoepune.com/?page_id=117	N.Y.E.
http://www.pccoepune.com/?page_id=547	N.Y.E.
http://www.pccoepune.com/?page_id=580	N.Y.E.
http://www.pccoepune.com/?page_id=623	N.Y.E.
http://www.pccoepune.com/?page_id=665	N.Y.E.
http://www.pccoepune.com/?page_id=722	N.Y.E.
http://www.pccoepune.com/?page_id=315	N.Y.E.
http://www.pccoepune.com/?page_id=123	N.Y.E.
http://www.pccoepune.com/?page_id=317	N.Y.E.
http://www.pccoepune.com/?page_id=2667	N.Y.E.
http://www.pccoepune.com/?page_id=319	N.Y.E.
http://www.pccoepune.com/?p=56	N.Y.E.
http://www.pccoepune.com/?p=53	N.Y.E.
http://www.pccoepune.com/?p=50	N.Y.E.
http://www.pccoepune.com/?p=39	N.Y.E.
http://www.pccoepune.com/?p=1	N.Y.E.
http://www.pccoepune.com/?page_id=608	N.Y.E.
http://www.pccoepune.com/?page_id=2660	N.Y.E.
http://www.pccoepune.com/?page_id=2280	N.Y.E.
http://www.pccoepune.com/pdf/NSS.pdf'	N.Y.E.
http://rawandale.com/bhartimela2012/'	N.Y.E.
http://www.pccoepune.com/pdf/R&D.pdf'	N.Y.E.
http://www.pccoepune.com/pdf/Student%20Welfare,%20Anti-%20ragging.pdf'	N.Y.E.
http://www.pccoepune.com/pdf/Highlights.pdf'	N.Y.E.
http://rawandale.com/bhartimela2012/\'	N.Y.E.
http://www.pccoepune.com/pdf/R&D.pdf\'	N.Y.E.
http://www.pccoepune.com/pdf/Student%20Welfare,%20Anti-%20ragging.pdf\'	N.Y.E.
http://www.pccoepune.com/pdf/Highlights.pdf\'	N.Y.E.
http://www.pccoepune.com/?page_id=19	N.Y.E.
http://www.pccoepune.com/?page_id=2434	N.Y.E.
http://www.pccoepune.com/pdf/MBA_syllabus.pdf	N.Y.E.
http://www.pccoepune.com/?page_id=1273	N.Y.E.
http://www.pccoepune.com/?page_id=1278	N.Y.E.
http://dreamworth.in/	N.Y.E.

N.Y.E. : Not Yet Evaluated

Table 1: Extracted Links from www.pccocepune.com

Iteration 2: Once all links gets extracted the tool begins checking individual links whether it is valid or not and sets the status accordingly as “ok”, “invalid”, “Not Yet Evaluated”. Iteration 2 will be evaluated for n times, as there are n links in website. After evaluation of all links the system displays result log to the user as below:

Link	Evaluation
http://www.pccoepune.com/	ok
http://www.pccoepune.com/?page_id=9	ok
http://www.pccoepune.com/?page_id=1174	ok
http://www.pccoepune.com/?page_id=1181	ok
http://www.pccoepune.com/?page_id	ok

=5	
http://www.pccoepune.com/?page_id=2556	ok
http://www.pccoepune.com/?page_id=2616	ok
http://www.pccoepune.com/?page_id=2625	ok
http://www.pccoepune.com/?page_id=2631	ok
http://www.pccoepune.com/?page_id=2636	ok
http://www.pccoepune.com/?page_id=1237	ok
http://www.pccoepune.com/?page_id=2642	ok
http://www.pccoepune.com/?page_id=1244	ok
http://www.pccoepune.com/?page_id=404	ok
http://www.pccoepune.com/?page_id=117	ok
http://www.pccoepune.com/?page_id=547	ok
http://www.pccoepune.com/?page_id=580	ok
http://www.pccoepune.com/?page_id=623	ok
http://www.pccoepune.com/?page_id=665	ok
http://www.pccoepune.com/?page_id=722	ok
http://www.pccoepune.com/?page_id=315	ok
http://www.pccoepune.com/?page_id=123	ok
http://www.pccoepune.com/?page_id=317	ok
http://www.pccoepune.com/?page_id=2667	ok
http://www.pccoepune.com/?page_id=319	ok
http://www.pccoepune.com/?p=56	ok
http://www.pccoepune.com/?p=53	ok
http://www.pccoepune.com/?p=50	ok
http://www.pccoepune.com/?p=39	ok
http://www.pccoepune.com/?p=1	ok
http://www.pccoepune.com/?page_id=608	ok
http://www.pccoepune.com/?page_id=2660	ok
http://www.pccoepune.com/?page_id=2280	ok
http://www.pccoepune.com/pdf/NSS.pdf'	invalid
http://rawandale.com/bhartimela2012/'	invalid
http://www.pccoepune.com/pdf/R&D.pdf'	invalid
http://www.pccoepune.com/pdf/Student%20Welfare,%20Anti-%20ragging.pdf'	invalid
http://www.pccoepune.com/pdf/Highlights.pdf'	invalid
http://rawandale.com/bhartimela2012/\'	invalid
http://www.pccoepune.com/pdf/R&D.pdf\'	invalid
http://www.pccoepune.com/pdf/Student%20Welfare,%20Anti-%20ragging.pdf\'	invalid
http://www.pccoepune.com/pdf/Highlights.pdf\'	invalid
http://www.pccoepune.com/?page_id=19	ok
http://www.pccoepune.com/?page_id=2434	ok
http://www.pccoepune.com/pdf/MBA_syllabus.pdf	ok
http://www.pccoepune.com/?page_id=1273	ok
http://www.pccoepune.com/?page_id=1278	ok
http://dreamworth.in/	ok

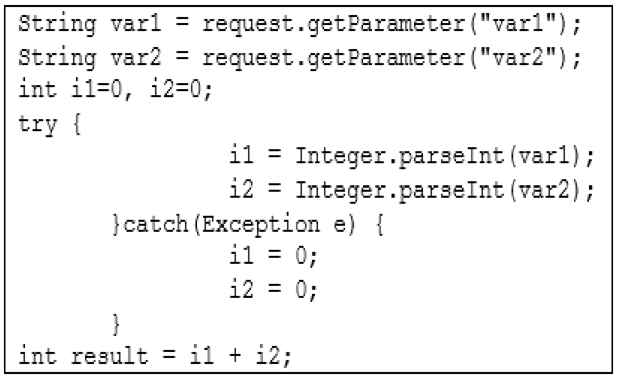
Table 2: Evaluated Links of www.pccocepune.com

```
Dynamic Testing:

1. Create or Generate profile(user will
   create his/her own Test cases).
      i) Enter the URL.
     ii) Set the parameters and their
         default values for the Above
         mentioned address.
    iii) Add to test case list.
     iv) Set expected output.
2. Apply Test case.
3. Generate error log.
4. Store or display error log.
```

Example:

Now consider, we having a website which contains addition servlet. This Servlet accepts two parameters from user and returns addition of these two. As the layman user is unaware about the data types of variables to be provided. So after whatever input given by user; the servlet will accept the inputs in the form of strings and will convert it into integer variables by using Integer.parseInt (String) method. The basic code for addition is shown below.



```
String var1 = request.getParameter("var1");
String var2 = request.getParameter("var2");
int i1=0, i2=0;
try {
                i1 = Integer.parseInt(var1);
                i2 = Integer.parseInt(var2);
        }catch(Exception e) {
                i1 = 0;
                i2 = 0;
        }
int result = i1 + i2;
```

Figure 2: code snippet for addition servlet

According to Dynamic Test generation technique, firstly we need to generate test profile by setting the URL with valid servlet address. After that we need to set parameters and their values. As we are considering very basic example of Addition Servlet, we don't need to trace the path of output. But in case of complex GUI servlets, we need to trace the path of output. To achieve this we are using Web Data Extraction technique to fetch the path of output. So that we can generate all possible test cases for given set of input variables.

The list of inputs and expected outputs for this basic addition servlet by considering all probable combinations is given below:

var1	var2	Expected Output
5	5	10
5	-10	-5
-5	10	5
-10	-10	-20
1.5	1.5	3
1.5	-0.5	1
-1.5	0.5	-1
-1.5	-1.5	-3
A	a	2a
A	-5a	-4a
-7a	2a	-5a
-2a	-2a	-4a
A	+b	a+b
A	-b	a-b
-a	b	b-a
-a	-b	-a-b
a+b	-c	a+b-c
a+b	x+y	a+b+x+y
5	2.5	7.5
5	-2.5	2.5
-5	2.5	-2.5
-5	-2.5	-7.5
5	A	5+a
5	-a	5-a
-a+b	17	17-a+b
-5a	-10b	-5a-10b
5+4i	2+3i	7+7i
7+3i	-2-2i	5+1i
10	2+2i	12+2i
x+yi	a+bi	a+x + (b+y)i
-3.5	A	a-3.5
10*10	25	125
sin(30)	0.5	1
sin(30)	cos(60)	1

Table 3: Probable combinations for addition servlet

While generating test cases we have considered the possible combinations of following data types:

1. Both integer values
2. Both floating point values
3. Both characters
4. Multiple character addition
5. Mixing of above data types
6. Mixing of all above with combination of (++,+-, -+,--)
7. Complex numbers
8. Mathematical expressions
9. Trigonometric terms

Before making test cases user must have full knowledge relevant to the web page to be tested.

Results:

Once all possible test cases are generated, user have to generate test profile and needs to start auto testing. The system will pick up these test cases individually and compare the observed output with expected output.

Var1	Var2	Expected output	Observed output	Results
5	5	10	10	OK
5	-10	-5	-5	OK
-5	10	5	5	OK

-10	-10	-20	-20	OK
1.5	1.5	3	0	ERROR
1.5	-0.5	1	0	ERROR
-1.5	0.5	-1	0	ERROR
-1.5	-1.5	-3	0	ERROR
A	A	2a	0	ERROR
A	-5a	-4a	0	ERROR
-7a	2a	-5a	0	ERROR
-2a	-2a	-4a	0	ERROR
A	+b	a+b	0	ERROR
A	-b	a-b	0	ERROR
-a	B	b-a	0	ERROR
-a	-b	-a-b	0	ERROR
a+b	-c	a+b-c	0	ERROR
a+b	x+y	a+b+x+y	0	ERROR
5	2.5	7.5	0	ERROR
5	-2.5	2.5	0	ERROR
-5	2.5	-2.5	0	ERROR
-5	-2.5	-7.5	0	ERROR
5	A	5+a	0	ERROR
5	-a	5-a	0	ERROR
-a+b	17	17-a+b	0	ERROR
-5a	-10b	-5a-10b	0	ERROR
5+4i	2+3i	7+7i	0	ERROR
7+3i	-2-2i	5+1i	0	ERROR
10	2+2i	12+2i	0	ERROR
2+2i	-a+bi	2-	0	ERROR
x+yi	a+bi	a+x +	0	ERROR
-3.5	A	a-3.5	0	ERROR
10*10	25	125	0	ERROR
sin(30)	0.5	1	0	ERROR
sin(30)	cos(60)	1	0	ERROR

Table 4: comparison of observed and expected outputs

According to the results we came to know that the given servlet is capable to perform addition of only integer values. And suppose it came across the above possible combinations then web service will give wrong answers.

5. Conclusion

We have presented a technique for finding faults in Web applications that is based dynamic test case generation and monitoring the inputs for probable crashes. The work is innovative in several respects. The static testing technique detects all invalid links which are present in a given website. And in dynamic testing we have provided easy interface to the user so that he can generate his own test profile by adding all probable test cases and will monitor the results.

We created the tool that implements the analysis. We evaluated our tool on basic sample Addition Servlet which takes two parameters and returns the addition. For this we provided all probable combination of inputs in numerous test cases.

6. References

[1] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M.D. Ernst, "Finding Bugs in Web Applications using Dynamic Test Generation and Explicit-State Model Checking ", *IEEE transactions on software engineering*, vol. 36, no. 4, july/august 2010.

[2] C. Cadar, V. Ganesh, P.M. Pawlowski, D.L. Dill, and D.R. Engler, "EXE: Automatically Generating Inputs of Death," *Proc. Conf. Computer and Comm. Security*, pp. 322-335, 2006.

[3] G. Wassermann, D. Yu, A. Chander, D. Dhurjati, H. Inamura, and Z. Su, "Dynamic Test Input Generation for Web Applications," *Proc. ACM/SIGSOFT Int'l Symp. Software Testing and Analysis*, pp. 249-260, 2008.

[4] Y. Minamide, "Static Approximation of Dynamically Generated Web Pages," *Proc. Int'l Conf. World Wide Web* 2005.

[5] S. McAllister, E. Kirda, and C. Kruegel, "Leveraging User Interactions for In-Depth Testing of Web Applications," *Proc.11th Int'l Symp. Recent Advances in Intrusion Detection*, pp. 191-210, 2008.

[6] Fonseca, J.; Vieira, M.; Madeira, H., "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks," *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, vol., no., pp.365,372, 17-19 Dec. 2007

[7] www.pccoepune.com