International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7480
***Article Received**: 22 May 2023 **Revised:** 15 July 2023 **Accepted:** 07 August 2023*

Performance Rubrics for Robustness Evaluation of Web Mutation Operators

Dr. S. Suguna Mallika[1], Dr. D. Rajya Lakshmi[2], Dr. T. Esther Rani[3]
[1]Department of CSE
CVR College of Engineering
Hyderabad, India
e-mail: suguna.kishore@gmail.com
[2]Department of CSE
JNTU – GV College of Engineering
Vizianagaram, India
e-mail: rajyalakshmi.cse@jntukucev.ac.in
[3]Department of ECE
CVR College of Engineering
Hyderabad, India
e-mail: estherlawrenc@gmail.com

Abstract—Web Applications are the predominant medium for not only business enterprises but also for service-based sector to establish and continue their online presence. However, the robustness of web application is mandatory in seamless interaction with customers for achieving sustainable business. Intruders and unethical hackers keep trying to gain unauthentic access to the web applications and hence it is more necessary for the web application to be resistant against any such attacks. The strength of a web application is indirectly responsible for gaining customer confidence leading to repeat business as well as attracting new customers for profitable longer run. Once the web application gains credibility it is bound to run successfully. In the current work, an attempt has been made to assess the robustness of mutation operators used to test web applications is made. A few rubrics have been proposed to ascertain the strength of projected mutation operators verified on some sample open-source web applications. The functional attributes of a web application are the functionalities offered by the web application. The non-functional attributes of a typical web application are security, performance, availability. Here, web applications are challenged against the afore mentioned non-functional attributes using rubrics like uniformity, uniqueness, reliability, unpredictability, and entropy. A comprehensive analysis has been made for the robustness of the projected web operators against the designed and formulated rubrics.

Keywords- Mutation Operators, Performance Rubrics, Web Application Testing, Uniqueness, Reliability, Entropy.

I. INTRODUCTION

Businesses across the world today experience a perennial need to ensure their operational feasibility online. This necessitates an immense need for skilled expertise in the development of web applications specifically in web technologies. The need is not only related to software application but also to emphasize upon the economic interlude with the web apps and their very purpose of development. To understand the need for testing of web applications it is important to quickly investigate some of the real ground statistics related to web applications. Currently, there are more than 1.8 billion web applications developed and hosted according to the internetlivestats.com web site and the number increases at a pace of 10 per minute. Most web applications are developed to support and advocate a business model and make it available to the internet users while inherently deriving sustained economic model for the businesses. Most web applications are parked domains and remain inactive while others are active. If at least 25% of the 1.8 billion web sites are active, the discussion is centred around 7.2 million web applications are active and running. The numbers are clear evidence of the need to have a proper testing in place to ensure security, performance for its users. With around 5 billion users using internet and various web applications, every minute, it is that if an error in a web application goes unnoticed into the deployment stage, the kind of damage incurred and the cost to fix the error and distribute it to all its users is inconceivable. Secondly, the most popular languages used for web application development happens to be Python, Java, C#, PHP and JavaScript. From a survey done by Smart Bear titled "The State of Testing "in 2017, around 3400 applications were examined to find the most popular language used. The findings of the survey are presented in Figure 1. The inclination towards usage of python for development of web apps is increasing at a great pace given the robust frameworks like dJango which ensure some fundamental security and performance of the web app.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7480
***Article Received**: 22 May 2023 **Revised:** 15 July 2023 **Accepted:** 07 August 2023*

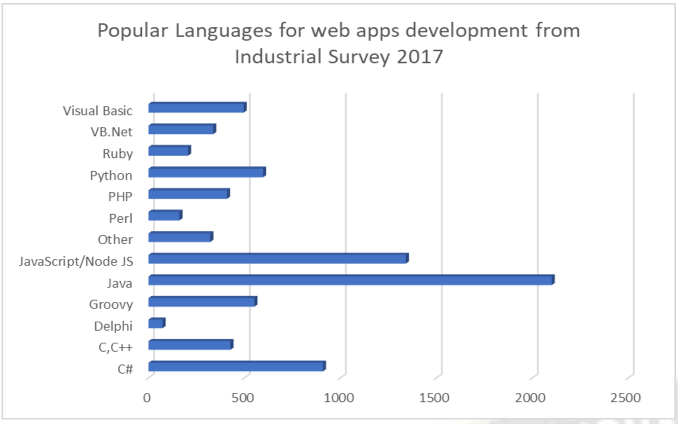

Figure 1 Survey Findings from 3400 applications based on language used for development

It is hence important to devise a standard set of important test cases for ascertaining the health of a web application. And these set of test cases could serve as a reference manual for the medium businesses to assure quality software being released to its consumers.

A recent industry survey report reveals some interesting facts about web application development, testing and effort involving manual testing and automation testing. Most important findings of the survey report are summarized in TABLE 1.

TABLE 1 Important Findings from Industry Survey Report

S. No	Assessment Query	Findings
1	What kind of applications are tested more?	Web applications around 85%
2	Team Sizes	33% of teams have a size of 2 to 5 people. 33% of teams have 6 to 15 people.
3	What web application testing means	Mostly API and UI testing
4	Percentage of testing done	63% perform UI and API testing only
5	Which kind of apps are tested more?	B2B than B2C
6	Most used testing technique	Exploratory testing, unit testing, UI testing,
7	Most used language for web development	Java/JavaScript
8	Concern for testing Applications	Functionality testing- Highest Concerned Availability – Least Concerned

Around 44% of testing teams are commonly comprising of only 2 to 5 people while the remaining teams are employing a strategy of deploying 2 to 15 people in the development teams itself taking responsibility of testing as well [5]. Latest project models like Agile, and frameworks like DevOps the lines between the developer and tester are fading while each member of the team is sporting a multi-faceted role both as a developer as well as a tester [5]. The blend in the roles is increasing day by day.

The rest of the paper is organized as follows. Section II consolidates the limitations of current web application testing from the industry survey report above. Section III comprises of the important attributes a web application must possess inorder to be called robust and thus gain user credibility. Section IV presents and classifies the various mutation operators proposed by the authors in their earlier research works based on non-functional attributes performance and security. Section V comprises of the five rubrics adapted for analyzing mutation operators of web applications. This section also applies the rubrics on the authors' proposed operators and analyzes their performance. Section VI concludes the work with the observations of the authors from the performed work.

II. LIMITATIONS IN CURRENT WEB APPLICATION TESTING

Robustness of a web application is the ability of an application to handle the erroneous behavior of input and deliver the appropriate message to the user [2]. To test the robustness of a web application the tester needs to be completely paranoid about certain attributes like security, performance of the web application [4,6]. If a tester is complacent with the testing, then robustness is affected adversely. Unfortunately, as per industry survey reports most of the testing is limited only to User Interface (UI) testing when it comes to web applications.

Section III elucidates the characteristics which depict that a given web application is robust.

III. NON-FUNCTIONAL ATTRIBUTES

A. Security of Web Application

With the involvement of financial implications in day-to-day transactions of a web application it makes security of a web application more critical [39]. The web application misuse can happen either from client side or server side because of unethical users [3]. This causes trouble to both genuine users and business enterprises. The consequences of lack of web security can range from mild to catastrophic events including cookie theft, browser hijacking, forgery, fake transactions, loss of privacy, and not to forget data theft. The user of the web application however is unaware of the impending loss and sometimes continues to keep using the unsafe applications. There are several types of attacks in the past that compromise the security of a web application [9,38]. A summary of the different types of security attacks that can occur on a web application are presented in Table 2. It is important that these attacks are also well known to the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7480
***Article Received**: 22 May 2023 **Revised:** 15 July 2023 **Accepted:** 07 August 2023*

developers as well as testers to ensure a safer application whenever a new web application is under construction. But unfortunately, many developers are not from security background and the testers focus on functionality testing, or third-party Application Programming Interface (API) functionalities rather than the security. Out of the four important attributes of a web application namely, functionality, performance, security and availability, maximum emphasis is only on functionality testing, after which performance is given importance [15,17]. However, security and availability are not given enough importance due to lack of security-based training.

TABLE 2 Types of Security Attacks on Web Applications

S.No	Type of Attack	Description
1	Cross-site scripting (XSS) [16]	These types of attacks involve executing a malevolent script with the intention of gaining unauthorized access to information.
2	SQL Injection (SQLI) [38]	Specific code to make damage to the database is inserted into the input data and if it is not properly cleaned, then the code can alter the database without proper authentication.
3	Path traversal [30]	Unauthorized access to malicious users can be provided because of some injection of patterns into the hierarchy of the web server.
4	Local File Inclusion [38]	These kinds of attacks run a local application somewhere in the remote location of the machine and can cause damage.
5	Distributed Denial of Service (DDoS) attacks [30]	Network is compromised by flooding with requests and not allowing the authentic users gain access to the services or functionalities of the application.

B. *Performance of Web Application*

One of the important aspects of web application usage is the seamless experience that a customer should receive while trying to use it. An enriched experience of a customer can turn into a prospective business for the enterprise. For such an enriched experience it is important for the web application to exhibit good performance. Good Performance of a web application is response time being less, and qualitative in terms of stability, scalability and interoperability [27,42,43]. Some of the common problems which are often overlooked by the developers, but which hinder the performance of a web application are as follows:

- DNS Issues and Network Connectivity – Domain Name Resolution is a tedious task and might consume a lot of time specially when there are a lot of incorrect DNS queries [14].
- Poor Code Quality – legacy software, software with memory leaks or unoptimized algorithms can cause performance degradation in the web applications.
- Sudden spikes in website traffic – certain times when there are some specific promotions on the web application or an activity, it experiences a sudden gush in the visitor traffic.
- Slow loading time – If servers are not capacitated enough it takes long time for rendering the application to the users thereby adversely affecting the performance.
- Improper Load Balancing – Sometimes web applications tend to perform poor due to poor delegation of load on other servers.
- Improper Title and Meta tags usage – Due to improper usage of title tags and meta tags the rendering of the page becomes abnormally slow while search engines try to retrieve the pages for the customers.
- Optimal Bandwidth Usage Failure - Factors like optimization of image and video compression without loss of quality, minimal usage of scripts, CSS aid in boosting the performance of a web application because in their absence bandwidth usage is not optimal.

IV. CLASSIFICATION OF OPERATORS AS PER NON-FUNCTIONAL ATTRIBUTES

Java is the most popular language for development of web applications according to an industry survey. Around 62% of testers responded using Java, PHP, Python and Groovy as preferred languages for testing of third-party API's as shown in Figure 1[5]. For development of web apps also these are the frequently favored languages by developers [7]. For testing web applications mutation testing is one powerful technique which is yet to be unleashed by the testers [24, 25]. Mutation testing relies on applying mutation operators to application under test and modifying source code to a mutant code [22, 23]. The mutant code is later subjected to a series of planned test cases whose outcome is compared to the outcome of running the test case on source code [7, 8, 10, 11]. If the outcomes vary then the application source code is completely right and if incase same outcome is seen, then the input sample space needs to be enhanced to be able to kill the mutant. There are several tools in the market to automate the implementation of mutation testing on applications [12, 19, 21, 31, 32, 33, 34, 35, 36, 37]. After due increase in the input sample space, if the outcomes continue to be same then the backtracking to dig the flaw in the source code needs to be performed. Having seen the importance of testing web applications against the non-functional attributes, to provide a suite of standard metrics for the medium enterprises to test their software [20, 28, 39, 41]. The authors in their earlier research work proposed 18 operators for defect prevention in web applications [40, 41]. The proposed operators are listed below.

Op 1: DPR – Port Number Replacement - In the code

conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/db?autoReconnect=true&useSSL=false","root","password");, if the port number is modified to any other number, and if an SQL connection exception is thrown, then the web app is bound to be working accurately. If it is not the case then there is a risk of running into connecting to the wrong database. Sometimes developers tend to keep duplicate copies of the database on the same server which were created in the production stages of the web app development leaving duplicate copies of the database on the hosting server.

Op 2: DPD – Port Number Deletion - In the code

conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/db?autoReconnect=true&useSSL=false","root","password");, if the port number is deleted, then an SQL connection exception is thrown.

Op 3: DDNR – Database Name Replacement - In the code conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/db?autoReconnect=true&useSSL=false","root","password");, if the database name is changed to another name and that database has the same userid and password, then it is illegally accessing the tables in other databases. The results of both the original and mutated query can be checked.

Op 4: DLDR – Load Driver Replacement – The existing driver used for connecting to the backend database is replaced with a different driver. If the mutant is killed, then the web application is robust otherwise it is evident that there is a potential vulnerability in terms of database connectivity as malignant users might try to gain unauthentic access to the database through alternate means. To establish connection with the database, the first step is to load the driver into the program, and this can be done by using the following code. Class.forName("com.mysql.jdbc.Driver"); Check the working by replacing the text in the class.forName("..") part and check with the original code execution.

Op5: DSSD – In the code session.setAttribute("name",username);

If the above method is deleted and the getAttribute('name') method checks for the session variable named 'name', then it would generate an error.

Connection Management: Following are operators for finding discrepancies in the code for database connectivity.

Op 6: DCD - Close Method Deletion - But on a larger scale, this creates congestion of connections where many objects hold many resources but do not release them which in turn could increase the network traffic.

Session Management: Incorrect session management can cause a lot of security problems as the pages unintended for the user are rendered if not implemented appropriately.

Op 7: DSSR – Session's setAttribute Name Replacement – Modifying the session variable with another value should ideally not result in false rendering of the pages pertaining to a different session.

Op 8: DSGD – Session's getAttribute function Deletion – in a servlet the session's getAttribute method deletion. This might lead to the problem of improper session closure. If a session is not closed properly then data pertaining to that session even after closing the application can be accessed by simply accessing the URL in the browser.

Op 9: DGSR – Session name Replacement – modification of a session variable. This operator is aimed at checking if there is any illegal rendering of other session's data with a change in the session name.

Op 10: DRUR - Modifying the URL in the code and examining the result with the original code execution. Request Dispatcher method remains unexamined in the previous works thus far.

Modifying the URL- Altering the Url in the code and examining the result with the original code execution.

RequestDispatcher rd=request.getRequestDispatcher("<opening page Name>.html");

Replace "<opening page name>.html" with another name

Cookie Management: Cookies are a way of maintaining user information at the client side for providing customized services and a better web experience to the user. But if cookies are incorrectly handled then the web applications are prone to several types of attacks. The following are the operators for validating cookie management code.

Op 11: DACD – Add Cookie method deletion: This operator simply deleted the cookie method from the source code.

Op 12: DCDM – Cookie Method Modification: This operator modifies the value of the existing cookie method to some random value and checks for the behavior of the web application.

Database Security Management: SQL injection attacks are way of compromising the security of the database of a web application. The web application might still be able to pass all the functional and non-functional test cases, but security of the web application can get compromised while operating it

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7480
***Article Received**: 22 May 2023 **Revised:** 15 July 2023 **Accepted:** 07 August 2023*

dynamically. The following are some of the mutation operators proposed to unearth SQL injection vulnerabilities with a web application.

Op 13: BAR – Basic Authentication Replacement: The regular SQL query which is run internally to authenticate the username and password could be manipulated. For instance we have the SQL query internally as follows:

SELECT * FROM users WHERE UserID = $_GET[‘UserID’] AND Password = $_GET[‘Password’];

For this query a username like ‘admin’ and password like ‘password’ would get replaced like

SELECT * FROM users WHERE UserID = ‘admin’ AND Password = ‘password’;

Instead of Username and Password if a malicious user were to enter, ‘x’ or ‘x’=’x’ as username and Password = ‘x’ or ‘x’=’x’. ‘x’ or ‘x’=’x’ is the mutated code. This will give unauthenticated access to a malicious user and the entire database could be compromised. The mutation operator helps the tester in identifying this vulnerability with the web application under test.

Op 14: AAR – Advanced Authentication Replacement: Here any random URL of the web application is taken and is appended with a “’” and page refreshed. If the web application still renders itself then the website is vulnerable to an SQL Injection Attack. We could easily retrieve the columns and other sensitive data stored in the database. This operator assists a tester in identifying whether the web application is vulnerable for an SQL Injection attack so that advanced security mechanisms could be implemented to protect the access of the database.

Op 15: Cross Site Scripting Vulnerabilities: Cross Site Scripting is the technique of uploading malicious scripts onto target web applications to gain unauthorized access to the web application under test. A typical web applications’s search field or guest book where the user is allowed to post some comments is taken and the following script is posted into it to see if it gets executed.

<script>alert(‘hello’);</script>

If the script gets executed then the site is vulnerable for cross site scripting with a scope to steal cookies, perform advanced phishing attacks by redirecting the URL, defacing the page with HTML and Javascript et.al. types of attacks. This mutation operator could serve the purpose of becoming a rigorous test case to expose the vulnerabilities of a web application under test aimed at strengthening the security aspects of the web application.

Op 16: DSID: This operator checks for accessing of the profile section page of a web user using url resubmission after the session time out of that particular user.

Op 17: DHBR: HTTP Boolean value replacement – logically validation of ongoing session might face issues by removal of the Boolean value.

Op 18: DFIR -Forward Include Replacement - This operator will replace 'forward' with 'include' and vice versa in the following code. But with respect to servlets this operator has not been validated.

The afore mentioned operators are classified based on the non-functional attribute that each of them can affect in the web application. The rationale led to the classification as displayed in the Table 3.

S. No	Name of the Operator	Description	Category	Identified Non-Functional Attribute
1	DSID[40]	This operator aids in identifying if the personal profile section of a web application is accessible despite the user logging out of the application.	Incorrect Session Management	Security
2	DACD[40]	Deletion of a cookie method and its repercussions on the functioning and efficiency of a web application are studied using this operator.	Incorrect Cookie Management	Performance
3	DHBR[40]	HTTP Boolean Replacement- logically validation of ongoing session might face issues by removal of the Boolean value.	Incorrect Session Management	Security
4	DFIR[40]	Forward Include Replacement –forward and include keywords are interchanged to examine the functioning of the servlets despite the interachange.	Incorrect Session Management	Security
5	DCD[40]	Close Method Removal – conn.close() method if unknowingly missed out by the developer while coding causes a retention of resources line like database connectivity which in turn could hinder the performance of a web application.	Incorrect Session Management	Performance
6	DSSR[40]	Sessions Set Attribute Name Replacement - setAttribute is manipulated to check the depiction of the web application.	Incorrect Session Management	Security
7	DGSR[40]	Session Name Replacement – URL session names are manipulated using some arbitrary values to check and see if unauthorised access to the web app can be obtained.	Incorrect Session Management	Security
8	DCDM[40]	Cookie Method Modification- cookie method is modified and a random value is substituted to verify the behavior of the web app.	Incorrect Cookie Management	Performance
9	BAR[40]	Basic Authentication Replacement- manipulates the query executed internally by the servlet to authenticate the user towards gaining access to the web application.	Incorrect Session Management	Security
10	AAR [40]	Advanced Authentication Replacement- Backend vulnerabilities of the web app are thoroughly checked using this operator.	Incorrect Session Management	Security
11	XSSC[40]	Cross Site Scripting Check-non malicious scripts are run through the search field offered by the web app to ensure the attempt is thwarted.	Cross Site Scripting Vulnerability	Security
12	DRUR[40]	URL modification is performed using script like below: RequestDispatcher rd = request.getRequestDispatcher("Welcome.html");	Incorrect Session Management	Security

TABLE 3 Classification of Mutation Operators Based on Non-Functional Attributes

V. ANALYSIS OF RUBRICS FOR WEB OPERATOR EVALUATION

As there is no consensus on the rubrics used for analysis of web application robustness, the current rubrics could be used among the scientific and research communities as the rubrics establish the authenticity of the mutation operators. These rubrics are formulated to ascertain the authenticity of all the operators. The rubrics are inspired from Zhong's et al 's contribution in design and implementation of enhanced light-weight memory-based and monostable physical unclonable functions [13]. They have been adapted in the context of evaluating mutation operators for testing web applications. The operators have been applied on some real time web applications with whom the authors hold a confidentiality agreement and hence they are being presented for evidence on open-source web applications available at github.

http://github.com/nanpj masses a series of open-source web apps developed by Upsorn and Offutt for performing their proposed mutation operators [18,26,29]. The same are currently used in the present work to analyze the efficacy of the operators used to test the web apps. A total of 10 applications have been taken into consideration in the present work where the proposed operators are applied against which the rubrics have been evaluated. Servlet based applications were selected for the testing and analysis purpose. The apps are referred to as Web Applications represented as WAi and WAi refers to Web Application i. webStutter, StudInfoSys, computeGPA ,BSVoting, Conversion , HLVoting, KSVoting, faultseeding, RandomString and Check24Online are ten applications selected for experimentation.

WebStutter gives occurrence of recurrent words for a given input. KSVoting, BSVoting and HLVoting are student voting online apps which allow a student to vote against other votes. Given a small educational institute management application where for the students enrolled credit hours and grades are given, computeGPA issues the GPA of a student. Conversion deals with converting one measurement to another online.

Table 4 summarizes the experiments under research with specifics like number of lines of code in each web application being used for analysis and the total number of components each web app is encompassed of [41].

TABLE 4 Experiments list and Size of Experiments

S.No.	Subjects	Components	LOC
1	webStutter	3	126
2	StudInfoSys	3	1766
3	computeGPA	2	581
4	BSVoting	11	930
5	Conversion	1	388
6	HL Voting	12	939
7	KSVoting	7	1024

A. *Uniformity*

Uniformity is a measure of the percentage of presence of a feature/functionality in a web application. It is a direct measure of the percentage of repeated functionality in various web applications.

$$\boxed{\text{Uniformity} = \frac{1}{n}\sum_{i=1}^{n} R_i * 100\%} \qquad (1)$$

Where Ri – ith response of n responses

Ideal value of Uniformity is 50%.

TABLE 5 Repeated functionality across web applications under test

	DSID	DHBR	DFIR	DRDUR	DSSR	DGSR	XSSC	DRUR	DSGD	DSSD
WA1	1	1	1	1	0	0	1	1	0	0
WA2	1	0	0	0	0	0	1	1	0	0
WA3	1	1	1	1	1	1	1	1	1	1
WA4	1	0	0	0	1	0	1	0	0	1
WA5	1	1	1	0	1	1	1	1	1	1
WA6	1	1	1	1	1	1	1	1	1	1
WA7	1	1	0	0	0	0	1	0	0	0

In the Table 5, if a particular operator was applicable on a web application, then the relevant cell is marked as 1 otherwise 0. Now, Calculating the uniformity rubric against the proposed operators here,

Uniformity = (1/70 ×45)×100 = 64.2%

Hence the uniformity rubric for the operators DSID, DHBR, DFIR, DRDUR, DSSR, DGSR, XSSC, DRUR, DGSD, DSSD is obtained as 64% which is much beyond the desired 50% level of uniformity.

B. Uniqueness

Uniqueness indicates the distinguishability of an operator from other operator in a sequence of mutation tests conducted over different web applications. Ideally the responses generated from two different operators should have varied responses.

$$\boxed{\text{Uniqueness} = \frac{Rui}{RTi} \times 100} \quad (2)$$

Where Rui – Number of Unique responses

RTi – Total Number of responses recorded

Ideal Uniqueness value is expected to be 50%.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7480
***Article Received**: 22 May 2023 **Revised:** 15 July 2023 **Accepted:** 07 August 2023*

TABLE 6 Uniqueness of Responses -I

Operator	DCD	DFIR	DSID	DACD	DHBR
Response	Conn closed	Success	Accessible	Moderate Performance	Session incorrect
	Conn open	failure	Inaccessible	Low performance	Rendered

TABLE 7 Unqiueness of Responses - II

Operator	DPR	DDNR	AAR
Response	Port failure	DB access denied	Threat
	success	Access permitted	secure

TABLE 8 Uniqueness of Responses - III

OPERATOR	XSSC	DGSR	BAR
RESPONSE	VULNERABLE	SECURE	FAILED
	SECURE	INSECURE	SUCCESS

In Table 6, DCD operator is a unique operator to perform a specific test of checking whether a connection is open or closed and is causing a deterioration in the performance. The response given for DCD operator could be either connection could be closed, or connection could be open. Though the exact message is not displayed on the testing tool, however this response serves as a guideline for the tester to understand and comprehend the behavior of the operator on the web app. Similarly, all proposed web operators give varied responses which are unique in nature and help to analyze uniqueness of the operator are provided in Tables 6, Table 7 and Table 8. These operators are for ascertaining the performance and security of the web application.

Here, Uniqueness from Equation (2), for the operators calculated from the above three tables, Tables 6-8 is observed to be more than desired.

Uniqueness = 16/22×100%=72.7%

C. RELIABILITY

Reliability is a measure of capability of an operator to reproduce uniform response across different web applications when applied. However, at times some of the operators are not relevant for a web application based on whether the features are used or not. The Reliability is expressed as shown in Equation (3). and is expected to be '1'.

$$\boxed{\text{Reliability} = 1-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} P(O_j, W_i)} \quad (3)$$

Where Oj – Operator j

Wi – Web Application i

and $\boxed{P(Oj,Wi) = \frac{no.of\ non-applicant\ mutation\ operators}{total\ sample\ space(m*n)}}$

If the operator is able to produce uniform response across different web applications wherever relevant, the respective entry has been marked as 1 and if any operator was not relevant to a web application, then the entry against the appropriate cell is marked as 0. Table 9 captures the responses of the web apps under test against some of the operators.

TABLE 9 Response of an Operator across Web Apps under Test

	DSID	DHBR	DFIR	DRDUR	DSSR	DGSR	XSSC	DRUR	DGSD	DSSD
WA 1	1	1	1	1	0	0	1	1	0	0
WA 3	1	1	1	1	1	1	1	1	1	1
WA 5	1	1	1	0	1	1	1	1	1	1
WA 6	1	1	1	1	1	1	1	1	1	1

Sample space = 9×4 = 36

Total number of values omitted due to non-relevance to the applications = 5

Therefore, Reliability = 1 - 5/36 = 0.86

Obtained value is more than the expected value.

For an sample space of 8 applications, reliability would be attained as

Reliability = 1-6/72 = 0.91

Likewise, as the sample space keeps increasing Reliability rubric approaches the expected value of 1. This is presented in the figure 2.

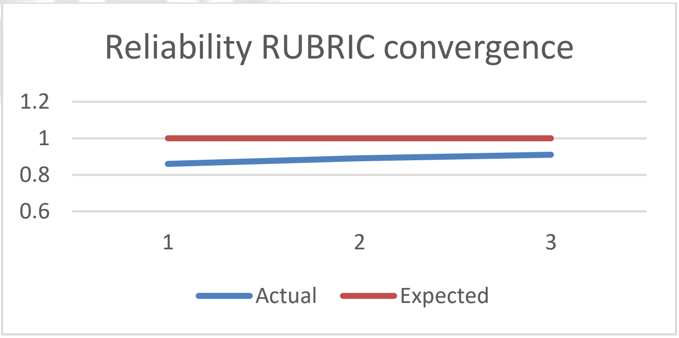

Figure 2 Reliability Rubric Convergence with increase in Sample Space

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9s
DOI: https://doi.org/10.17762/ijritcc.v11i9s.7480
***Article Received**: 22 May 2023 **Revised:** 15 July 2023 **Accepted:** 07 August 2023*

D. UNPREDICTABILITY

A stronger assertion of unpredictability is non-clonability which refers to the resilience of an operator applicability on a web application. An operator applicable on a web application for an instance cannot be duplicated by another operator which means only non-equivalent mutants would be generated by the proposed operators. Applicability of all operators cannot be guaranteed on each and every web app. At times, some features might not be implemented by developers due to which measuring is not complete here.

E. RANDOMNESS/ENTROPY

Measure of randomness of presence of operator on a web application is defined by using entropy. Entropy is defined by

$$\boxed{\hat{E} = -\sum_{x\in\chi}\log(\Pr(X))\Pr(X)} \quad (4)$$

Where X is a random event from set χ

TABLE 10 Applicability of Operator against Web Apps under Test

	DACD	DCD	DPR	DPD	DDNR	DLDR	BAR	AAR	DCDM
WA 1	0	0	0	0	0	0	1	1	0
WA 2	1	0	0	0	0	0	1	1	0
WA 3	0	0	0	0	0	0	0	0	0
WA 4	0	0	0	0	0	0	0	0	0
WA 5	0	0	0	0	0	0	0	0	0
WA 6	0	0	0	0	0	0	0	0	0
WA 7	0	0	0	0	0	0	0	0	0
WA 8	1	1	1	1	1	1	1	1	1
WA 9	0	1	0	0	0	0	0	0	0
WA 10	0	1	0	0	0	0	0	0	0

For any parametric evaluation, it is difficult to predict the randomness of an operator. In the Table 10, '1' signifies the applicability of an operator to a web app and '0' signifies the non-applicability of an operator to a web app. At times, some of the operators are applicable and not applicable based on the nature of the web app and its intended functionality.

Here the randomness/Entropy is obtained as

Randomness/Entropy = -2^9/9×100 = -56.8% Eq. 4

A graph plotting the expected values verses the actual values obtained for the rubrics evaluating the operators against the sample web applications is presented in Figure 2.

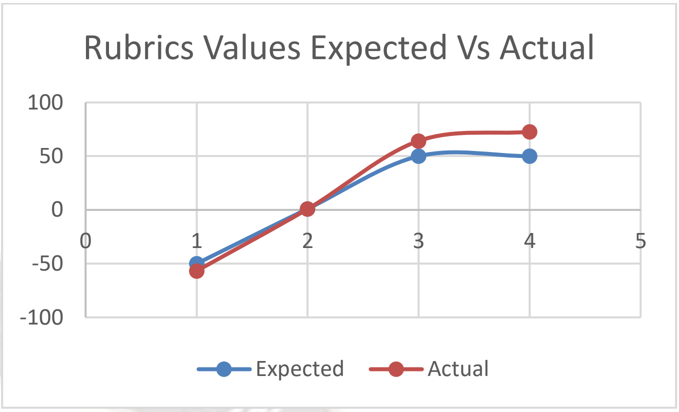

Figure 3 Rubrics values Expected Vs Actual

VI. CONCLUSION

Web application robustness is important for any business to retain customers through integrity and service. Web mutation operators aid in ascertaining the integrity and performance of the web app. The operators are analyzed empirically evaluated for proving their authenticity and establishing the usefulness of the operators. The non-functional attributes using the performance rubrics like uniformity, uniqueness, reliability, unpredictability and Entropy/ randomness are comprehensively analyzed and presented for utilization by budding web developers. Additional mutation operators can be proposed for non-functional features like ease of use of a web application in the future and the same rubrics could be applied against any newly proposed mutation operators or existent mutation operators to assess the quality of mutation test suite.

ACKNOWLEDGMENT

The authors are grateful to the open source repository left for experimentation by Upsorn and Offutt from George Mason University.

REFERENCES

[1] Shabnam Mirshokraie, Ali Mesbah, Karthik Pattabiraman, Guided Mutation Testing for JavaScript Web Applications, IEEE Transactions on Software Engineering, VOL. 41, NO. 5, MAY 2015.

[2] Anuranjan Misra, Mutation Based Test Case Generation, International Journal of Recent Technology and Engineering, Volume-2, Issue-6, January 2014.

[3] Kazuki Nishiura and Yuta Maezawa, Hironori Washizaki, Shinichi Honiden, Mutation Antonio Di Lucca, Anna Rita Fasolino, Francesco Faralli, Ugo De Carlini. "Testing Web Applications". Proceedings of the International Conference on Software Maintenance (ICSM 02) 0-7695-1819-2/02 2002 IEEE.

[4] Yue Jia, Mark Harman, An Analysis and Survey of the Development of Mutation Testing, IEEE Transactions on Software Engineering, VOL. 37, NO. 5, SEPTEMBER/OCTOBER 2011.
[5] http://qablog.practitest.com/wp-content/uploads/2017/03/State_of_testing_2017_final_report.pdf accessed as on 28-01-2021
[6] J.J. Dominguez-Jimenez , A. Estero-Botaro, A. Garcia-Dominguez, I. Medina-Bulo, Evolutionary mutation testing, Information and Software Technology, Elsevier, March 2011.
[7] Sergio Segura, Robert M. Hierons, David Benavides, Antonio Ruiz-Cortés, Mutation testing on an object-oriented framework: An experience report, Information and software Technology, Elsevier, April 2011.
[8] M R Woodward, Mutation testing its origin and evolution, Information and Software Technology, Volume 35, No 3, March 1993
[9] Ben H. Smith *, Laurie Williams, Should software testers use mutation analysis to augment a test set?, The Journal of Systems and Software, Elsevier, June 2009.
[10] Ying Jiang, Shan-Shan Hou, Jin-Hui Shan, Lu Zhang, Bing Xie, Contract-Based Mutation for Testing Components, Proceedings of the 21st IEEE International Conference on Software Maintenance, 2005
[11] Jeff Offutt and Paul Ammann, Lisa (Ling) Liu, Mutation Testing implements Grammar-Based Testing, IEEE Computer Society, 2nd ISSRE workshop on Mutation Analysis, 2006
[12] Shufang Lee, Xiaoying Bai, Yinong Chen, Automatic Mutation Testing and Simulation on OWL-S Specified Web Services, IEEE Computer Society, 41st Annual Simulation Symposium, 2008.
[13] Zhong, H. S. (2019). Design and implementation of enhanced lightweight memory-based and monostable physical unclonable functions. Master's thesis, Nanyang Technological University, Singapore
[14] S. Singh and N. Singh, “Internet of Things (IoT): Security challenges, business opportunities and reference architecture for E-commerce,” in Proc. 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Noida, India, Oct. 2015, pp. 1577-1581
[15] Rui WANG, Ning HUANG, Requirement Model-Based Mutation Testing For Web Service, 4th International Conference on Next Generation Web Services Practices, IEEE, 2008.
[16] Hossain Shahriar and Mohammad Zulkernine, MUTEC: Mutation-based Testing of Cross Site Scripting, ICSE Workshop, IEEE, 2009.
[17] Ying Jiang, Ying-Na Li, Shan-Shan Hou, Lu Zhang, Test-Data Generation for Web Services Based on Contract Mutation, Third IEEE International Conference on Secure Software Integration and Reliability Improvement, 2009.
[18] Upsorn Praphamontripong and Jeff Offutt, Applying Mutation Testing to Web Applications, ICSTW, April 2010.
[19] Filipe Gomes Leme_, Fabiano Cutigi Ferrari, Jos´e Carlos Maldonado, Awais Rashid, Proteum/AJv2: A Mutation-based Testing Tool for Java and AspectJ Programs, 2012
[20] Romain Delamare, Benoit Baudry, Sudipto Ghosh, Yves Le Traon, A Test-Driven Approach to Developing Pointcut Descriptors in AspectJ, International Conference on Software Testing Verification and Validation , IEEE, 2009.
[21] Pradeep Kumar Singh, Om Prakash Sangwan, A Study and Review on the Development of Mutation Testing Tools for Java and Aspect-J Programs, I.J. Modern Education and Computer Science, November 2014.
[22] YuSeung Ma, Jeff Offutt, YongRae Kwon, MuJava: A Mutation System for Java, ICSE, 2006.
[23] David Schuler · Andreas Zeller, Javalanche: Efficient Mutation Testing for Java, ESEC-FSE,ACM, August 24–28, 2009.
[24] Evan Martin and Tao Xie, A Fault Model and Mutation Testing of Access Control Policies, International World Wide Web Conference Committee (IW3C2), ACM, May 8-12, 2007.
[25] David Schuler, Valentin Dallmeier, and Andreas Zeller, Efficient Mutation Testing by Checking Invariant Violations, ISSTA, ACM, July 2009.
[26] Yu-Seung Ma, Yong-Rae Kwon, Jeff Offutt, Inter-Class Mutation Operators for Java, Proceedings of the 13 th International Symposium on Software Reliability Engineering, ISSRE,IEEE, 2002.
[27] Filippo Ricca and Paolo Tonella. “Analysis and Testing of Web Applications”. 0-7695-1050-7/0 2001 IEEE.
[28] Mohan, A., Dinesh Kumar, R. ., & J., S. . (2023). Simulation for Modified Bitumen Incorporated with Crumb Rubber Waste for Flexible Pavement. International Journal of Intelligent Systems and Applications in Engineering, 11(4s), 56–60. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2571
[29] Qianqian Wang, Hirohide Haga, A novel approach to the design and implementation of mutation operators for Object-Oriented programming language, 4th World Congress on Information and Communication Technologies, IEEE, December 2014.
[30] Upsorn Praphamontripong, Jeff Offutt, Lin Deng, and JingJing Gu, An Experimental Evaluation of Web Mutation Operators, 2016 IEEE International Conference on Software Testing, Verification and Validation Workshops, 2016.
[31] Jay Nanavati, Fan Wu, Mark Harman, Yue Jia and Jens Krinke, Mutation Testing of Memory-Related Operators, IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops 2015.
[32] https://galera.ii.pw.edu.pl/~adr/CREAM/ref.php
[33] Mike Papadakis , Marcio E. Delamaro , Yves Le Traon, Proteum/FL: a Tool for Localizing Faults using Mutation Analysis, 13th International Working Conference on Source Code Analysis and Manipulation, IEEE, 2013.
[34] Mr. Dharmesh Dhabliya, Dr.S.A.Sivakumar. (2019). Analysis and Design of Universal Shift Register Using Pulsed Latches . International Journal of New Practices in Management and Engineering, 8(03), 10 - 16. https://doi.org/10.17762/ijnpme.v8i03.78
[35] http://alarcos.esi.uclm.es/testooj3/doc/testooj3usersmanual.pdf , found as on 24/2/17.
[36] Simona Nica and Franz Wotawa, EqMutDetect – A Tool for Equivalent Mutant Detection in Embedded Systems, EqMutDetect – 10th International Workshop on Intelligent Solutions in Embedded Systems,2012.

[37] Haitao Dan and Robert M. Hierons, SMT-C: A Semantic Mutation Testing Tool for C, Fifth International Conference on Software Testing, Verification and Validation, IEEE, 2012.

[38] https://pypi.python.org/pypi/MutPy/ as on 24/2/17.

[39] Jacobs, M., Georgiev, I., Đorđević, S., Oliveira, F., & Sánchez, F. Efficient Clustering Algorithms for Big Data Analytics. Kuwait Journal of Machine Learning, 1(3). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/138

[40] Shweta Rani, Bharti Suri, Sunil Kumar Khatri, Experimental comparison of automated mutation testing tools for java, Reliability Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions) 2015 4th International Conference on, pp. 1-6, 2015.

[41] LaShanda Dukes, Xiaohong Yuan, Francis Akowuah. “A Case Study on Web Application Security Testing with Tools and Manual Testing”. 978-1-4799-0053-4/13 2013 IEEE.

[42] Lakshmi, D.R., Mallika, S.S. “A review on web application testing and its current research directions” International Journal of Electrical and Computer Engineering, 2017, 7(4), pp. 2132–2141

[43] Suguna Mallika, S., Rajya Lakshmi, D. “MUTWEB-A testing tool for performing mutation testing of java and servlet based web applications” International Journal of Innovative Technology and Exploring Engineering, 2019, 8(12), pp. 5406–5413

[44] Suguna Mallika, S., Rajya Lakshmi, D. “Mutation Testing and Its Analysis on Web Applications for Defect Prevention and Performance Improvement” International Journal of e-Collaboration, 2021, 17(1), pp. 71–88

[45] Rao, A. Ananda, and K. Narendar Reddy. "Detecting bad smells in object oriented design using design change propagation probability matrix." Proceedings of the international multiconference of engineers and computer scientists. Vol. 1. 2008.

[46] Dattatreya, V., KV Chalapati Rao, and V. M. Rayudu. "Agile Programming and Design Patterns in Web Development-A Case Study." International Journal of Software Engineering & Applications 3.1 (2012): 37.