International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 04 | Apr -2017 www.irjet.net p-ISSN: 2395-0072

# Software Testing methods in the view of Resource preservation and Information outbreak

**R. Sacin Varma[1], M.K. Ajay Krishnan[2], P.S.S. Akshay Krishna[3], L. Ramanathan[4]**

*[1]Dept. Of computer science engineering, VIT university, Vellore, India*
*[2]Dept. Of computer science engineering, VIT university, Vellore, India*
*[3]Dept. Of computer science engineering, VIT university, Vellore, India*
*[4]Dept. Of computer science engineering, VIT university, Vellore, India*

---

**Abstract -** *Software testing research has not kept up with current programming trends, design framework, plans and applications. This means that software engineering education falls short of providing individuals with the type of knowledge and training that other engineering specialties require. Testing researchers ought to give careful consideration to areas that are right now pertinent for practicing programming designers such as embedded systems, mobile devices, safety-critical systems and other modern paradigms in order to provide usable results and techniques for testers. This survey paper focuses on visiting these grounds of software testing and the various procedures, methods and approaches in achieving the same.*

***Key Words*: Software testing strategies, Software testing, testing tools, Test plans, Software testing principles, Research orientation, Risk management, Software quality assurance.**

## 1.INTRODUCTION

In this paper we are going to describe the steps to be taken to correctly assist a software project in terms of resource management i.e. optimizing the project to run faster, have a smaller size both on the memory and on the hard drive, and identifying the information leakage points which could be in terms of compromising private data or simply exploiting them to bypass security checks. This paper mainly focuses on the development and identification perspectives and not on the remedy, as a remedy is always easier to find than the vulnerability itself. As Sun Tzu once said "If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.". We have reviewed papers that work on providing relevant information on testing and debugging for some of the most common vulnerabilities and optimization issues which are rather ignored even though they are so prevalent. It should always be remembered that the goal of software testing is to evaluate the capability or usability of a program by detecting the errors systematically, preferably in a stepwise order, so as to spend minimum amount of time and efforts on one of the most important phases of software development i.e. the software quality assurance (SQA). It should also be noted that all the errors or bugs which are detected need not be fixed if the risk associated with the problem is ignorable, which can only be assessed through testing. In this Survey Paper, we've reviewed papers that explore on the same.

### 1.1 Resource Preservation

One of the most important subject in resource preservation is to use the resources to their minimum. For optimizing data in terms of operation, it is always a good practice to identify the clusters of code that are function independently. On doing this the code can be put into multiple threads, to take full advantage of the multi core architecture of the modern computers to make multiple processes run at the same time leading to a smaller execution time thus making the project faster.

For optimizing data in terms of memory, a top-down approach can be followed i.e. write the code and then try minimizing the number of static variables used first and then the lines of code and then reduce the number of dynamic variables where ever it is possible. Optimizing data in terms of physical memory is very important as the user cannot use multiple applications at an optimal level if all the memory is occupied by one program only. Whereas the secondary memory i.e. the lines of code need not be paid much attention as secondary memory is generally very abundant.

Using resource preservation doesn't only mean optimizing the data usage with the resources that are used but also keeping backup for the resources and protecting the present system resources. Of course it the application doesn't use the internet the previously mentioned resource optimization in terms of data would be enough but if that is not the case then there is a lot more to be worried about like the databases, the data in the databases, the servers, the high end routers that deal with ISP level transfers etc. they can all be altered or deprived of their functions or content. In fact this paper mainly focuses on such type of software.

### 1.2 Information Outbreak

The leakage of information of any sort i.e. the user private data or the source code of the software or a potential threat that was not patched.

### 1.3 Why Do Testing, What Is the Worst That Could Happen?

A hacker might delete all the records in the database and this will lead to preforming the restore operations which are very tedious. The hacker might do even worse if he manipulated the data of multiple users in a random order. The hacker might embed a script to transfer all the cookies of a user to another website, that might include passwords or the cookies of other websites which can be used to directly login to someone else's account. The hacker might embed a script which redirects the present page to a phishing page or simply generate multiple alert messages or multiple tabs or place one page over the other to steal details etc. to irritate or take advantage of the user.

The system resources taken to generate an alert message or create a new tab may not be much but they still do irritate the user and they would definitely discourage the user from visiting or opening the software. This is clearly the most dangerous threat for a customer driven company because they might lose a potential client.

## 2. BEFORE THE TESTING PROCEDURE

Before testing, it is important to decide the type of testing you are going to perform.

### 2.1 Different Types of Testing

#### 2.1.1 White Box Testing

In this testing, source code and structure of framework is made obvious. Thus, it is exceedingly proficient in recognizing and resolving problems, because bugs can be found even before they cause any problem. It is a strategy for finding errors in which the analyzer has complete knowledge of how the software components work and interact. This strategy is generally not used for debugging in large systems, so it is mostly used for web service based applications. A few types of white box testing techniques are basis Path Testing, Loop Testing, Control Structure Testing.

#### 2.1.2 Black Box Testing

A black box method is used when internal details and workings are not understood or accessibility is not granted to its user. It is testing which involves cleaver guesses and trial-and-error methods to determine the errors as coding knowledge or internal structure need not be known. The main aim of this test is to check how well the software goes with the specified requirements. Thus, it just inspects the crucial part of the framework. It acts like an input output model in which the tester checks for a relevant output for a given input. Different types of Black box testing methods are Equivalent Partitioning, Boundary value Analysis, Cause-Effect Graphing Techniques, Comparison Testing, Fuzz Testing, Model-based Testing.

#### 2.1.3 Grey Box Testing

As of late, a third testing technique has been likewise considered i.e. grey box testing. It is a testing method used when the analyzer has some knowledge over the internal working of the system and its underlying code. It uses lesser guess work as some of the things are known, it is the way most of the companies give their product if it is to be tested by a third party tester. This method may not give the best results theoretically when compared to white box testing but it one of the most natural way of testing the software. This particular method is mostly unbiased because the tester is not required to have access to the source code, so it generally leads to more creative attacks while testing.

### 2.2 How to Test Effectively

Test a program in order to make it fail: Testing is the way toward executing a program with the expectation of discovering bugs and mistakes. Testing turns out to be more successful when bugs and errors are discovered.

Begin testing early: This helps in finding and settling various errors in the early phases of the software development, this further reduces the time spent in finding errors in the future.

Test Plan: Test Plan ordinarily portrays test procedure, test scope, test goals, test condition, deliverables of the test, dangers and alleviation included, levels of testing to be connected, strategies and techniques.

Effective Test cases: Effective test cases must be proposed with the goal that they can be measured and so that clear test outcomes can be obtained.

Test conditions of validity: The project should first be tested with valid inputs, then it should be tested for invalid test inputs and unexpected output conditions are expected.

Test at various levels: Distinctive testing must be done at various levels of testing so that individuals can perform different testing methods at all level.

## 3. FINDING THREATS USING THREAT MODELLING

Threat modelling is the way of detecting threats and potential threats and their effects on a framework. It is

typically used as a part of the software development life-cycle (SDLC) or risk management procedure to empower engineers and analyzers to have a global perspective of potential dangers posed to a framework. Along these lines, threat modelling gives an establishment to the risk profile associated with the arrangement of web application services, and the exposure made with the accessibility of these administrations.

### 3.1 Make an Application Overview

The application overview is normally depicted in pictorial form like a module or deployment diagram. It ought to be refreshed as the framework is decomposed and new components are added or discovered.

### 3.2 Breakdown The System

By appreciating a greater amount of the internal workings of the framework, the ease with which dangers can be found is enhanced and the completeness of the test or analysis will be increased. Breaking down the system can work on various attributes like trust level, assets, entry points, information flow and use cases.

### 3.3 Threat Enumeration

Threat enumeration and discovery of bugs and errors is normally performed by security masters, designers and testers as a group like a brainstorming exercise. For every risk, it is important to recognize the cause and attack vectors and the potentially affected resources. It might likewise be valuable to decide the trust levels required to utilize the alarming entry point.

Risk assessment helps us comprehend the kind of threat better and to take the vital steps required to completely remove or reduce the effect of the consequences. One type of arrangement is known as the STRIDE model which includes checking for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of privilege. All of which are common problems which are mostly ignored due to trust in the software being used on a regular basis by the companies.

### 3.4 Document The Threat Profile

So as to complete the documentation of the threat profile, we should have the Test Plan Document, Threat Profile Document, Test Case Execution Results. Through this documentation procedure we would arrive at Testing Report which can be readily executed and a Technical Testing Report.

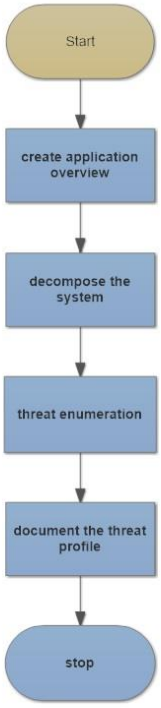


**Fig -1**: Threat modelling flow chart

## 4. Dynamic Analysis

This is generally employed testing in web applications. Security vulnerabilities in web applications may bring about stealing of important and private information, breaking of information integrity or influence web application accessibility. In this way the assignment of securing web applications is paramount. This paper presents augmentations to the Tainted Mode Model which permits inter-module vulnerabilities detection. Also, this paper displays another way to deal with vulnerabilities which combines favorable circumstances of penetration testing and dynamic analysis.

The most effective method for discovering security vulnerabilities in web applications is manual code review. This method is extremely tedious, requires expert skills, and is inclined to overlooked errors. Subsequently, security society effectively creates computerized ways to deal with discovering security vulnerabilities.

### 4.1 Tainted Mode Model

This model was executed by the approach of static analysis for PHP and Java advances and by methods for dynamic analysis for Perl, Ruby, PHP, and Java.

A few drawbacks of the tainted model include

-Any cleansing routine eliminates tainted flags from the string values without much relation to the critical statement that uses this value subsequently. This may bring about overlooked SQL injection vulnerability. This drawback reduces the wholeness of the investigation.

-Inability to deal with validation that is established as conditional branching

They consolidate static and dynamic analysis methods to recognize faulty sanitization methods which can be circumvent by an attacker. The approach first employs string analysis to develop grammar syntax representing a set of values in the given statement. Subsequently, dynamic analysis is connected to check the outcomes acquired by the initial step. The objective of the second step is to decrease false positives.

Inject malicious data into Web applications. Common methods used include:

• Parameter tampering: gives uniquely created vindictive values in fields of HTML forms.

• URL control: utilize uncommonly created parameters to be submitted to the Web application as a feature of the URL.

• Hidden field control: set shrouded fields of HTML forms in Web pages to malicious values.

• HTTP header tampering/altering: control parts of HTTP solicitations sent to the application.

• Cookie poisoning/harming: put malicious information in cookies, small files sent to Web-based applications.

• SQL injection: pass input containing SQL commands to a database server for execution.

• Cross-site scripting: abuse applications that yield unchecked input verbatim to trap the client into executing vindictive scripts.

•HTTP response splitting: misuse applications that yield input verbatim to perform Web page mutilations or Web cache poisoning attacks.

## 5. Security Vulnerabilities

Most of the security vulnerabilities are because of backdoors put by employees or forgetting to remove a piece of code that they used during testing.

These vulnerabilities have existed from a very long time and every company should make sure that their products are definitely not vulnerable to them.

**Table -1:** STRIDE testing

| Category | Description |
|---|---|
| Spoofing | Spoofing covers the large usage of faked credentials to access assets that the attacker is not expected/certified to hold access to. If a web service insufficiently checks the credentials handed out by its clients, it might be vulnerable to spoofing attacks.<br><br>Credential forgery, session hijacking and impersonation attacks are examples. |
| Tampering | Tampering is the unauthorized alteration of information in a framework or as it streams between parts of a framework. Tampering attacks comprise of changing corporate information, man-in-the-middle attacks and the inclusion of malicious software like viruses and Trojans. The impacts can be fatal and make interruptions in service and loss of data integrity. |
| Repudiation | Repudiation is the refusal by an agent of having completed an action, notwithstanding the action having really been finished by that agent. This threat exists when different agents have no technique for demonstrating that something occurred. It can prompt information irregularities and the failure to demonstrate or discredit transactions have happened. |
| Information Disclosure | Information disclosure is the capacity for clients to access information that they are not authorized to access. If the web service doesn't entirely verify the identity of its clients it might be unveiling private information or web services implantation details to unapproved or malicious clients. Information disclosure can have drastic results for an association. |
| Denial of Service | A denial of service threats focuses on exhausting the computing or network resources of a framework or unconditional usage of an execution flaw with a specific end goal to keep system from offering services to its legitimate users. |
| Elevation of privilege | An elevation of privilege attack includes an attacker accessing information or functionality of frameworks to which they are not authorized. These vulnerabilities can emerge due to many factors but are more often than not a result of defective authorization mechanisms. The outcomes of an elevation of privilege attack are serious and may bring about total compromise of the system. |

## 6. CONCLUSIONS

There is no structure that is particularly custom fitted for the security testing of web based services. Web services are

growing in importance in the Enterprise sector and therefore, the security of web services will end vital. This paper recognizes and points at a way to deal with web services security testing including devices, procedures and processes. It intends to give a manual for associations wishing to receive an institutionalized procedure for assessing security mechanisms in web services.

**ACKNOWLEDGEMENT**

We would like to thank our project guide Mr. Ramanathan L.(Assistant Professor) of the Computer science engineering department for encouraging and guiding us throughout the project.

**REFERENCES**

1. T. Sofia (Pg Scholar) Mr. R. Kannan (M.E. Asst Proff)," detecting security vulnerabilities in web applications using dynamic analysis with penetration testing ", IJISAER, *Volume 13, Issue 40 Ver. II (Jan. 2015)*
2. Ana Rosa Cavalli, Azzedine Benameur, Wissam Mallouli, "A Passive Testing Approach for Security Checking and its Practical Usage for Web Services Monitoring".
3. Andrey Petukhov, Dmitry Kozlov," Detecting Security vulnerabilities in Web Applications Using Dynamic Analysis with Penetration Testing".
4. Rasneet Kaur Chauhan and Iqbal Singh,"Latest Research and Development on Software Testing Techniques and Tools".
5. Colin Wong, Daniel Grzelak ,"A web services security framework", SIFT SPECIAL PUBLICATION.
6. Thomas J. Ostrand, Elaine J. Weyuker, "Software testing research and software engineering education".
7. Antonia Bertolino, "Software Testing Research: Achievements, Challenges, Dreams".
8. Khushal Singh, Vikas, " Analysis of Security Issues in Web Applications through Penetration Testing".
9. Jeff Orloff, "Web application security: Testing for vulnerabilities", IBM developerWorks.
10. Vala .R, Jasek .R, "security testing of web applications".