Verification and Validation of an Open Source Software

Part-I: Static Analysis

Siddhartha Srinadhuni
9508286854
sisr16@student.bth.se
Madhukar Enugurthi
9506077537
maen16@student.bth.se

Venkata Sai Anurudh Gudivada
9410222872
vegu16@student.bth.se
Divyani Pamulapati
9502220701
dipa16@student.bth.se

***Abstract*— Testing is a process of analyzing the source code of the software under inspection and to investigate the vulnerabilities that are presented. The two types, Static and Dynamic analysis of an open source have been presented in this document. Further, this paper reports the detailed static and dynamic analysis performed on an open source community plugin, BuddyPress. The static analysis has been achieved by using open source tools, RIPS and Visual Code Grepper(VCG). Analogously, the dynamic analysis has been performed using the open source tools, Selenium IDE and OWASP Zed Attack Proxy (ZAP).**

Keywords— static analysis, dynamic analysis, BuddyPress, Selenium IDE, automated test case generation, OWASP ZAP, Virtual Code Grepper.

I. INTRODUCTION

WordPress is a reputed and popular online content management used in many blogs, home pages and social communities. There are over millions of users that use WordPress every month. The major reason of its popularity is because of the vast range of plugins provided by it. One of the social community plugins is BuddyPress [1]. The task was to analyze an open PHP code obtained from BuddyPress, statically and dynamically.

The code under test is an open source PHP code. Hypertext Preprocessor (PHP) is a cross platform scripting language that is designed to integrate with a server [2]**.** Other than being an open source technology, it has a well structured Application Program Interface (API) and also allows further functional extensions. We have used Apache web server for this process of analysis of PHP code.

This document has been divided into two parts. Dealing with the Static Automated Testing is classified as Part I and Part II explains about the Dynamic Analysis. Static Analysis is a prime technique which is extensively used for the better understanding of programs in software tools. We have used RIPS scanner and Visual Code Grepper tools for the testing of the given source code and investigated the yielded outcomes of the errors that the tool has provided

Part I is subdivided as follows: Section II deals with the inspection goals that we have set. It is followed by the motivation behind selection of the tool to analyze and implement the static analysis in Section III. Further, Section IV talks about code sampling, and Section V and VI discusses about warnings, outcomes and the review process respectively. Findings about the use of the static analysis tool are carried out as conclusions in Section VII.

II. INSPECTION GOALS

For our literature, it is important to choose goals as they form the basis of choosing a tool. four major aspects of the code which are clarity, security, finding defects, consistency and optimization have been chosen as our primary goals. Since these factors drive the code, this became the major rationale behind choosing these particular goals as our code.

- Clarity: Being an open source PHP code, primarily demands to be understood so as to improve the quality. Hence clarity on point, is required in reference to the open source code. This became a major motivation behind the choosing of this goal of inspection.
- Security: This being an inspection goal, gives us an insight as to what kind of vulnerabilities we are dealing with. Through primary studies we have learnt that the security is a primary concern in PHP through different security attacks. This has been another major rationale for us behind the choosing of this tool [3].
- Finding Vulnerabilities: When it comes to testing, evidently, bugs and defects have to be inspected for and have to be detected. The major motivation to end up at this is to address the correctness of the code. This being our self-defined inspection goal, is set to make our task reliable.
- Consistency: Inspection of the open source PHP has to be done inline of the existing code when the changes are made to the existing code in order to improve the quality of the code. This also has been our prime inspection goal.
- Reliability: The test tools, evidently make a noticeable change in terms of size and its complexities. It makes a tester to detect the bugs with ease. Hence reliability is ought to be a prerequisite before carrying out a test [3].

Static analysis tools have been selected in order to meet the requirements of the inspection goals and we surmised that after further selecting a tool on the basis of scrutiny is explained in the next section.

III. SELECTION OF THE TOOL

Analysis by static tools analyze and identify the qualitative and quantitative information by checking the code throughout for bugs such as bad smells and programming errors [4]. Our quest, in the inception of this literature, has been understanding the tools and its features through various documentations that are provided on the official websites and relevant forums and understanding them through the best of our abilities. We have narrowed down the static analysis to 8 most commonly used and most powerful PHP analysis tools based on the inspection goals that we have set. This list has been provided through extensive search of internet and taking reviews to consideration. We have used www.google.com, www.owasp.org and [5] for this process of understanding the tools on the basis of our needs, for this literature.

Since there has been a constraint of time and also we, possessing limited knowledge, we again narrowed the tools down by studying the documentation. We also installed few tools and rejected as they were not satisfying our inspection goals. This helped us to understand the tools and led us to accept or reject them accordingly.

We have listed the tools that are analyzed and the motivation behind accepting/rejecting below in **table number 1.** After carefully examining the features of each tool, we have come to a conclusion that RIPS scanner [6] [7]and Visual Code Grepper [8] will help us carry out the static analysis inline of the defined inspection goals.

For the initial scrutiny, through the best of our abilities, we have rejected PHP storm as it has too complex dynamics and requires Oracle java to run and the major reason being it is a commercial tool.

- PHP Code Sniffer has been rejected because it can use only single coding standard for checking.
- Rejection of PHP Mess Detector was done because it is a very young tool and it only provides a limited set of predefined rules and also it cannot check JavaScript files.
- We have decided not to use PHP Unit as it does not cover a majority of code metrics.
- PHP Security audit tool and PHP Security Scanner were straight away rejected because there were not much reviews provided and we were in the dire need of a reliable tool.
- After rejecting the above tools, we were shortlisted with PIXY, Visual Code Grepper and RIPS Scanner. After reviewing few scholarly articles, we decided on not using PIXY as malicious data can never arise from the code constructs [9].

Hence, after the final scrutiny we were left with RIPS Scanner and Visual Code Grepper. RIPS Scanner being the most popular tool for static analysis has been the prime motivation for us to select this particular tool.

Also, scanning of the PHP code became user friendly with the tools we have used. Another rationale being that RIPS can detect vulnerabilities such as XSS attacks, File disclosure, manipulation attacks, PHP Flow control, SQL and PHP Injection attacks which will enable the user to examine the code and determine if it is a true or a false positive. Visual Code Grepper, another tool that has been taken into consideration after the final scrutiny as this tool. Apart from being cross platform and open source, like RIPS Scanner, VCG is also inline to our inspection goals. Whilst performing complex checks, it also provides a void to search for any bad functions that are present in the code. Hence this made us to choose RIPS and VCG over the tools that have been mentioned above.

Aspects	RIPS Scanner
Platform	Cross Platform
Inputs	PHP Code
Output	Vulnerabilities
License	GNU General Public License version 3.0 (GPLv3)
Void for Tool Upgradation	Yes
Language that are supported	PHP
Compatibility with other application	Yes
Commercial (or) Open Source	Open Source
Documentation	Provided extensively on the website [6]
Last Updated	2015-10-14

RIPS documentation analysis

Aspects	Visual Code Grepper
Version	2.0.2

Inputs	PHP Code
Output	Vulnerabilities
License	GNU General Public License version 3.0 (GPLv3)
Void for Tool Upgradation	Yes
Language that are supported	C/C++, C#, VB, PHP, Java and PL/SQL
Compatibility with other application	Yes
Commercial (or)) Open Source	Open Source
Documentation	Provided extensively on the website [8]
Last Updated	2015-01-13

Visual Code Grepper Documentation Analysis

S.No	Tool	Accept/Reject	Motivation
1	PHP MessDetector	Rejected	Provides only a set of predefined rules. Not In line with our goals of inspecting security
2	PHP CodeSniffer	Rejected	Can use only single coding standard for checking and analyzing the code. Not compatible with our inspection goals
3	RIPS Scanner	Accepted	Provides information about errors and classification of the bugs are made. Inspection Goals have been fulfilled.
4	PIXY	Rejected	Malicious data can never arise from the constructs through the analysis by this tool. Correctness of the code is not maintained.
5	PHP Storm	Rejected	Too complex dynamic and Commercial. Not optimized and not fulfilling one of our goals.
6	PHP Security audit tool and PHP Security Scanner	Rejected	No proper quality reviews available. Not reliable thereby breaching our inspection criteria.
7	Visual Code Grepper	Accepted	Monitors code quality. Provides a void to search for any bad functions. Hence reflecting our goals.

Table 1 List of Static Analysis tools selected/rejected.

IV. CODE SAMPLING

Time has always been the most important factor of any project and our literature is no exception. Due to the time constraint the entire code could not be inspected, only particular modules of code were selected. Based on perspective study we have selected particular components of the code for inspection. A perspective study guarantees that sufficient requirements were met and support for later stages of software development is given. Perspective study is the technique in which each individual of the team inspects the code in their own perspective.

Various studies implement that PBR [Perspective based review] has several beneficial characteristics because it is systematic, focused, goal oriented and transferable via training. The study was performed by assigning roles to the participants. Each individual plays the role of a tester, user and developer respectively and analyses the code. Due to our lack of experience, we have prepared a questionnaire and based on the answers obtained we have selected the important components of the code. After several discussions, necessary decisions were made regarding the selection of code. The selected code was then tested by our static automated tool

V. REVIEW PROCESS

Initially for the review process, we have followed a step by step process as follows

- Planning
- Kick-off
- Individual analysis
- Review
- Rework

The group is divided into 3 parts based on **Fagan's Inspection** [10]**.**

- Moderator
- Recorder
- Inspectors

The code investigation and survey was completed over a time of 23 days. Challenges have been overcome to form a group of 4 and we were supposed to download a zip file from Buddypress [11]. We being inexperienced in the field of PHP, sat down and referred to **W3SCHOOLS**. The sole reason behind referring w3schools was to get insights of PHP and present an unbiased report. This process step was followed by deciding the goals of inspection. After discussions among our group, we have voted for four inspection goals and moved further to search for the tools which will fulfill the goals that are set. Podio [12] and Microsoft excel being the tools that we used to track our progress, assessment of tools has been done for the static analysis.

Log minutes were also tracked to review the hours spent on working on analysis and review. After installing RIPS and VCG, we have also installed XAMPP server to run the tools on our computers. Each of us have scanned the PHP files and came up with different types of vulnerabilities. The different types of vulnerabilities can be seen in the excel sheets that are addressed as an attachment with this project assignment.

The **table number 2** below will give an insight as to how many minutes it took to perform the test in different tools that we have used.
As the survey process progressed, the time taken to scrutiny the vulnerabilities have reduced. This was followed by individual reflections on the vulnerabilities. We have recorded every one of the occasions that occurred during the time spent breaking down the code.

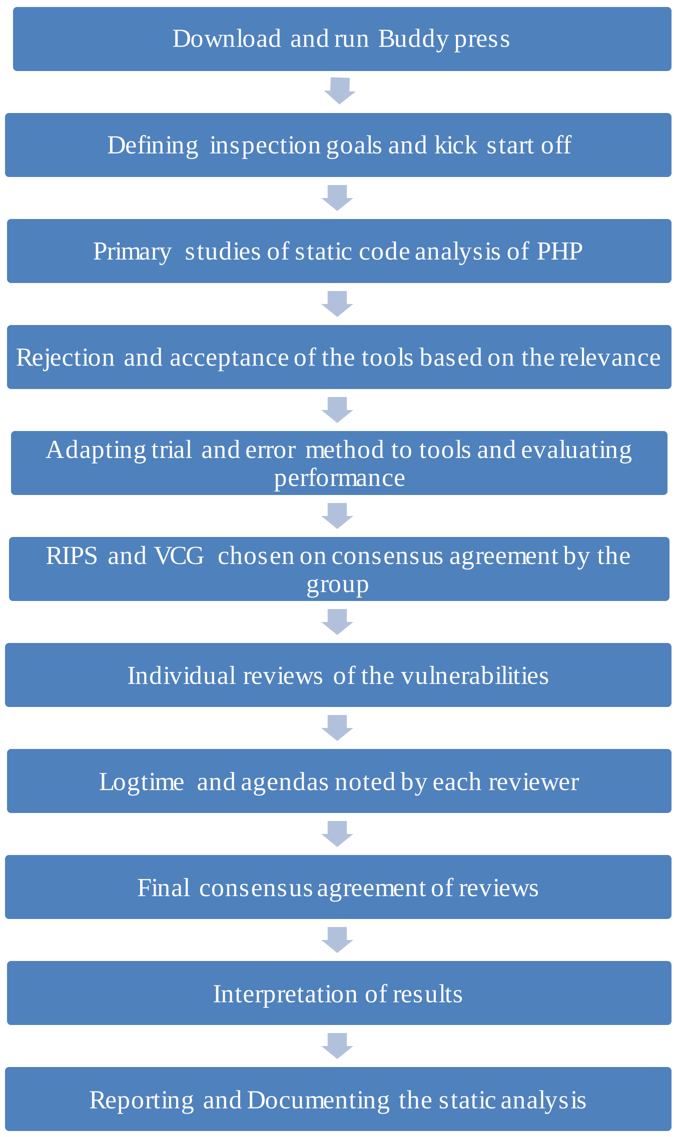
VI. ANALYSIS OF THE OUTCOMES

Investigating the outcome of the results after performing the static analysis, we have detected several vulnerabilities by using RIPS tool primarily. We have set an objective to review vulnerabilities that are presented by RIPS. After analyzing the vulnerabilities individually, we have classified the vulnerabilities on the basis of

(i) True Positive
(ii) False Positive
(iii) Uncertain.

A true positive is evidently true and can cause a potential threat to the software irrespective of the stage it is in. A false positive is an error but it will not bring any potential harm to the system and cannot be ignored. An Uncertain error in this context are the ones which we could not assess the impact. Lack of experience in this field is a major contributing factor for not being able to assess this kind of error. Further, we have come across few security vulnerabilities such as XSS attacks where sensitive sink is reached. Sensitive sink is where the data comes from the outside the script including Database-results, user inputs, file systems (typically, tainted data). While assessing these errors, we have also found and assessed other errors such as File Manipulations and Disclosures, PHP Flow Control and SQL Injection.

After individual assessment, we had a group meeting and we decided upon true and false positives. The attachment of the HTML file will give an insight of our consensus assessment. A generic flow chart will help understand the process of our Analysis and Outcomes.


Download and run Buddy press
Defining inspection goals and kick start off
Primary studies of static code analysis of PHP
Rejection and acceptance of the tools based on the relevance
Adapting trial and error method to tools and evaluating performance
RIPS and VCG chosen on consensus agreement by the group
Individual reviews of the vulnerabilities
Logtime and agendas noted by each reviewer
Final consensus agreement of reviews
Interpretation of results
Reporting and Documenting the static analysis

A Generic flowchart of the process of static analysis

VII. CONCLUSION

Subsequently, after static automated testing, we could make a few determinations addressing the vulnerabilities the tool has created and about the advantages and disadvantages of utilizing an automated static tool. The goals of inspection were satisfied by the tools that we have selected. We have talked about them here:

Impact of the tool:

RIPS was primarily selected on the basis of our three inspection goals for the analysis of open source PHP code provided by Buddypress. Visual Code Grepper has been chosen to fulfill our goal of reliability. These two tools have been selected from a list of several open source testing tools for PHP.

We have concluded that there were many scenarios of vulnerabilities being true positives and very less number of vulnerabilities have turned out to be under the classification of false positives and uncertain. Taking factors that might influence our literature are as follows.

- Possessing limited knowledge and has been a major challenge to assess the violations. Considerable possibility might be involved to contradict our assessment and could be a potential threat to our validation.
- Time being a constraint has limited our study and could be considered as a potential threat in acquiring more knowledge in this field.

Comparison of individual reviews:

Since two tools have been used, we have divided the amount of assessment equally. After assessing individually, all of us have gathered to decide on the category of the vulnerabilities that they belong to. In contrast to majority voting, we assessed and have taken the most logical explanation to the vulnerabilities into consideration. We had similar process of evaluation in most cases thereby landing decisions on the same categories that vulnerabilities belong to. Analogously, there were good number of cases where we have contradicted each other on the logical based evidence provided by each group member. There were certain errors such as XSS and SQL injection attacks which were decided with ease but debate was seen in few cases to decide upon in few cases since ambiguity was also present in the assessment. Since many errors were addressed, contexts have also influenced to certain extent to take a decision upon. For instance, Vulnerability is assessed as true positive at a certain Line Of Code (LOC) and also assessed as false positive at some other LOC. Few scenarios where no one had an idea as to which category they belonged were also present and are classified as uncertain.

Performance of the tools:

The inspection goals of the authors were clarity, finding bugs, consistency and reliability. Selected static tool by us, helped in the understanding of the dynamics of the PHP errors by the descriptions that have been provided. Compared to RIPS scanner, VCG has been optimal when time has been taken as a parameter whereas RIPS has been consistent throughout, showing the major bugs and classifying them accordingly. Both tools have been efficient enough in finding the bugs alike. Inspecting the code modules on the basis of inspection goals was carried out all throughout. RIPS when compared to VCG is less user-friendly. But, that challenge has been overcome to meet our four goals of inspection. We, primarily consulted internet forums and documentations regarding the working of the given two tools. Both tools have described the errors alike and provided us with a need to solve the errors.

Cost Effectiveness:

Log minutes and **agendas** for every meeting have been maintained by each reviewer in the process of auditing of each vulnerability that has been found. Reviewing for the first time has been concluded to take relatively more time due to the limited knowledge that we possess in regards to the usage of the static tool.

Eventually, time taken for reviews has decreased because of the familiarity that we gained in the usage of the tool. The important advantage of the static tools that we have used, is that relatively basic knowledge is enough to review the vulnerabilities presented by the tool. Compared to dynamic code analyzers, this aspect of static analyzers reduces the cost by a huge margin. Since the open source code has been used by us, anyone possessing programming skills can attack the source code and manipulate it. Mainly because of the two tools that we have used, unveiling of the bugs that have been lurking were also addressed. **We believed that experimentation has to be repeatable to be valid.** Through the tools we used observations have been made such as increase in the scope of errors when the functionality changes. These changes, in particular can be investigated using static code analyzers. Hence we surmise that static tools are less expensive when compared to dynamic test tools. However, static testing also gave us an insight about the dynamic analysis of the software which falls to be part II of our project.

Reviewer's Name	Time Taken for RIPS	
	Violations found	In seconds
Siddhartha Srinadhuni	72	1001 seconds
Madhukar Enugurthi	72	1100 seconds
GVS Anurudh	72	982 seconds
Divyani Pamulapati	72	2647 seconds

Table 2 Individual time log of defects

Reviewer's Name	Time Taken for VCG
	Violations found
Siddhartha Srinadhuni	1609
Madhukar Enugurthi	1609
GVS Anurudh	1609
Divyani Pamulapati	1609

Table 3: Violations found by each team member.

Part-II: Dynamic Analysis

1. INTRODUCTION

WordPress is an open source project which has a high degree of freedom when it comes to payment. WordPress has been the largest self-blogging tool since 2003 and is extensively used by millions all over the world [13]. As a part of WordPress, there are plugins and as of 2013, there are over 28000 plugins and over 2000 themes [14]**.** It is by far, the easiest and the most powerful website content management system in existence, today. In part II Analysis of the open source code is done dynamically. The selection of Selenium IDE and OWASP ZAP is done to carry out this dynamic analysis. Selenium IDE being a web testing tool runs directly with in a browser [15]**.** This also enables the user to map their way to applications in Mozilla Firefox, to perform tests and other actions.

The test plan is prepared on the basis of penetration and acceptance testing. The tools have been chosen inline to our tools and further rationales have been documented below, as it is essential to produce good test inputs to test software dynamically, we present test cases of both the tools below. Pre and post conditions are major criteria for the testers, to test a state of art. They provide significant inputs to the testers whilst evaluation of the subject. We have carried out a dynamic analysis to portray the vulnerabilities that are in the open source PHP code and presented detailed conclusions of the same.

This document is divided into the following sections: Section I presents the introduction. Section II describing the scope and the background. Section III investigates about the inspection goals while Section IV includes the selection of tools. Section V involves Results and Analysis and further references are included in Section VI.

II. SCOPE

The prime objective of this assignment is to conduct a dynamic analysis for the open source web application that is available through WordPress. We depict a detailed testing methodology that enables the practitioners to identify the defects, correct them and have a reduction policy to deal with it in the case, if it ever come across again in the foreseeable future.

With respect to the fact that the application under test is open source, the scope of the test plan is limited to Acceptance testing and Penetration testing for the following reasons

III. INSPECTION GOALS

The group has progressed to consensus agreement of the inspection goals of security and reliability before concluding which test to perform. Based on these two inspection goals, we have selected the automated testing tools in the further testing process.

- **Security:** Security has always been a major concern in today's software systems and ensuring it to be checked is essential. The national vulnerability database lists that more than 20000 security problems are prevalent, these days. We intended to investigate the security through penetration testing and hence it comprises one of our inspection goals for this task [16]
- **Reliability:** As testing is an important part of Software Development Life Cycle, having dodgy tools will not help a tester to present a qualitative product [17]. Hence a tool requires to be reliable. With considering the fact that robustness and performance are also checked with reliability as our inspection goal, this has been the only motivation behind taking up 'reliability' as our inspection goal. This concludes our inspection goals.

Depending upon these inspection goals, the search for tools which are primarily inline to our goals was initiated. After narrowing the search on different basis (IN SECTION), we have selected OWASP ZAP and Selenium IDE as our tools.

Test Conditions:

- We have carried out the test process based on the test strategy that was tailored according to acceptance testing and penetration testing.
- The choice of the testing was based on the Perspective Based Reading technique that we have employed.
- With the limited resources both in terms of code and people, we have conducted the dynamic testing in our personal laptops.

IV. BACKGROUND

The different tests that are selected/rejected and the rationales behind them are listed down below.

- **Acceptance testing**: This is a formal test conducted to determine whether a system suits the acceptance criteria-from a point of view of a user. This test plan includes user acceptance testing because the features that are tested are from a user's perspective[18]. It places the user to control the criteria and determined whether or not to accept the system. Acceptance testing helps the development of the software and it becomes more flexible eventually during the life cycle of a software. Therefore, as mentioned earlier, we have identified key features assuming various roles as a part of Perspective Based Reading. It is again, using the same strategy that we could come to certain attributes that are accepted by the end user. The major motivation behind selection of this test is that in our project to test dynamically, WordPress, performing an automated acceptance test was the most practical choice.
- Security breaches is most predominant these days and we intended to carry out a test that question the breaches that are present and **Penetration Testing** provides us a void to investigate the security concerns. It can also check the safety of a network system or a web application from the perspective of an attacker. The process surmises the imitation of the attackers by the testers to simulate a real attack on the target system[19]. A penetration testing assesses the security. The vulnerabilities that are questions could be flaws such as end user behavior, application flaws and the quality of the defensive mechanisms. This prompted us to take up penetration testing as our goal of inspection for this particular assignment.

- **System Testing**: A stage of testing a software where the system, that is integrated, is tested as a whole, typically is known as System testing. Testing the key functionalities is comprehended by a system test. The major motive of system testing, is investigating defects as a whole. Testing for the defects on a security level is what we thought should be a rationale and hence we have not adopted system testing as our test strategy.

- **Integration testing** is not implemented as it is not suitable because the application, from our understanding, has frequent changes. As the users are free to add and modify the features that are existing, testing each module is not feasible and lacks proper structure in maintaining modules

The above justifications form our rationales for the choices we made for the tests. The one major concern through our perspective was that we are users without industrial experience. We have no absolute background of testing. It is through learning several reliable literatures that we have achieved testing. The only possible threat to our task could be our limited knowledge in this particular area. Despite of this threat, we tried the best of our abilities to put up reliable results. We have conducted a Perspective Based reading and have prioritized based on the requirements on which test to conduct. The limited time constraint has always been an influence and the finalization of the selection of the test plan might not be the be best in terms of yielding the results.

IV. SELECTION OF THE TOOL

Manual effort is largely reduced by the automation of the generated test cases[20]. Since, the test is in the context of PHP, the group has conducted a search for dynamic tools which tests the PHP code. The search has been narrowed down to Selenium IDE, Veracode, Metasploit, PHP Unit, PHP Spec and OWASP ZAP. The motivations behind choosing and rejection of these particular tools is listed in Table number 4.

Reasons for selection of Selenium IDE:

- No official experience with programming or scripting is required to test via Selenium IDE tool.

- Selenium IDE has been consistently rapid when it comes to testing. Comparative to other tools which provide a void for testing acceptance, Selenium is relatively a fast tool.

- Not only provides a user friendly environment, but also runs with different web browsers Internet Explorer, Firefox, Opera and other phone browsers as well [21].

- Places the user in control to develop test cases in a graphical environment by simply interacting with the Web application under test [15].

- Allows the user to modify to arguments or add more commands to the already recorded test cases.

- We, implementing it as a Firefox plugin, provides the easiest way to develop or generate test cases.

S.No	Tool	Accept/Reject	Motivation
1	Veracode	Rejected	Within our inspection goal of penetration testing but, the group had to reject it because of it being a commercial tool.
2	PHP Unit	Rejected	A tool for unit testing of PHP. Not inline to either of our goals of inspection. Hence rejected.
3	Selenium IDE	Accepted	Within our inspection goals of Acceptance testing. Provides a user friendly environment to generate test cases. Hence accepted
4	PHP Spec	Rejected	A tool used for inspection of Unit testing in PHP. Not Inline to our goals of inspection and therefore rejected.
5	Metasploit	Rejected	In-line to our goals in penetration testing. The group has rejected Metasploit as it requires a permission to acquire the license and takes around 14 days. Time constraint has been a challenge and this tool is hence ruled out [22]
6	OWASP ZAP	Accepted	Most popular tool free security tool. Open source web application scanner and In-line to our inspection goal of penetration testing.

Table 4 List of Dynamic Analysis tools

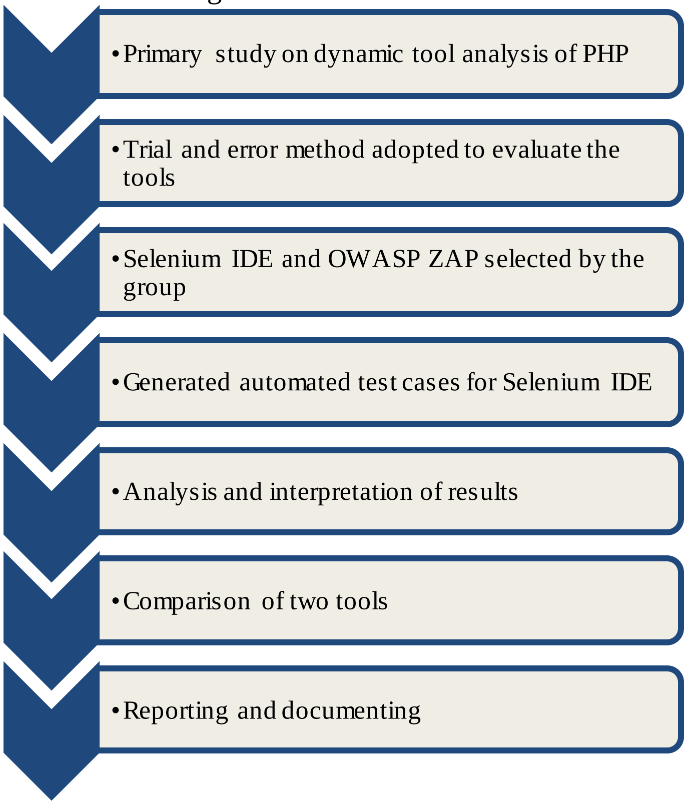
Reasons for selection of OWASP ZAP:

- OWASP Zed Attack Proxy (ZAP) checks every functionality of an Application Under Test (AUT) without peeing into its internal structure [23].
- Several studies have shown that OWASP ZAP is easy to use integrated Penetration testing in web applications.
- Classifies the vulnerabilities into high, medium and low vulnerabilities for the better understanding of the user.
- The number of true and false positives that this tool yields out is reliable.
- OWASP ZAP is certified to be better than any of the web vulnerabilities scanners.

Latest and relevant tools could have been included because of the commercial license that is required and limited time.

V. REVIEW PROCESS

After the completion of the static analysis, we have moved forward to take up dynamic analysis part of the given assignment. The process of dynamic analysis was initiated and search for the most common tools for acceptance and penetration testing, after agreements among the group. The threshold studies have shed some light on the dynamics of the working of different tools. Google search engine was used for the primary studies about the dynamic tools. The steps in our review process can be reflected in the figure below.


• Primary study on dynamic tool analysis of PHP
• Trial and error method adopted to evaluate the tools
• Selenium IDE and OWASP ZAP selected by the group
• Generated automated test cases for Selenium IDE
• Analysis and interpretation of results
• Comparison of two tools
• Reporting and documenting

A Generic flowchart of the process of Dynamic analysis

VI. TEST MODEL

From the tester's perspective, different aspects of testing like test design, test cases, test data, test plans are specified.

i) *Test Strategy:* Our test strategy is tailored in such a way that it achieves maximum branch coverage and validating the quality and determine their errors in the test cases created for each module in acceptance testing. For Penetration testing, running the application and directing the path to buddy press has been done to achieve maximum branch coverage.

ii) *Test cases:* For acceptance testing, certain test cases have been generated such as logging into WordPress with a wrong Password, sending a public and a private message, posting a thread and changing the themes. Selenium IDE is used to record the AUT and the results are shown below.

iii) *Test Execution:* Testing the artifact against the set of cases that have been generated to inspect the defects that are present. The results of the test of it being passed or failed has been provided with relevant information to verify the expected outcome against the desired outcome. The test cases were developed such that the cases could be repeatable and independent of the tester. Hence the implementation of the test cases was executed at this stage.

iv) *Test Analysis:* Analyzing the results towards the end of the test process, the errors under various test case environments. This stage involved all of the team members to review the type of error that is there, its occurrence and its influence on the AUT. During this stage of analysis, the vulnerabilities have presented us with valuable inputs in understanding them. Analyzing the test results and concluding if it is a true or a false positive is the basis of this approach of analyzing the test.

The above mentioned steps were implemented as a part of the test approach to implement the test plan. Typical steps involved in a software testing life cycle (STLC) have formed the basis for the test strategy that we implemented [24].

VI. RESULTS AND ANALYSIS

As a part of this project assignment, we are to compare the two dynamic tools and hence, we needed few grounds of comparisons. The following are the **grounds of comparisons** upon which we have compared our two dynamic test tools.

- Results and Analysis
- Features and Ease of use
- Impact of the tools

Results and Analysis:

i. OWASP ZAP:

As this tool is used for penetration testing, this tool shows the severity of the vulnerabilities that are present in the source code. It also segregates the errors in High, Medium and Low severity.

- After primary studies, on this Web Scanner, the tool was run and has detected 2422 errors.
- This investigates the web vulnerabilities such as Application Error Disclosure, Directory Browsing, X- Frame Options Header Not Set flagged as Medium severity.
- XSS Protection Not Enabled and X-Content Type options header missing as Low severity. On further study, we have interpreted
- Classification of the vulnerabilities yielded out are shown in the Table number 5.

Vulnerability	Percentage of errors present	Severity
Directory Browsing	2.3%	Medium
Application Error Disclosure	20.72%	Medium
X-Frame Options Not Set	25.6%	Medium
Web Browser XSS Protection Not Enabled	25.6%	Low
X- Content type Options header missing	25.6%	Low

Table 5 : Results of OWASP ZAP

ii. Selenium IDE:

This being a testing tool for user acceptance testing, we played the roles of users to generate test cases from a user's perspective. Certain test cases have been taken into consideration whilst testing the widget of WordPress. The following are the test cases that have been taken to test the AUT.

- Inputting the right username and wrong password has been our primary test case scenario.
- Posting a public message to other users on WordPress.
- Changing the theme of the user's WordPress account.
- Publishing a page on WordPress has been another test case.
- Posting a 'post' on WordPress concludes

our test cases.

These test cases were taken into consideration as they are the most used modules a user would use. Hence, we have taken these test cases for our consideration. Results can be seen in the table below.

Test case ID	Title	Purpose	Results expected	Outcomes	Tested By
TC1	Inputting a wrong password	Credentials to log in to the page	Show an error message and unable to log in	TC1 Passed	Sisr16
TC2	Changing the theme	Changes the background of the interface	Theme should differ from the existing one	TC2 Passed	Maen 16
TC3	Posting a page	Post a page on the website	A page to be posted as an outcome	TC3 Passed	Dipa16
TC4	Posting a public message	Sending a public message	A public message is sent	TC4 Passed	Vegu16

Features and Ease of Use:

i. OWASP ZAP:

Based on the study and usage of the dynamic analysis tools by the respective teams, OWASP ZAP has been chosen to perform the dynamic analysis for penetration testing. This tool is in line to security, our major inspection goal. Compared to other tools, OWASP ZAP provides wider range of features in testing. The one of a kind feature about OWASP is what is known as a "Spider" which enhances the scanning and detects new URL's on the AUT. Coverage of the whole AUT is possible with this tool and hence we have selected this tool as a part of analysis. [19][25]

ii. Selenium IDE

With that being said, Selenium IDE is an easy Firefox Plugin and is used to develop Selenium test cases. It is said to generate most efficient test cases with the recorder that it provides. One of the important and unique features of Selenium is "Log" where the error and information messages showing the progress are displayed automatically and "Record" is used to check for the recording for the user interaction with the website. Hence, these two tools are used for this part of dynamic analysis[26].

Impact of Tools

i. *OWASP ZAP:*

Being a test for efficient penetration testing, this tool is very reliable when it comes to technical and business impacts. As the goal of the tool is to estimate the vulnerability that has been exploiting the system, traditional security concerns such as confidentiality, integrity, availability and confidentiality are maintained. Fixing the errors would need very less configuration but will have a much greater impact on the security of the web application

ii. *Selenium IDE:*

One of the popular tools for acceptance testing, Selenium IDE provides valid results for user interaction with the website. The impacts include flexibility from the user's perspective, debugging at a large scale and set breakpoints for recording. It spreads wider horizons of testing as there is no programming language experience involved.

VIII. THREATS TO VALIDITY

There are certain threats of validity for this assignment that we would like to put forth. The following are the threats to validity.

- Being the novice testers that we are, might bring certain influence on the outcomes.
- Time has always been a constrain, starting from finding groups to concluding this assignment.
- Given the fact that we are new to this field, primary studies on PHP has taken considerable amount of efforts and time.
- Literature support has not been provided to a certain extent.

POSTMORTEM ANALYSIS:

We have followed [27] to do a post mortem analysis of this project assignment. We, as a group have started planning which is followed by searching for the tools and validating if they are suitable for our inspection goals and then we considered every other's feedback on how this project assignment can be better.

IX. CONCLUSIONS

By performing the static code analysis on the open source PHP code, using RIPS and VCG, we inferred that the code had good attributes of clarity and quality with wide scope of finding vulnerabilities in the perspective of security. There was also a void to improve the consistency as well. Given the focus on dynamic code analysis, we identified that the security and reliability have been major concerns. Of the two tools, OWASP ZAP and Selenium IDE, they have addressed both the concerns respectively.

For producing and developing high quality software, both static and dynamic testing always play an integral part and we conclude that they are inevitable for all the software development.

REFERENCES

[1] T. Koskinen, P. Ihantola, and V. Karavirta, “Quality of WordPress plug-ins: an overview of security and user ratings,” in *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, 2012, pp. 834–837.

[2] L. Atkinson and Z. Suraski, *Core PHP programming*. Prentice Hall Professional, 2004.

[3] H. Al-Kilidar, K. Cox, and B. Kitchenham, “The use and usefulness of the ISO/IEC 9126 quality standard,” in *2005 International Symposium on Empirical Software Engineering, 2005.*, 2005, p. 7 pp.–.

[4] R. Plosch, H. Gruber, A. Hentschel, G. Pomberger, and S. Schiffer, “On the relation between external software quality and static code analysis,” in *Software Engineering Workshop, 2008. SEW’08. 32nd Annual IEEE*, 2008, pp. 169–174.

[5] “Free Code Analysis Tools – A Step in the Right Direction,” *Checkmarx*, 15-Oct-2014. .

[6] “RIPS,” *SourceForge*. [Online]. Available: https://sourceforge.net/projects/rips-scanner/. [Accessed: 14-Mar-2016].

[7] “RIPS - free PHP security scanner using static code analysis.” [Online]. Available: http://rips-scanner.sourceforge.net/#about. [Accessed: 03-Apr-2016].

[8] “VisualCodeGrepper V2.0.0,” *SourceForge*. [Online]. Available: https://sourceforge.net/projects/visualcodegrepp/. [Accessed: 14-Mar-2016].

[9] N. J. Christopher Kruegel, “Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities.” [Online]. Available: http://www.seclab.tuwien.ac.at/papers/pixy_techreport.pdf. [Accessed: 18-Apr-2016].

[10] M. Fagan, “Reviews and inspections,” *Softw. Pioneers–Contributions Softw. Eng.*, pp. 562–573, 2002.

[11] L. Sabin-Wilson, *BuddyPress For Dummies*. John Wiley & Sons, 2010.

[12] “Podio - There’s a Better Way to Work,” *Podio*. [Online]. Available: https://podio.com/en. [Accessed: 18-Apr-2016].

[13] “Blog Tool, Publishing Platform, and CMS — WordPress.” [Online]. Available: https://wordpress.org/. [Accessed: 18-Apr-2016].

[14] L. Eshkevari, G. Antoniol, J. R. Cordy, and M. Di Penta, “Identifying and locating interference issues in php applications: the case of wordpress,” in *Proceedings of the 22nd International Conference on Program Comprehension*, 2014, pp. 157–167.

[15] “Web Application Testing with Selenium.” [Online]. Available: http://delivery.acm.org.miman.bib.bth.se/10.1145/1770000/1767729/10601.html?ip=194.47.129.64&id=1767729&acc=ACTIVE%20SERVICE&key=74F7687761D7AE37%2EA87D2C672813D63B%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=598807856&CFTOKEN=63491868&__acm__=1460213889_d350d2c18a1e9a4b86bc3a7386237761. [Accessed: 09-Apr-2016].

[16] F. Elberzhager, A. Klaus, and M. Jawurek, “Software Inspections Using Guided Checklists to Ensure Security Goals,” in *International Conference on Availability, Reliability and Security, 2009. ARES ’09*, 2009, pp. 853–858.

[17] S. M. . Q. Sheikh Umar Farooq, “Software Testing – Goals, Principles, and Limitations.” [Online]. Available: http://www.ijcaonline.org/volume6/number9/pxc3871448.pdf. [Accessed: 08-Apr-2016].

[18] B. Haugset and G. K. Hanssen, “Automated acceptance testing: A literature review and an industrial case study,” in *Agile, 2008. AGILE’08. Conference*, 2008, pp. 27–38.

[19] X. Qiu, S. Wang, Q. Jia, C. Xia, and Q. Xia, “An automated method of penetration testing,” in *2014 IEEE Computing, Communications and IT Applications Conference (ComComAp)*, 2014, pp. 211–216.

[20] M. Palacios, J. García-Fanjul, J. Tuya, and G. Spanoudakis, “Automatic test case generation for WS-Agreements using combinatorial testing,” *Comput. Stand. Interfaces*, vol. 38, pp. 84–100, 2015.

[21] I. Altaf, J. A. Dar, F. u Rashid, and M. Rafiq, “Survey on selenium tool in software testing,” in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015, pp. 1378–1383.

[22] “Penetration Testing Software, Pen Testing Security,” *Metasploit*. [Online]. Available: https://www.metasploit.com/. [Accessed: 18-Apr-2016].

[23] Y. Makino and V. Klyuev, “Evaluation of web vulnerability scanners,” in *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 2015, vol. 1, pp. 399–402.

[24] J. Tang, “Towards Automation in Software Test Life Cycle Based on Multi-Agent,” in *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, 2010, pp. 1–4.

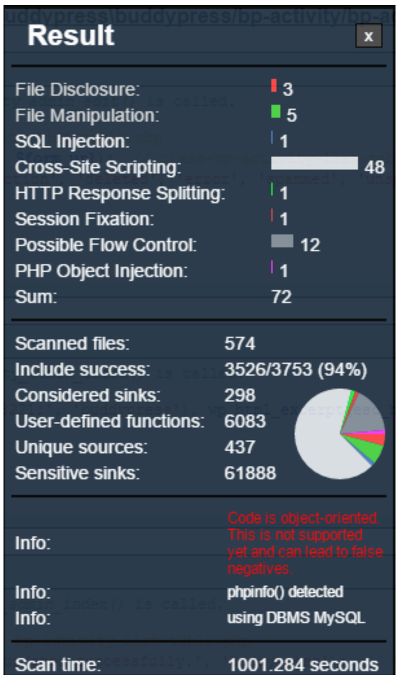
[25] “OWASP Zed Attack Proxy Project - OWASP.” [Online]. Available: https://www.owasp.org/index.php/OWASP_Zed_A

ttack_Proxy_Project. [Accessed: 18-Apr-2016].
[26] "Selenium - Web Browser Automation." [Online]. Available: http://docs.seleniumhq.org/. [Accessed: 18-Apr-2016].
[27] M. J. Tiedeman, "Post-mortems-methodology and experiences," *IEEE J. Sel. Areas Commun.*, vol. 8, no. 2, pp. 176–180, Feb. 1990.

Group Member Participation

Participants	No. of hours	Percentage of Contribution
Siddhartha Srinadhuni	65	35%
Venkata Sai Anurudh Gudivada	35	25%
Divyani pamulapati	20	15%
Madhukar Enugurthi	40	25%

APPENDIX A : Results of RIPS scanner and VCG



Result

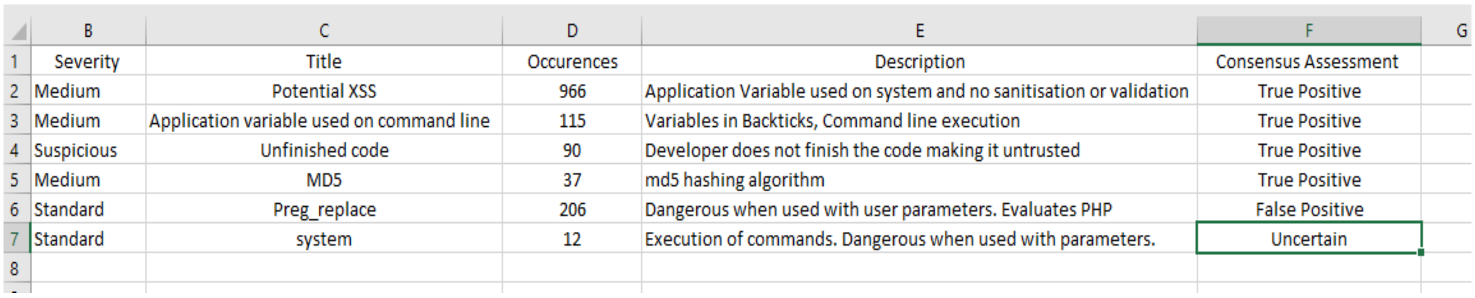
File Disclosure:	3
File Manipulation:	5
SQL Injection:	1
Cross-Site Scripting:	48
HTTP Response Splitting:	1
Session Fixation:	1
Possible Flow Control:	12
PHP Object Injection:	1
Sum:	72

Scanned files:	574
Include success:	3526/3753 (94%)
Considered sinks:	298
User-defined functions:	6083
Unique sources:	437
Sensitive sinks:	61888

Info:	Code is object-oriented. This is not supported yet and can lead to false negatives.
Info:	phpinfo() detected
Info:	using DBMS MySQL

Scan time:	1001.284 seconds

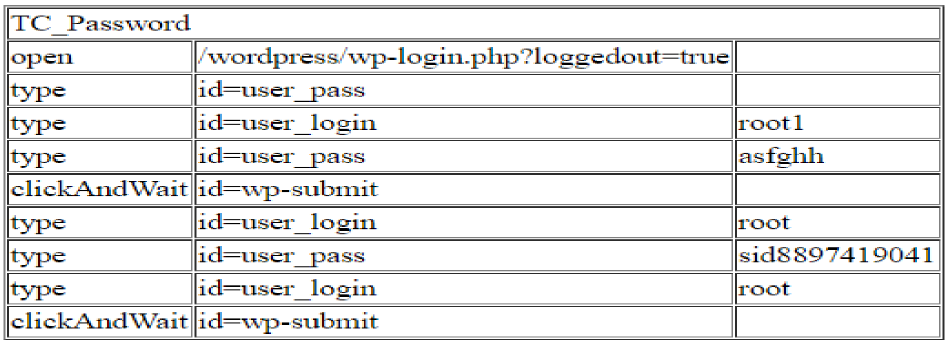
Results of RIPS scanner



	B	C	D	E	F
1	Severity	Title	Occurences	Description	Consensus Assessment
2	Medium	Potential XSS	966	Application Variable used on system and no sanitisation or validation	True Positive
3	Medium	Application variable used on command line	115	Variables in Backticks, Command line execution	True Positive
4	Suspicious	Unfinished code	90	Developer does not finish the code making it untrusted	True Positive
5	Medium	MD5	37	md5 hashing algorithm	True Positive
6	Standard	Preg_replace	206	Dangerous when used with user parameters. Evaluates PHP	False Positive
7	Standard	system	12	Execution of commands. Dangerous when used with parameters.	Uncertain
8					

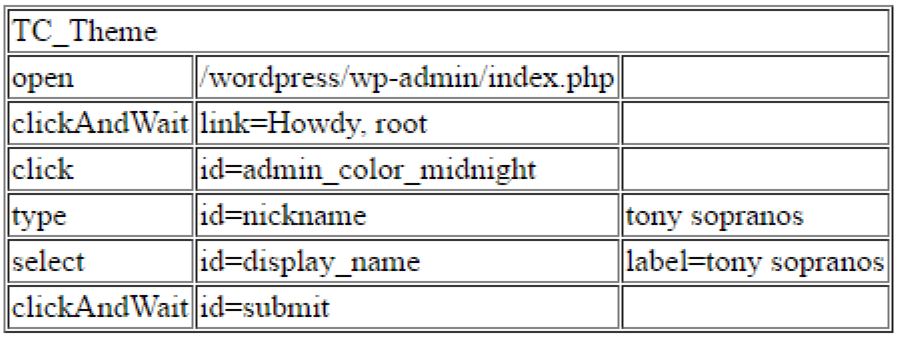
Results of Visual Code Grepper

APPENDIX B: Results of Selenium IDE and OWASP ZAP



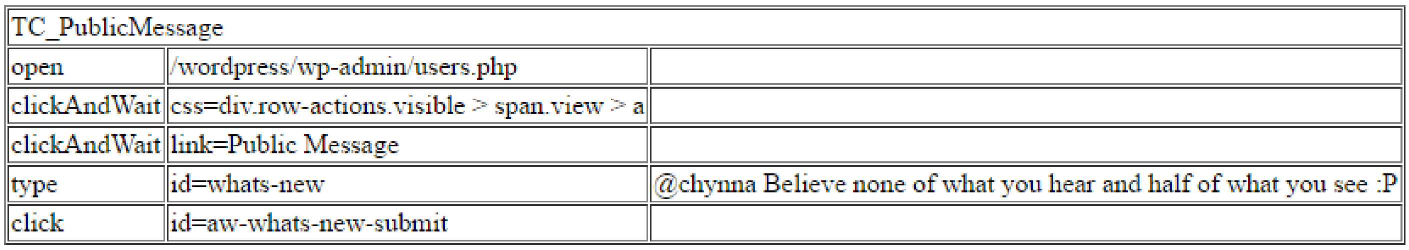
TC_Password		
open	/wordpress/wp-login.php?loggedout=true	
type	id=user_pass	
type	id=user_login	root1
type	id=user_pass	asfghh
clickAndWait	id=wp-submit	
type	id=user_login	root
type	id=user_pass	sid8897419041
type	id=user_login	root
clickAndWait	id=wp-submit	

Test case of inputting wrong password



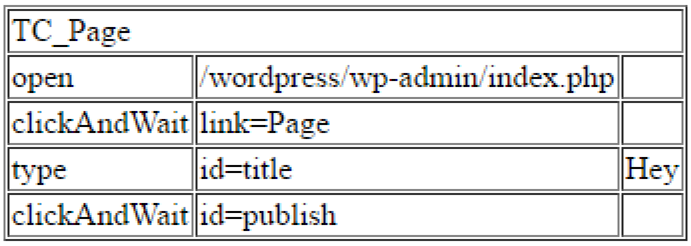
TC_Theme		
open	/wordpress/wp-admin/index.php	
clickAndWait	link=Howdy, root	
click	id=admin_color_midnight	
type	id=nickname	tony sopranos
select	id=display_name	label=tony sopranos
clickAndWait	id=submit	

Test case of changing the theme



TC_PublicMessage		
open	/wordpress/wp-admin/users.php	
clickAndWait	css=div.row-actions.visible > span.view > a	
clickAndWait	link=Public Message	
type	id=whats-new	@chynna Believe none of what you hear and half of what you see :P
click	id=aw-whats-new-submit	

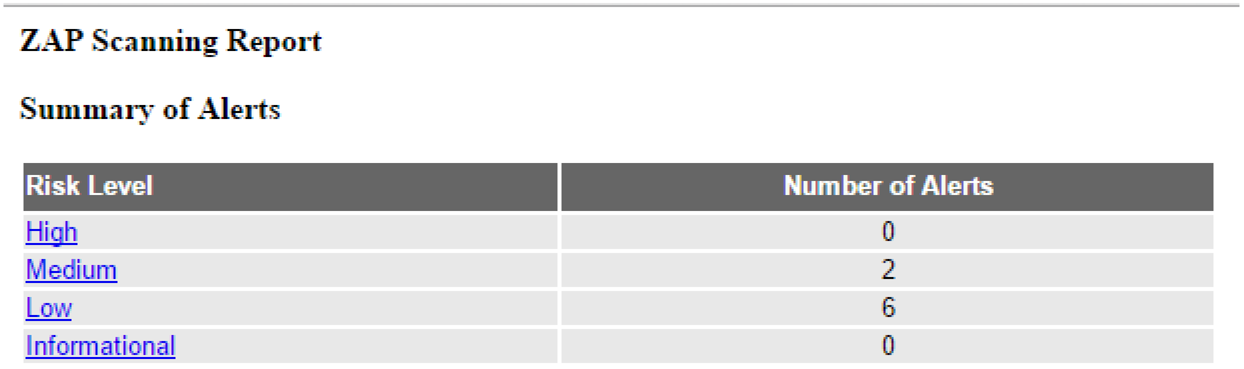
Test case of posting a public message



TC_Page		
open	/wordpress/wp-admin/index.php	
clickAndWait	link=Page	
type	id=title	Hey
clickAndWait	id=publish	

Test case of publishing a page

RESULTS OF OWASP ZAP



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	2
Low	6
Informational	0