ABSTRACT
The number of potential vulnerabilities and threats are increasing for software. Security testing is an essential process to confirm software security requirements and to identify security related vulnerabilities. In this paper we use analytical hierarchy process (AHP) to analyze and compare different available security testing techniques based on a pre defined criteria. Five testing techniques were selected which includes Source code analysis, Fault code injection, Robustness, Stress and Penetration testing techniques. They have been evaluated against five criteria which include cost, thoroughness, ease of use, effectiveness and efficiency. The outcome of the study is helpful for researchers, testers and developers to understand effectiveness of each technique in its respective domain. Also the study helps to compare the inner working of testing techniques against a selected criterion to achieve optimum testing results.

KEY WORDS
Software security, Security testing, testing techniques, AHP

1. Introduction
Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item [3,4]. In the process of testing software item is passed under specified conditions to observe it for particular aspects. There are two main goals of software testing one objective is that to probe the software for bugs so that these can be removed, and to ensure that the software works according to specifications. Software errors and defect give rise to vulnerabilities, which is the main cause of software failure. Most of the software contains flaws and errors that are often exploited to compromise the functions and security of the software. Software security assurance is an evolving subject and is much less mature than software quality assurance and software safety assurance. Software security assurance objective is to ensure the confidentiality, integrity and availability of software system by following different techniques and mechanism throughout the software development life cycle SDLC. Security testing activities are performed to validate security requirements and identify potential vulnerabilities. Standard software processes identifies all types of related to software quality attribute and software functional aspects but security vulnerabilities can also be discovered through standard testing process. The objective of the security testing is to assess security properties and behavior of the software as it interact with the external or internal entities interact regardless of the functionality that software implements. Functional testing is performed to ensure that the software behaves as intended but fortunately it also contributes towards the identification of software security related bugs. In this paper we used different types of functional testing techniques to discover software security issues, analyze them, and compare them in term of a predefined criterion.

We choose five types of functional testing techniques which include both black box and white box approaches, these includes Source code analysis, Fault code injection testing, Robustness testing, Stress testing and Penetration testing techniques. These techniques are first analyzed to understand how they work and how these can be used to identify security related vulnerabilities and bugs in software systems. Than these techniques have been compared based on a criteria which we think will help the software security testers and researches to select the optimum tool in particular scenario. Multi criteria decision support system MCDM based on analytical hierarchy analysis AHP has been used to evaluate the selected testing techniques. AHP is a structured based on mathematics and intuitive developed by Thomas L. Saaty
in 1970s [8] and has been extensively used in fields such as government, business, industry, healthcare, and education. AHP enables the evaluation of inconsistency of the decision-maker known as consistency check, inconsistencies below 10% are accepted for matrices of the range $n \geq 5$ (5% for $n=3$ and 9% for $n=4$). Otherwise, the judgments made must be revised or the matrix discarded [1]. The study also helps to know the relationships of known vulnerabilities and how particular testing techniques deal with it.

2. Security Testing Techniques

We have selected the five major types of testing techniques that perform crucial role in security enhancements, both white box and black box techniques are included in the selected techniques. Although some of the techniques behave as hybrid, the following section analyze the selected techniques and terms of its working mechanism, performance, types of security flaws identified and effectiveness.

2.1 Source code analysis

Source code analysis is the process of analyzing the source code, before compilation (static analysis) or analyzing the both source code and executable (dynamic analysis), for coding errors, insecure practices and vulnerable code. In manual code analysis the tester inspect the source code for vulnerable code such as finding strcpy () functions without the use of a tool. But modern security testing analyzers are much more sophisticated in terms of identifying bugs, it also reduces false alarms. Although automatic testing tools facilitate and enhance the process of analyzing the code but these tools are not capable of replacing a security analyst. In dynamic source analysis the compiled executable is run and feed as input for testing the program variables in order to detect code behavior. Depending on the type of testing tool some errors and discrepancies are identified but some are harder to be identified. There is a tradeoff in terms of effectiveness and efficiency between missed vulnerabilities (false negative) and false alarms (false positive). Many testing experts would like to scan vulnerabilities as much as possible and that is why the false negative increases the effectiveness instead of generating false alarms. But to reduce the cost of testing in terms of time the testing tools must reduce the false positive and false negative simultaneously.

Source code analysis is an impressive way of detecting vulnerable code and errors, following section identified the types of errors or insecure code which can be identified through source code analysis.

- Source code analysis tools [5] has the ability to examine calls in the argument to insecure library functions, e.g. the C/C++ testing tools have the ability to preprocess the source code which enable the analyzing tool to see the same code as seen by compiler.

- Bound detection and checking error functionality enable these tools to detect vulnerabilities due to integer overflow, integer truncation and unsigned underflow etc.

- Attackers use heap overwrite mechanism to cause buffer overflow in allocated chunks of memory by misusing the link lists. There are several variants of this vulnerability and has serious threat for software security, source code analyzer has the capacity to detect memory allocation errors.

- To detect vulnerabilities associated with incorrectly implemented sequences of operations, security analyzers often look for specific library function calls and print a warning about potential security problems associated with those functions.

- Source code tools use data flow analysis to false negative and positive by examining whether particular buffer overflow code is exploitable or not.

- Pointer aliasing is a static analysis that tries to solve the problem when two pointers pointing point to the same data.

Example: following code shows the scenario where the source code analyzer identify vulnerabilities [5,6].

```c
char *strcpy(char *str1, char *str2) {  while (*str2) *str1++ = *str2++;  return str2; }

int main(int argc, char **argv) {   char *buffer = (char *)malloc(16 * sizeof(char));   strcpy(buffer, argv[1]);   printf("%s\n", buffer);
   return str2; }
```
The above code is both susceptible to buffer overflow (by using strcpy()) and bound checking errors. These types of bugs can be easily identified by source code analysis tools an mention earlier.

2.2 Fault code injection

In this type of testing the bugs are intentionally injected into the code, the code is then compiled and executed so the tester can determine how software reacts when it is forced in anomalous states. Fault code injection increases the robustness and reliability by identifying incorrect use of pointers and arrays, the presence of dangerous calls and race condition. This type of testing is used in situation where high assurance is required against well known serious vulnerabilities but is a complex process because every scenario cannot be simulated. Fault propagation analysis it is not only observed that how code behaves with injected faults but it is also the propagation of the fault (in the source code) is analyzed through fault trees. This enables the tester to determine the impact of a fault on a module, and system as a whole. Interface propagation analysis enables the tester to determine how a fault in one component affects other component of the system.

2.3 Stress testing

Also known as load or performance testing, in stress testing the system is passed through stressful states to expose vulnerabilities arises as a result of when software are exposed to maximum design load and beyond it.

2.4 Software penetration testing

Penetration techniques have long been used in network security but this testing technique has also made it place to penetrate software systems for faults and bugs. Software penetration testing is the type of black box which focuses on vulnerabilities having external access. The idea of penetration testing is more like ethical hacking that is “attempt to compromise the security of the systems under test”. Penetration testing helps to expose complex vulnerabilities e.g. vulnerabilities arises as a result of inter and intra component communication or communication of software to its resources and environment. In software security one of the vital activities is to increase the test coverage and penetration tests can be more extensive in its coverage. Penetration testing currently faces two major challenges that is a push towards automation and minimizing the cost in term of labor time associated with test cases. Host based penetration tools have direct access to each part of the system due to the presence of these systems inside the host in contrary to network base tool. Although in penetration testing the systems is seen as an outside attacker might see it and is therefore consider as black box mechanism but it can also be used in white box fashion. Penetration testing is now widely being used and is advancing towards general software testing but this type of testing has its own pros and cons.

2.5 Vulnerability scanning

In this type of testing the software is scanned for well known vulnerabilities based on repository of “signatures” to observe software’s behavior associated with attack pattern. Host based scanners sophisticatedly analyzed the internal of the system such as the insecure configuration, while network based scanners are good to analyzed attack carried out from outside remotely. Vulnerabilities scanners exercise vulnerabilities on the target system, it has the ability to probe every network service and applies all available “signatures”. Scanners observe the application for vulnerabilities like buffer overrun, cookie manipulation, Sql injection and cross site scripting etc. vulnerability scanner works in black box manner and can be used only against small set of attack pattern.

3. Methodology and Implementation

Multi-criteria evaluation is a fundamental step of the rational decision-making process in order to gain reliable information on strengths, weaknesses and overall utility of each option. The purpose of our study is to identify and analyze the strength and weaknesses of security testing techniques in particular direction. The process is several steps including selecting a goal, list criteria/ sub criteria, determining the alternatives, assignment of priorities, calculation of weights, results and discussions. These steps have been explained in the following section.

3.1 Selecting goal and objectives

The goal of this work is to analyze the role security testing techniques to based on a criteria/sub criteria to help the testers in applying these technique according to requirements efficiently.
3.2 Criteria/sub criteria to evaluate testing

Software tester and professionals have different option available to test software at different level of abstraction; depending on the security requirements of the system testers normally prioritize the security tasks. We have selected five key criteria (standard) to evaluate software testing mechanisms against them. The first criteria is cost. It has been selected due to the fact that cost of testing process plays an important role in the selection of testing type due to the usual budget limits and restrictions. The second criteria is Thoroughness. Thoroughness is very important for applications where security is given a high priority because of its high coverage to parts of software. Ease of use is the third criteria. It is related to the required time to master the testing and hence the total cost of the testing process in terms of time and budget. Fourth and fifth criteria are effectiveness and efficiency respectively. They have been chosen because of time and cost where much better testing results can be obtained with minimal cost. It is very important to notice that most important factor here is the cost. This is because many of testing tasks are ignored or omitted because of their high costs and accordingly more and more bugs will be identified when software is in production time. The following section briefly introduce those criteria.

3.2.1 Cost

To use a particular technique it is vital to understand its cost in term of skills required, labor time to develop and execute test cases, tool and utility support and integration. We have two sub criteria 1. Skills required 2. Testing time, the sub criteria contribute to main criteria.

3.2.2 Thoroughness

Thorough check that every segment of software has been tested is required to secure it, it also encompasses that every possible interaction during runtime has been covered. White box technique offers the opportunity to be more thorough as it can see inside the code. We have divided this criteria into two sub criteria that 1.Coverage 2. Completeness. Coverage or code coverage analysis is an important measurement of the effectiveness of a testing tool. Code coverage determines the degree of covered paths, flow and statement during a test process. We have included coverage analysis as one of the criteria because it enables the testers to understand the tool in term of its percentage coverage, the type of vulnerabilities that is predicted or identified. There are various types of coverage such as path coverage, statement coverage, function coverage etc. Coverage analysis enables the tester to make decisions relevant to redundant tests, further tests needed, missed paths. Coverage analysis cannot guarantee the absence of faults or errors, but greatly increase the confidence in code exercised through specific type of testing coverage. Completeness means that the entire code or modules have been covered through test cases and is closely relevant to code coverage.

3.2.3 Ease of use

Particular support or facilities provided by testing technique and its tools to ease the process testing. We have to sub criteria integration means the how tightly a testing technique is integrated to the application under test. Platform and tool support is another relevant sub criterion to denote how well particular technique support is available for different platforms and also the degree of interoperability with other testing techniques.

3.2.4 Effectiveness

In our scenario security testing effectiveness means how well the security bugs have been identified by particular testing technique or the number faults identified by the technique. According to Weyuker [9] “effectiveness of a test technique is only possible to measure if you can compare two techniques for the same set (i.e. software), but the result is not general”

3.2.5 Efficiency

Denotes the testing consumed resources [7] such as time, testing resources, the amount of code required.

3.3 Alternatives

The alternative selected for the purpose of this work are Source code analysis, Fault code injection testing, Robustness testing, Stress testing and Penetration testing techniques. The selected alternative includes both black box and white box technique; some of the techniques are hybrid, one being more bended to either black box or white box.

3.4 Assigning priorities

The priorities are assigned to criteria sub criteria and alternative on the basis of the table 1. Priorities are the numbers assigned to criteria, sub criteria associated with an alternative.
Each alternative has been evaluated against criteria and sub criteria and the priorities have been assigned in the form of weights. We have assigned priorities to criteria/sub criteria against each alternative by using the previous studies results and the results obtained from the outcome of the testing tools of selected techniques. The basis of assigning priorities has been discussed in the background study of this paper. Calculated Local and global weights for the main criteria and sub criteria in the through MCDM tool has been shown in Table 2.

Table 2

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Local</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>L=11.4%</td>
<td>G=11.4%</td>
</tr>
<tr>
<td>Skills Required</td>
<td>L=25.0%</td>
<td>G=2.9%</td>
</tr>
<tr>
<td>Testing time</td>
<td>L=75%</td>
<td>G=8.6%</td>
</tr>
<tr>
<td>Thoroughness</td>
<td>L=34.6%</td>
<td>G=34.6%</td>
</tr>
<tr>
<td>Coverage</td>
<td>L=80.0%</td>
<td>G=27.7%</td>
</tr>
<tr>
<td>Completeness</td>
<td>L=20%</td>
<td>G=6.9%</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>L=8.2%</td>
<td>G=8.2%</td>
</tr>
<tr>
<td>Integration</td>
<td>L=75%</td>
<td>G=6.1%</td>
</tr>
<tr>
<td>Tool support</td>
<td>L=25.0%</td>
<td>G=2.0%</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>L=42.1%</td>
<td>G=42.1%</td>
</tr>
<tr>
<td>Efficiency</td>
<td>L=3.8%</td>
<td>G=3.8%</td>
</tr>
</tbody>
</table>

4. Results and Discussion

This section presents the results obtained using MCDM systems, figure 2 shows the alternative ranking against criteria and by observing the chart given in the figure some useful information could be obtained. For security vulnerability effectiveness and thoroughness source code analysis has topped the list, but the relative cost of the source code analysis is also relatively high. Vulnerability scanning the type of testing which has been proved as a less effective but also it consumes less resource. In term if effectiveness and coverage analysis Penetration testing and fault code injection methods are at number two and three respectively but the cost of the fault injection method is relatively high.
Figure 3 shows the alternative comparison in a diagonal graph, the figure highlight the fact that source code analysis being more twisted towards effectiveness and the cost of fault code injection is relatively on higher degree.

**Figure 3**

**Alternatives comparison**

![Figure 3](image)

Figure 4 illustrate the pair wise comparison of our main criteria in percentage, using pair wise comparison the relative importance or preference of one criterion over another has been expressed. Because it is an important in testing to measures the number of defects per test case, therefore effectiveness has comparatively more important than other criteria.

**Figure 4**

**Criteria weight preferences and percentage**

![Figure 4](image)

Figure 5 depicts the graph for effectiveness, static and dynamic source code analysis is thirty degree higher than vulnerability scanning. Penetration testing and fault code injection methods score is also good for effectiveness.

**Figure 5**

**Alternative ranking for Effectiveness**

![Figure 5](image)

5. Conclusion

Software testing plays crucial role to ensure software quality assurance but in this paper we have analyzed five testing techniques to check its application to software security. Each testing technique identifies various types of vulnerabilities in software according to its own way and logic. In software industry testing is a tedious and costly job but an essential step so therefore we choose the criteria to evaluate selected techniques in a way that each optimum test case can be generated according to requirements. The study carried out in this paper use MCDM method and tool to evaluate each alternative and the study results shows the strength and weaknesses of each technique. According to this study the most effective technique to identify and cover more code for vulnerabilities and bugs is both source code analyses. Penetration testing and fault code injection also effective but fault code injection is more time consuming and needs special skills. Vulnerability is the weakest one in term of effectiveness but require less skills, time and resources. The results of this study can be used by testers before developing test cases to optimize testing and reduce the security risk with optimum resources and time at different phases of SDLC. In future we intend to extend this study by comparing testing techniques against specific type of threats such as sql injection and buffer overflow in real world scenario.
References


