Application of T-SEC to Measure the Performance of Static Analyzers and Penetration Testing Approaches

Akwasi Amponsah
Mamp. Tech. Coll of Educ.
Asante Mampong, Ghana

Richard Amankwah
Accra Institute of Technology
Accra, North Ghana

Daniel PaaKorsah
Komenda Coll of Educ
Komenda, Ghana

ABSTRACT

Software vulnerability analysis is very relevant in the process of investigating the existence of bugs (referred to as vulnerabilities) in software application. Recently, several empirical studies such as static code analyzers (SCA) and penetration testing approaches such as web vulnerability scanners (WVS) have been purported to aid the analysis of vulnerabilities in web applications. Although, there are several SCA and penetration testing tools (both open and commercial source) proposed in literature, the performance of these tools varies and make vendors skeptical in relation to the one most suited for detecting a particular type of vulnerability or bug, have a high precision and recall value, a low false positive and a high detection rate.In this study, we applied the standard evaluation criteria (T-SEC), namely precision and recall, Youden index, OWASP web benchmark evaluation (WBE) and the web application security scanner evaluation criteria (WASSEC) to measure the performance of the aforementioned approaches using the Damn Vulnerable Web Application (DVWA) and extracted report from the Juliet Test Suite.

General Terms

Software Privacy • Information security • Software Analysis

Keywords

Open-source scanner, Vulnerability detection, Vulnerability scanner, damn vulnerable web application

1. INTRODUCTION

Security vulnerabilities are uncovered on a regular basis in modern-day systems such as networking, application software and most importantly web applications. Currently,the web application has become the main attacking spot by hackers due to its enormous benefits. The National Vulnerability Database (NVD) [1]which is managed by the National Institute of Standards and Technology (NIST) showsthat vulnerability such as SQL Injection, File Inclusion and Cross-Site Scripting (XSS) continually increased at an astronomical rate yearly in web application [2]. This is because most of the web applications deployed are not totally devoid of vulnerabilities. These vulnerabilities normally cause data breaches and have serious security implications when they are exploited by attackers. To address such a challenge, vulnerability analysis such as manual code inspection, static code analyzers (SCA) and penetration testing approaches have been proposed as a better alternative to improve the quality and efficiency of the manual procedure used in previous studies for bug detection.Unfortunately, the traditional method, which involves manual examination of numerous lines of code is often difficult, unproductive and produce a high rate of false positives. Current techniques which involve the use of automated SCA and WVS also shows varied efficiency and detection capabilities as reported by Antunes and Vieira [3], Makino and Kleve [4], making it difficult to select the appropriate tool for vulnerability detection. Consequently, this study presents an application of the standard evaluation criteria (T-SEC), namely precision and recall, Youden index, OWASP Web Benchmark Evaluation (WBE) and the Web Application Security Scanner Evaluation Criteria (WASSEC) to measure the performance of the static code analyzers and penetration testing approach using the Damn vulnerable web application (DVWA) and vulnerability report from the Juliet Test Suite.The key idea of this study is to apply the standard evaluation criteria (T-SEC):

To evaluate the performance of eight WVS, namely Acunetix, HP WebInspect, IBM AppScan, OWASP ZAP, Skipfish, Arachni, Vega and Iron WASP in identifying security vulnerability in web service environment using the DVWA.

To evaluate the effectiveness of seven widely use SCA, namely Findbug, PMD, LAPSE+, JLint, Bandera, ESC/Java and YASCA using Juliet Test Suite v1.2 test cases.

To suggest possible measures that can be used to improve SCA and WVS

The remaining section of the paper is organized as follows: Section presents the standard evaluation criteria which were used to measure the performance of the tools. Section 3 discusses the methodology and experimental setup for the study. In section 4, we present evaluation of the SCA and the WVS tools. Section 5 present the conclusion and future directions in this domain of study.

2. THE STANDARD EVALUATION CRITERIA (T-SEC)

We evaluated the performance of the tools using the standard evaluation metrics: precision and recall, Youden index, OWASP Web Benchmark Evaluation (WBE) and the Web Application Security Scanner Evaluation Criteria (WASSEC) following a similar procedure in[1].

2.1 Precision and Recall

Precision [5] which is also known as predictive value is the percentage of a correctly detected bug to the number of all detected bugs (i.e. number of bugs detected by the tool that are actually rear bugs). Eq.1 shows how it is calculated. Precision value of 100% represents a high detection accuracy of the exact bug.

$$\text{Precision} = \text{TP}/(\text{TP}+\text{FP}) \qquad (\text{Eq. }1)$$

Recall [6] is the percentage of a correctly detected bug to the number of known bugs (i.e. a number of bugs that were supposed to be detected by the tool but couldn’t detect. Eq.2 shows the formula for recall.

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN}) \qquad (\text{Eq. }2)$$

2.2 Youden Index

The Youden index [7] was proposed by Youden to evaluate the performance of analytical tests (diagnostic tests). The values for the index ranges from -1 to 1. For instance, if a tool is able to detect all bugs without any false positive present it obtains a Youden index of 1 (i.e., no false positive and false negative), this is called a perfect bug detection. However, if the tool could not detect actual bugs but produced false positives then it obtains a Youden index of -1. A Youden index of 0 means the tool recorded the same result for test cases with bugs and test cases without bugs.Eq 6 shows how the Youden index is calculated.

$$J = TP/(TP + FN) + TN/(TN + FP) - 1 \quad (Eq. 3)$$

2.3 Web Application Security Scanner Evaluation Criteria (WASSEC)

The Web Application Security Scanner Evaluation Criteria (WASSEC) [8]is an evaluation guideline to help developers assess the detection capabilityof web application security scanners. The aim is to help stakeholders in this domain to make appropriate decision that meets their specification (in terms of tools capabilities) and for future improvement (by developers) of the tools. The evaluation criteria of WASSEC comprises of the following: protocol support, session management, testing, parsing, authentication, crawling reporting and command control.

2.4 OWASP WBE Result Interpretation Guide

OWASP Benchmark Project proposed a system to measure and evaluate the effectiveness of static analysis tools, which is termed as the WBE result interpretation guide [9]. The WBE result interpretation is a visual representation of static analysis tool performance based on their fallout and recall rate. The guide illustrates how effective a tool had performed in detecting a bug. From Figure. 1, the line extending from the point (100%, 100%) is the “guessing line” where the True Positive rate of bug detection is equivalent to False Positive rate of bug detection. A plot of a tool false positive rate against its true positive rate on the cartesian plane (X-Y) which meet at the top right corner of the guessing line indicates that the tool reported every bug in the test case. On the other hand, if the resulting detection point falls at bottom left corner of the guessing line it means the tool recorded no vulnerability. A top left corner indicates the ideal detection efficiency of the tool.

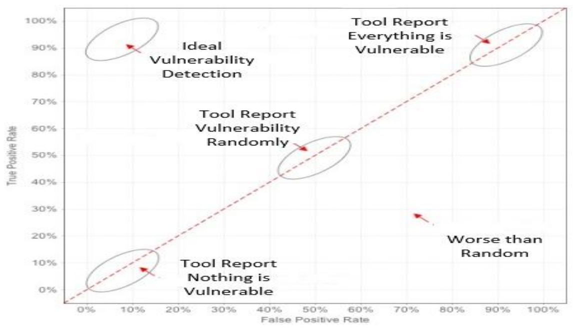
Ideal Vulnerability Detection
Tool Report Everything is Vulnerable
Tool Report Vulnerability Randomly
Worse than Random
Tool Report Nothing is Vulnerable
True Positive Rate
False Positive Rate
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%

Figure 1: WBE results interpretation guide

3. METHODOLOGY AND EXPERIMENTAL STUDY

This section of the paper discusses the methodology and experimentalsetup for the study.

3.1 Experimental setup

The experimental activity is divided into three steps: Pre-Experimental Activities, Experimental Activities and Post Experimental Activities.

Pre-experimental Activities

- Gather information about the SCA and the WVS under studied.
- Gather information about the web service and the dataset under studied.
- Generate the workload based on the information gathered in the previous step

Experimental Activities

- Input the URL of DVWA into the text field of the scanners to scan for vulnerability
- Imported source code for the selected test cases to NetBeans ID Environment together with the installed static analysis tools.
- Also, import test cases source code to the LAPSE+, JLint, Bandera, ESC/Java and YASCA installed on standalone computer.
- Scan for bug in the respective test cases

Post experimental Activities

- Analyze the vulnerability report from the scanners.
- Evaluate the performance of SCA and WVC using The Standard Evaluation Criteria (T-SEC)

3.2 Dataset Description

This paper used Juliet Test Suite v1.2 with a total of 247 java source code files (test cases) which is made up oftwenty Common Weakness Enumeration (CWE) flaw classes. The source code is obtained from the National Institute of Standards and Technology (NIST) Software Assurance Metrics and Tool Evaluation (SAMATE) Project [10]. The Juliet test suite (files) contain test cases which are made up of two descriptions (i.e., Bad and Good) in the file names. When a tool detects a bug in a method with the description “Bad” in its name, then it is classified as a True Positive. In the sense that every "Bad” method contained in the test cases are considered to be known vulnerability. On the other hand, when a tool detects a bug in a method with the description “Good” in its name, it is also classified as a False Positive because it is expected that no actual bug should be found in the method.

3.3 Web Service Tested

In order to test our approach, we identify a vulnerable web application program. We used the open source vulnerable web application commonly referred to as Damn Vulnerable Web Application (DVWA).DVWA[11] has a friendly user interface that allows developer, teachers, and students to explore and analyze web service security.

3.4 Static Code Analyzers Studied

The static analysis technology has grown from early lexical analysis to formal verification method and its detection capability has now improved a lot [12]. Static code analyzers investigated in this study include: Findbug [13], PMD [14] Yasca [15] , JLint [16], Bandera [17] and Extended Static Checking system for Java ESC/Java [18]

3.5 Penetration Testing Tools Studied

A Web scanner examines an application by going through its web pages and performs penetration testing. Most web application scanners consist of three main components: a crawling component, an attacker component, and an analysis component. The web vulnerability scanners investigated in this study include: ZAP Skipfish [19]Arachni[20]IronWASP (Iron Web application Advanced Security testing Platform) Vega [21]Acunetix[22], WebInspect[23]AppScan.

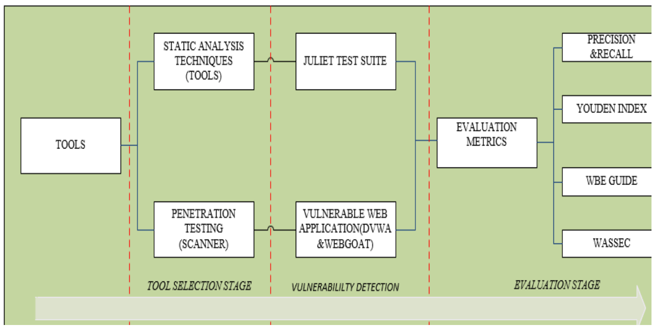

Figure 2: Framework for the tool detection.

4. RESULTS AND DISCUSSION

This section presents evaluation of the SCA and the WVS tools.

4.1 Precision And Recall Analysis

In this study, both precision and recall metrics are measured in the range of 0-100%. For instance, an effective tool whose detection has no false negative and false positive would have a value of 100% or 1 for precision and recall. The result shows that all the WVS obtained a recall value of 1 in DVWA which is an indication of the tools ability to detect rear vulnerabilities in web application under study. Additionally, there is variation in terms of the precision values of the scanners. For example, Skipfish obtained a precision value of 0.75 in DVWA, whereas Acunetix obtained a precision value of 0.68 in DVWA and 0.64. Zap, Arachni and Vega obtained a precision value of 0.56 respectively. This is because the aforementioned tools detected vulnerabilities that are actually not correctly classified as rear vulnerability (false positive). We observed from the SCA results that most of the tools show variations in the precision and recall figures. For example, JLint shows a high precision for CWE-568 and a low recall, which indicates that the number of bugs it's detected were correctly found. Findbug and PMD shows a high precision and a high recall figure for CWE-328, CWE-382, CWE-476, which means, some of the bugs detected are actually not correctly classified as bugs (False Positive). We also observed that tools such as Bandera and ESC/Java could not detect bugs in some of the test cases, this could be attributed to the way the tools are design to detect specific types of bugs. Most of the tools could not detect CWE-198 apart from Findbug and PMD and we believe it is because such bug is difficult to detect in the Juliet test cases.

4.2 OWASP WBE Analysis

As explained in section 2, a tool effectiveness is determined by its position on the WBE interpretation guide.Zap position is at the top right corner of the guessing line, which means the tool detect and report" everything is vulnerable" (i.e., the tool performance in terms of false positive and true positive is high). Iron Wasp fell in the category of "nothing is vulnerable", which means the tool false positive and true positive is low.

The remaining scanner such as, Acunetix, Arachni, WebInspect fell into the "random detection of vulnerability" category except AppScan which was close to "everything is vulnerable category. Conversely, the result of the SCA shows that Findbug recorded a True Positive Rate of 136% and False Positive Rate of 44% which makes it an ideal bug detection tool. PMD had a True Positive Rate of 129% and False Positive Rate of 51% which also makes it an ideal bug detection tool. LAPSE+ recorded a True Positive Rate of 20%% and False Positive Rate of 6% which means the tool detection rate is low. ESC/Java and Bandera falls into this same category. YASCA is an ideal bug detection tool.

4.3 Youden Index Analysis

IronWASP obtained the highest Youden Index of 0.83 which implies that the tool was able to detect the vulnerability it was intended for with no or little False Positive. Followed by Skipfish, AppScan, WebInspect and Acunetix with 0.45, 0.31, 0.23 and 0.21 respectively.Zap recorded the lowest Youden index of 0.08. LAPSE+ obtained the highest Youden Index of 0.9 which implies that the tool was able to detect all bugs it was intended for with no or little False Positive. Followed by Bandera and ESC/Java with index of 0.8. Findbug recorded the lowest Youden Index of 0.1.

4.4 The Web Application Security Scanner Evaluation Criteria (WASSEC) Analysis

We evaluated the performance of the scanners based on the WASSEC criteria (i.e., protocol support, session management, testing, parsing, authentication, crawling). The results show that in the protocol support, Acunetix scanners are very good followed by AppScan and Skipfish. But in the area of session management, the differences in the performance of the scanners are not much. Additionally, although there are differences in the performance of the scanners as far as the criteria is concern, there are also similarities in the area of crawling, authentication and testing. Generally, we can say thatAcunetix and AppScan are very effective considering their average evaluation factors of 0.81 and 0.65 respectively. However, scanners such as Skipfish and Zap are alternative for stakeholder with an average index of 0.43 and 0.40 respectively which is better than WebInspect.

5. CONCLUSION AND FUTURE WORK

In this study, we applied the standard evaluation criteria namely precision and recall, Youden index, OWASP Web Benchmark Evaluation (WBE) and the Web Application Security Scanner Evaluation Criteria (WASSEC) to measure the performance of Static Code Analyzers (SCA) and penetration testing approachesusing the Damn Vulnerable Web Application (DVWA) and extracted report from the Juliet Test Suite. The results show that, the commercial WVS are effective in detecting security vulnerabilities in web application. Again, the experimental outcome of the SCA shows a lot of diversities, LAPSE+ static analysis tool has the highest Youden Index of 0.9, making it the best effective tool to detect all bugs with relatively low false positive.However, tools such as Findbug, PMD, YASCA, and JLint also have a high rate of precision, but their level of false positive rate is high. Theremaining section presents recommendations for possible replication of the study.

5.1 Future Direction

We recommend and make the following suggestions for future research direction and possible replication of this study:An improvement in the internal structure of both SCA and WVS to enhance accuracy, coverage and a reduction in false positive rate. We additionally suggest a further upgrade and maintenance of SCA and WVS. Again, we recommend an approach that can integrate SCA and WVS to compliment the weakness and strength of each other in detecting vulnerabilities or bugs. We observed that the higher the lines of code the greater number of bugs detected. We therefore recommend better quality coding style by programmers. Finally, IDE enhance bug detection in program source code, hence developers should ensure the integration of static analysis tools into existing IDE's.

6. REFERENCES

[1] N. Antunes and M. Vieira, "Benchmarking vulnerability detection tools for web services," in 2010 IEEE International Conference on Web Services, 2010, pp. 203-210.

[2] C. L. Blackmon, D. F. Sang, and C.-S. Peng, "Performance Evaluation of Automated Static Analysis Tools," GSTF Journal on Computing (JoC), vol. 2, 2014.

[3] Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan, "A cost-effective approach to evaluating security vulnerability scanner," in 2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2013, pp. 1-3.

[4] 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, pp. 399-402.

[5] E. Goubault and S. Putot, "Static analysis of finite precision computations," in International Workshop on Verification, Model Checking, and Abstract Interpretation, 2011, pp. 232-247.

[6] N. Ayewah, W. Pugh, D. Hovemeyer, J. D. Morgenthaler, and J. Penix, "Using static analysis to find bugs," IEEE software, vol. 25, pp. 22-29, 2008.

[7] W. J. Youden, "Index for rating diagnostic tests," Cancer, vol. 3, pp. 32-35, 1950.

[8] W. A. S. Consortium, "Web application security scanner evaluation criteria WASSEC," ed, 2016.

[9] "OWASPBenchmarkProject. https://www.owasp.org/index.php/Benchmark (visited 2016-06-2).", ed, 2016.

[10] P. E. Black, "Samate and evaluating static analysis tools," Ada User Journal, vol. 28, pp. 184-188, 2007.

[11] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in Proceedings of the 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015, 2015, pp. 399-402.

[12] R. Amankwah, P. K. Kudjo, B. K. Agyman, K. Mensah, B. Brew, and S. Y. Antwi, "A Theoretical Framework for Software Vulnerability Detection based on Cascaded Refinement Network," International Journal of Computer Applications, vol. 975, p. 8887.

[13] N. Ayewah, W. Pugh, J. D. Morgenthaler, J. Penix, and Y. Zhou, "Evaluating static analysis defect warnings on production software," in Proceedings of the 7th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, 2007, pp. 1-8.

[14] N. Rutar, C. B. Almazan, and J. S. Foster, "A comparison of bug finding tools for Java," in null, 2004, pp. 245-256.

[15] H. H. AlBreiki and Q. H. Mahmoud, "Evaluation of static analysis tools for software security," in Proceedings of 2014 10th International Conference on Innovations in Information Technology (INNOVATIONS), , 2014, pp. 93-98.

[16] C. Artho, "Finding faults in multi-threaded programs," ed, 2001.

[17] J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Pasareanu, and H. Zheng, "Bandera: Extracting finite-state models from Java source code," in Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium, 2000, pp. 439-448.

[18] C. Flanagan, K. R. M. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata, "Extended static checking for Java," ACM Sigplan Notices, vol. 37, pp. 234-245, 2002.

[19] T. P. Chiem, "A study of penetration testing tools and approaches," Auckland University of Technology, 2014.

[20] C. Mainka, J. Somorovsky, and J. Schwenk, "Penetration testing tool for web services security," in 2012 IEEE Eighth World Congress on Services, 2012, pp. 163-170.

[21] N. Antunes and M. Vieira, "Penetration testing for web services," Computer, vol. 47, pp. 30-36, 2013.

[22] H. M. Z. Al Shebli and B. D. Beheshti, "A study on penetration testing process and tools," in 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2018, pp. 1-7.

[23] R. Amankwah, J. Chen, P. K. Kudjo, and D. Towey, "An empirical comparison of commercial and open-source web vulnerability scanners," Software: Practice and Experience, vol. 50, pp. 1842-1857, 2020.