Benchmark Requirements for Assessing Software-based Security Vulnerability Testing Tools

Reza M. Parizi
Department of Software Engineering and Game Development
Kennesaw State University
Marietta, GA, USA
rparizi1@kennesaw.edu

Kai Qian
Department of Computer Science
Kennesaw State University
Marietta, GA, USA
kqian@kennesaw.edu

Fan Wu
Department of Computer Science
Tuskegee University
Tuskegee, AL, USA
fwu@tuskegee.edu

Lixin Tao
Department of Computer Science
Pace University
Pleasantville, NY, USA
ltao@pace.edu

***Abstract*—Consistent growth in the software sector of the world economies has attracted both targeted and mass-scale attacks by cybercriminals. Producing reliable and secure software is difficult because of its growing complexity and the increasing number of sophisticated attacks. Developers can't afford to believe that their security measures during development are perfect and impenetrable. In fact, many new software security vulnerabilities are discovered on a daily basis. Therefore, it is vital to identify and resolve those security vulnerabilities as early as possible. Security Vulnerability Testing (SVT), as an active defense, is the key to the agile detection and prevention of known and unknown security vulnerabilities. However, many software engineers lack the awareness of the importance of security vulnerability and the necessary knowledge and skills at the testing and operational stages. As a first step towards filling this gap, this paper advocates for building skills in selecting proper benchmarks for the assessment of SVT tools to enable distinguishing valuable security tools from trivial ones. Thus, we provide a set of requirements in fulfillment of this need, primarily addressing newcomers and researcher to the discipline.**

***Keywords*—software security, security testing tools, vulnerability, security vulnerability testing, benchmark**

I. Introduction

The spread of computing has expanded society's dependence on secure software applications to ensure national security and safety. Producing reliable and secure software is difficult because of its growing complexity and the increasing number of sophisticated attacks, which pose serious challenges to individuals, companies, and nations. It is very likely that security vulnerabilities remain in software applications as the development progresses and henceforth. Developers can't afford to believe that their security requirements during development are perfect and impenetrable, no matter how thorough their precautions might be [1]. Having a perimeter and defending it are not enough because the perimeter fades away due to continuous integration and delivery, advocated by the DevOps [2]. That is, passive defense approaches alone are insufficient to counter dynamic-nature vulnerabilities which may be used by attackers to get past on-development security measures and do worse damage to the cyber-infrastructure of our modern society and economy.

Among the various aspects of cybersecurity, software security plays a critical role. Software security testing is the process of assessing an application to discover risks and vulnerabilities of the application and its data. In a software-run world, what is wanted are more capable skilled professionals whom will be able to put in motion active defense in real-world software and web applications. However, many software engineers lack the awareness of the importance of security vulnerability and the necessary secure knowledge and skills at the testing and operational stages. This has resulted in a severe shortage of security-vulnerability testers and analysts. Education for SVT of modern software applications is still a very undervalued topic and has not been well represented in most schools' computing curriculum. We believe this problem will be not addressed unless computer and software educators start to include this notion as a first-class problem in their agenda for future generation of experts.

As a first step towards filling this gap, in this paper, we propose and discuss the rationale behind an initial set of benchmark requirements that we believe could be useful in building a must-to-possess fundamental skill for the evaluation of off-the-shelf and newly proposed security tools (either Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), or Interactive Application Security Testing (IAST) tools for finding vulnerabilities). Without the ability to meaningfully measure and scale tools' performance, it is impossible to understand their pitfalls and advantages, and consequently the needs for the future advancement. In addition, we illustrate the use of our proposed requirements by using them to guide the selection of possible benchmark candidates amongst existing open source projects.

II. Proposed Benchmark Requirements

Automated SVT tools and scanners (SAST, DAST, and IAST) can be evaluated in terms of various quality and performance factors using synthetic benchmarks or benchmarks based on real-world software. Fig. 1 shows the holistic benchmarking process devised in our context.

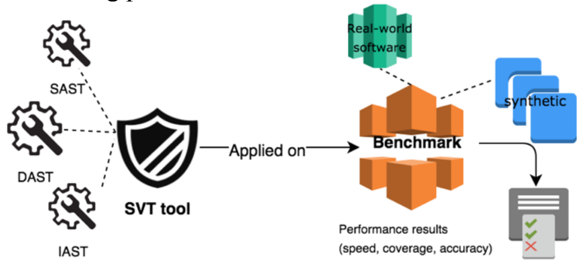

Fig. 1. Holistic view of benchmarking process.

Performing benchmark process is a non-trivial task, particularly with real-world software. The biggest problem of using real-world software for benchmarking is the unavailability code and its documentation to public domain, and most importantly it involves legal issues. As a result, open-source benchmarks have been commonly used instead to compare the outputs of different SVT tools. In fact, the software engineering research community has long relied on Free and Open Source Software (FOSS) projects as common benchmarks with which to conduct and replicate a wide variety of studies. However, selecting a right FOSS benchmark that unbiasedly captures a tool's performance is challenging due to lack of vulnerability realism, uncertain ground truth, and large amount of irrelevant findings (i.e., background noise [3]) by the benchmark tests. In this spirit, our aim is to come up with a set of requirements that capture the benefits of both synthetic benchmarks and real-world software by eliminating the above-mentioned issues. Table I presents our proposed benchmark requirements, as a start point for what constitutes an 'ideal' benchmark for SVT tools' assessment and empirical research.

TABLE I. BENCHMARK REQUIREMENTS

Requirement	Rationale
R1: Easy Access from a Version Control Repository	The benchmark's source code and release history should be easily accessible from a public version control system (VCS), such as GitHub.
R2: Support for Automated Scoring and Interpretation	The benchmark should provide a tool to support for the calculation of benchmark score using metrics* (TP, FN, TN, FP), and their results interpretation guide.
R3: Support for Dependency Management	The benchmark should provide support by automatically downloading and installing locally all external software artifacts (e.g., components, libraries) required to create a build of the benchmark project and its tests.
R4: Contain Diversified Benchmark Test Suits with CWE Numberings	The benchmark should provide diversified large number of test cases aligned with Common Weakness Enumeration (CWE) to promote generally accepted security practices and to reduce confusion.
R5: Support for Automated Benchmark Test's Execution	The benchmark should provide fully runnable and exploitable tests.
R6: Availability of Vulnerable and Fixed Versions of each Benchmark Test	The benchmark should provide both fixed and vulnerable versions of each test to enable evaluating the "actual" tool performance considering only the alerts relevant to the analyzed vulnerability.
R7: Community Usage & Interest	The benchmark should be easy to use and attract the interest of its target education and research community.
R8: Alternate Versions	The benchmark should provide alternate implementations in terms of programming languages for broader accessibility.

* https://www.owasp.org/index.php/Benchmark

From a practical perspective, meeting these requirements is important, as this would allow software engineers to evaluate and experiment with industry-strength DevOps practices and technologies when conducting SVT tools' assessment studies.

III. Analysis and Result of a Benchmark Selection

To illustrate the benefits of our proposed set of requirements, we have used them to guide the selection of possible benchmark candidates from FOSS projects. We have exhaustively searched the web for existing open source applications that could be good candidates for a benchmark according to our proposed requirements. As expected, there are very a few publicly-available applications that could satisfy our needs, thus two existing projects, WAVSEP [4] and OWASP benchmark [5] were identified.

In order to assess whether each benchmark candidate would satisfy each of our proposed requirements, we have preliminarily examined each candidate based on their provided documentation and on the information available in their respective software repositories. Our assessment results are presented in Fig. 2.

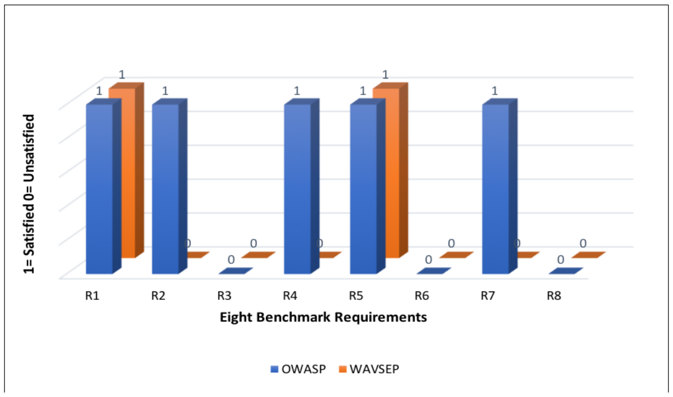

Fig. 2. Benchmark candidate assessment results.

As it can be seen from the figure, no single benchmark candidate fully satisfies all our proposed requirements. Our early results indicate that, although none of the projects analyzed is advanced enough to be used as a community-wide benchmark, OWASP may already be useful to fulfill the needs and promote the reproducibility of specific evaluation studies.

IV. Conclusion

This paper presented an initial set of requirements for a candidate benchmark project to be used in assessment of SVT and detection tools. The proposed requirements were discussed and illustrated in the context of a selection analysis. By using these requirements better benchmarks will be selected, helping individuals and companies select the most appropriate SVT tool for their projects, and bringing awareness to tool developers to improve SVT tools for sharper results.

Acknowledgment

The work is partially supported by the National Science Foundation under award: NSF proposal 1723586, 1723578, and 1438858.

References

[1] R. M. Lee and R. Lee, "The Who, What, Where, When, Why and How of Effective Threat Hunting", *SANS Institute*, 2016.
[2] I. W. L. Bass and L. Zhu, "DevOps: A Software Architect's Perspective", *Addison-Wesley Professional*, 2015.
[3] I. Pashchenko et al., "Delta-Bench: Differential Benchmark for Static Analysis Security Testing Tools," in *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 163-168, 2017.
[4] WAVSEP, https://github.com/sectooladdict/wavsep.
[5] OWASP, https://github.com/OWASP/Benchmark