Using Software Engineering Metrics to Evaluate the Quality of Static Code Analysis Tools

ABSTRACT

- This paper presents a framework for evaluating the quality of static code analysis (SCA) tools in the context of different software engineering metrics. The framework supports up to 38 software engineering metrics. We applied the framework against both open-source and commercially available SCA tools.
- The results of our experiments show that software engineering metrics, such as cyclomatic complexity, fan-out, knots, and essential complexity can impact the ability of a static code analysis tool to identify potential vulnerabilities in source code.

EXISTING SYSTEM

- potential vulnerabilities that could eventually compromise software security. SCA is traditionally carried out by running many different SCA tools against the code base. For example, developers may choose from a mix of open-source and commercial SCA tools because different SCA tools may have some overlap and produce different results.
- Likewise, some of the SCA tools may focus on identifying a specific vulnerability from the list of Common Weakness Enumerations (CWEs) Regardless ofthe SCA tool used to locate potential vulnerabilities, the end goal of this exercise is to improve software quality.

DISADVANTAGES

- Software developers and testers have many SCA tools to choose from, a challenge they face is identifying what tool to use against their code base. As mentioned above, different SCA tools have their strengths, weaknesses, and performance characteristics, which we call its quality, in terms of being able to correctly identify potential vulnerabilities.
- The problem is exacerbated when multiple SCA tools claim to check the same vulnerabilities, but generate different results.

PROPOSED SYSTEM

- We applied the framework against both open-source and commercially available SCA tools.
- The results of our experiments show that software engineering metrics, such as cyclomatic complexity, fan-out, knots, and essential complexity can impact the ability of a static code analysis tool to identify potential vulnerabilities in source code.

ADVANTAGES

- software developers and testers have many SCA tools to choose from, a challenge they face is identifying what tool to use against their code base. As mentioned above, different SCA tools have their strengths, weaknesses, and performance characteristics, which we call its *quality, in terms* of being able to correctly identify potential vulnerabilities.
- The problem is exacerbated when multiple SCA tools claim to check the same vulnerabilities, but generate different results. In this scenario, at least one of the SCA tools is generating both false positives (FPs), which are locations in source code that are incorrectly labeled to have a flaw, and false negatives

HARDWARE REQUIREMENTS

- Processor :Intel Pentium IV 1GHz
- RAM :256MB (Min)
- Hard Drive :5GB free space
- Monitor :1024 * 768, High Color inch
- Mouse :Scroll Mouse(Logitech)
- Keyboard :104 keys

SOFTWARE REQUIREMENTS

- OS : Windows XP/7/8
- Front End : Visual Studio 2010/ netbeans 7.1
- Back End : SQL Server 2005/ heidisql 3.2
- Browser : Any Web Browser

CONCLUSION

- Based on our current results the software engineering metrics can be listed in decreasing order based on how they affect the TP rate for each SCA tool. From the experimental results, we observed that most of the five SCA tools achieve lower value of TP rate when the source code has a high degree of complexity.
- This is because the source code will be more complicated, error prone, and difficult-to-understand. The second type of software engineering metric that leads the SCA tools to find low number of flaws are object-oriented metrics such as Count Output. For example, when the source code an SCA tool's ability to find more flaws. Last, volume metrics have the least negative impact on the number of flaws identified by an SCA tool.

REFERENCES

[1] Owasp. ”Static Code Analysis.” Internet: https://www.owasp.org/index.php/Static Code Analysis, Sep. 29,2017 [Oct. 15,2017].

[2] Wikipedia. ”List of tools for static code analysisWikipedia, The Free Encyclopedia.” Internet:

https://en.wikipedia.org/w/index.php?title=List of tools for static code analysis&oldid=739038439, Sep. 12,2016 [Sep. 13,2017].

[3] U.S. Department of Homeland Security. ”The Common Weakness Enumeration (CWE) Initiative, MITRE Corporation.” Internet: http://cwe.mitre.org/index.html, Jan 16, 2018 [Jan 20, 2018].

[4] A. R. Knudsen. ”Evaluating the ability of static code analysis tools to detect injection vulnerabilities.” Bachelors thesis, Ume University, Sweden, 2016.

[5] T. Boland, and P. E. Black. ”Juliet 1.1 C/C++ and Java Test Suite.” IEEE

CS, vol. 45, pp: 88-90, 10 - Oct.2012.