Southern New Hampshire University

CS-330: Assignment 2-2: Module 2 Journal

Prepared on: March 18, 2024

Prepared for: Prof. Angelo Luo

Prepared by: Alexander Ahmann

Contents

1 Introduction

Software testing is an important stage of the *Software Development Life Cycle*.[1] I have given a brief overview of the SDLC in a previous assignment (Ahmann, 2024), elaborated specifically on the testing stage,[2] discussed the importance of software testing,[3] and how, at times, the process is not a simple procedure, but rather nuanced and "branched out."[4]

Software testing is done at two levels: *static analysis* and *dynamic analysis*, which tests software at rest and while being executed, respectively. In this paper, I will discuss the static and dynamic approaches, as well as other methods and related notions.

[1] Which I shall shorten down to "SDLC."

[2] Ibid.

[3] Ibid.

[4] Ibid.

2 Levels of Software Testing

2.1 Static Testing

Static testing can be defined as a set of techniques to test software without executing it.[5] This approach is more applicable than dynamic testing because it does not just apply to the software *itself*. Static analysis can be applied to scripts and compiled binaries, and it can also be applied to the software's design specifications, user guides, web pages, design models, et cetera Hambling et al. (2019, p. 75). In fact, static testing is general to the point that it can even be applied to "epics" and fictional stories.[6]

The thing being tested here are abstractions related too the software and its actual implementation in source code, binaries, or scripts. These things are read, without execution, and double checked for correctness, reliability of output, and good programming practices. Static analysis can be used to work out the causes of failures[7] and find bad coding practices.[8] It has the advantage of looking at "the whole picture" because every possible path of execution is laid out to the tester and automation tools.

2.2 Dynamic Testing

Dynamic testing can be defined as "a software testing method used to test the behaviour of software code at runtime."[9] Rather than trying to get "the whole picture" of the software being tested, it instead analyses the parts that get executed at runtime and observes what paths the software takes and how it interacts in a hostile operating environment. This reduces the amount of information to be processed and takes into consideration possibilities and exceptions that may not be known to testers or automated static testing tools.

[5]Paraphrased from Hambling et al. (2019, p. 73).

[6]Ibid.

[7]Hambling et al. (2019, p. 75).

[8]In software development, *linting* is "the automated checking of your source code for programmatic and stylistic errors." They are good stuff for working out where a software crash may occur, and exactly what may have caused it. They are also good for spotting bad coding practices. See the following article (Retrieved on Mar. 17, 2023) to learn more: https://owasp.org/www-project-devsecops-guideline/latest/01b-Linting-Code

[9]Paraphrased from Hamilton (2024).

Static Testing	Dynamic Testing
Code is executed	Code is not executed
Tests all code	Tests code only if it is executed
Finds the cause of failure	Demonstrates failure but not its cause

Table 1: Differences between static and dynamic testing. After Hambling et al. (2019, pp. 76–77).

3 Discussion

3.1 Differences in approach

The textbook discusses the differences between static and dynamic analysis. I do not want to rehash what they said, so I list some of them in table 1. Instead, I want to discuss what *I think* are significant differences based on what was described. The main difference that I noticed is an advantage that dynamic testing might have over static testing. A security professional called Greg Hoglund gave a presentation on *Active Reversing* ("killab66661", 2010) where he contends that software is complex and that dynamic analysis methods will filter out for the relevant bits of code.[10]

Of course, Hoglund is speaking from a *reverse engineering* perspective as opposed to a *software testing* perspective. He is more interested in finding out how software that has been released works, whereas software testers want their code to be more reliable. The reverse enngineer is "working backwards," whereas the software tester wants to proceed forward.

Nonetheless, the techniques employed by reverse engineers can be useful to software testing. *Fuzzing* has become an important part of developing software that is not only reliable, but also secure.[11] Software fuzzing injects random characters and inputs into software being executed, allowing software developers to observe how it would behave in hostile environments. In my opinion, this is one advantage to static testing, as a prescribed methodology or a automated static analysis tool may not be able to work out all the possible results for each given result.

[10]This is a gross oversimplification and I recommend that the reader watch the presentation for themselves.

[11]See Godefroid (2020) to learn more.

3.2 Multipronged testing approaches

Gray-box analysis can be defined as techniques to evaluate software at runtime while knowing its design and implementation.[12] This is a blend of both static testing and dynamic testing approaches: where the testers will execute each and every function in the software and observe how it would behave in a hostile operating environment.

This gives the software developers the "best of both worlds" where they can overcome the limitations of dynamic analysis and static analysis. They can not only determine that a problem occured, but what caused the problem. They can now dynamically execute all possible code paths and have a more complete picture of how the software behaves in hostile operating environments and not be overloaded with too much information.

4 Conclusions

- Static analysis allows for software developers and quality assurance testers to get a "full scale picture" of how the software product is developed and shows all possible code paths.[13]
- Dynamic analysis allows for the software to execute and shows specific code paths that are executed depending on its operating environment and the end user's input. It does not give a whole picture, but rather just specific relevant information regarding the software's execution.[14]
- The major differences between static analysis and dynamic analysis are that static analysis does not run the software whereas dynamic analysis does run the software, and that static analysis gives "the whole picture" whereas dynamic analysis just gives relevant bits of information.[15]
- It is important to use both static and dynamic testing methods because each of them have strengths and weaknesses in their approaches, and "the best of both" of them can be taken into a *grey-box analysis*.[16]

[12]Paraphrased from Imperva (n.d.).

[13]This answers the question: "what is static testing?" in the homework guidelines (CS-320, n.d.).

[14]This answers the question "what is dynamic testing?" in the homework guidelines (CS-320, n.d.).

[15]This answers the question "what are the differences between static and dynamic testing?" in the homework guidelines (CS-320, n.d.).

[16]This answers the question "why is it important to use both static and dynamic testing?" in the homework guidelines (CS-320, n.d.).

References

Ahmann, A. (2024). *CS-330: Assignment 1-3: Module 1 Journal.*

Godefroid, P. (2020). *A brief introduction to fuzzing and why it's an important tool for developers.* Microsoft Research Blog. Retrieved on Mar. 18, 2024 from: https://www.microsoft.com/en-us/research/blog/a-brief-introduction-to-fuzzing-and-why-its-an-important-tool-for-developers/

CS-320 (n.d.). *Module Two Journal Guidelines and Rubric.* Southern New Hampshire University.

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *Software testing : An istqb-bcs certified tester foundation guide — 4th edition.* BCS Learning & Development Limited.

Hamilton, T. (Mar. 9, 2024). *What is Dynamic Testing? Types, Techniques & Example.* guru99. Retrieved on Mar. 17, 2024 from: https://www.guru99.com/dynamic-testing.html

Imperva (n.d.). *What is Gray Box Testing?* Retrieved on Mar. 18, 2024 from: https://www.imperva.com/learn/application-security/gray-box-testing/

"killab66661" (2010). *2007 BlackHat Vegas V30 Hoglund Active Reversing 00.* YouTube Video. Retrieved on Mar. 18, 2024 from: https://www.youtube.com/watch?v=PAFM9jSIXEQ