University of Basel

Optimizing Symbolic Execution Through Taint Analysis and Path Prioritization

Bachelor thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Databases and Information Systems (DBIS) Group
https://dbis.dmi.unibas.ch/

Examiner: Dr. Marco Vogt
Supervisor: Prof. Dr. Christopher Scherb

Ruben Hutter
ruben.hutter@unibas.ch
2020-065-934

02.07.2025

Acknowledgments

I would like to thank my supervisor Prof. Dr. Christopher Scherb for the support in undertaking this thesis by helping me understand relevant topics and providing valuable insight for carrying out this thesis. I would also like to thank Dr. I would not be able to realise this thesis without him. Marco Vogt for giving me the opportunity to write this thesis with an external supervisor and for the advice in writing scientific papers. I have learned a lot while writing this thesis and have gained further insight into the world of cybersecurity, which I am fascinated by. I would also like to thank all of my friends that were supporting me through my bachelors degree, which lead me to writing this thesis. I would especially like to shout out Giovanni, thank you for biting through all project with me.

Abstract

Symbolic execution is a powerful program analysis technique widely used for vulnerability discovery and test case generation. However, its practical application is often hampered by scalability issues, primarily due to the "path explosion problem" where the number of possible execution paths grows exponentially with program complexity. This thesis addresses this fundamental challenge by proposing an optimized approach to symbolic execution that integrates taint analysis and path prioritization.

The core idea is to move away from uniform exploration of all program paths towards a more targeted analysis, focusing on paths that are most relevant to security-critical aspects. Specifically, this work prioritizes paths originating from memory allocations and user inputs, as these are common sources of vulnerabilities. By leveraging taint analysis, the system identifies and tracks data originating from these critical sources, allowing the symbolic execution engine to concentrate its efforts on paths influenced by such "tainted" data, while ignoring paths with no dependency on external inputs.

The effectiveness of this optimization is evaluated through a comparative analysis, examining criteria such as runtime efficiency, the number and relevance of discovered program paths, and the identification of potential vulnerabilities. This approach aims to significantly improve the efficiency and applicability of symbolic execution for large and complex software systems.

Table of Contents

1 Introduction

Symbolic execution is a powerful program analysis technique used for vulnerability discovery and test case generation. It explores program execution paths by using symbolic variables instead of concrete inputs. This allows for a more comprehensive analysis compared to traditional testing methods.

1.1 Motivation: Addressing the Path Explosion Problem

A key challenge in symbolic execution is the "path explosion problem". As program complexity increases, the number of possible execution paths grows exponentially, making analysis computationally expensive and time-consuming. This limits the scalability of symbolic execution to larger, real-world applications.

1.2 Problem Statement

This thesis addresses the inefficiency of symbolic execution when applied to large software. Existing approaches often explore all paths indiscriminately, wasting resources on non-critical code sections. A more targeted approach is needed to focus the analysis on potentially vulnerable areas.

1.3 Goals and Contributions

This work aims to improve the efficiency of symbolic execution by integrating taint analysis and path prioritization techniques. The main contributions are:

- **Improved Efficiency:** Reducing the impact of the path explosion problem by prioritizing relevant paths.
- **Taint Analysis Integration:** Using taint analysis to track user-controlled inputs and memory allocations, focusing on security-critical data.
- **Path Prioritization:** Guiding the symbolic execution engine (e.g., Angr) towards paths influenced by tainted data, and ignoring irrelevant paths.

- **Evaluation of Effectiveness:** Assessing the performance and vulnerability detection capabilities of the optimized approach.

This thesis presents a novel approach to optimizing symbolic execution, providing a more efficient method for security-focused program analysis.

1.4 Structure of the Thesis

The remainder of this thesis is structured as follows:

- **Chapter 2: Background** - Foundational concepts, including symbolic execution, taint analysis, and the Angr framework.
- **Chapter 3: Conceptual Implementation** - Theoretical framework and prioritization strategies.
- **Chapter 4: Practical Implementation** - Technical details, tools, and libraries used.
- **Chapter 5: Evaluation** - Experimental setup and performance results.
- **Chapter 6: Conclusion** - Key findings and their implications.
- **Chapter 7: Future Work** - Potential extensions and future research directions.
- **Chapter 8: Related Work** - Existing research in symbolic execution and related areas.
- **Chapter 9: Usage of AI** - Documentation of AI-supported technology used in the thesis.

2 Background

Base for the thesis, definitions, concepts, and related work (similar approaches or works that give a background for the thesis).

2.1 Symbolic Execution

2.1.1 Core Concepts

2.1.2 Path Explosion Problem

2.1.3 Existing Techniques for Path Pruning

2.2 Taint Analysis

2.2.1 Definition and Use Cases

2.3 Program Representation

3 Improving Symbolic Execution ...

This is a short conclusion on the thesis template documentation. If you have any comments or suggestions for improving the template, if you find any bugs or problems, please contact me.

How does it work conceptually? (not implementation) which path i choose and why. limiter -¿ je weiter unter desto schwieriger dass ich eine vulnerability triggere Pseudo code of the algorithm

4 Practical Implementation

This is a short conclusion on the thesis template documentation. If you have any comments or suggestions for improving the template, if you find any bugs or problems, please contact me.

How does the script work? Implementation details, how to use it, how to run it, how to set up the environment. Strategies that I use, LoopSeer...

5 Evaluation

Compare my work to default angr strategy and other tools. (Maybe I will implement a benchmark program to compare the performance of my tool with angr and other tools.)

6 Related Work

Evtl. already mentioned in background chapter

Here other approaches to symbolic execution, like the Paper: ”MACKE: Compositional Analysis of Low-Level Vulnerabilities with Symbolic Execution”

7 Conclusion

Rückblick auf die Arbeit, was wurde erreicht (evtl. merge with Future Work)

8 Future Work

Some ideas for future work could be: - Change meta file to actual header file - Make it work also for ARM and X86 (checking stack and heap arguments) - Check that it works also for libraries (not only for main function) - Let the script analyze a complex program (multiple files) and get an output over all (now it only works for one file at a time)

9 Usage of AI

Used AI tools, like ChatGPT, to help with the writing of this thesis. I used it to generate ideas, to write parts of the text, to improve the text, and to check the grammar and spelling. I also used it to generate code snippets and to explain code snippets.

Bibliography

A
Appendix

University of Basel

Faculty of Science

Declaration on Scientific Integrity

(including a Declaration on Plagiarism and Fraud)
Translation from German original

Title of Thesis:

Name Assessor: ______________________________

Name Student: ______________________________

Matriculation No.: ______________________________

I attest with my signature that I have written this work independently and without outside help. I also attest that the information concerning the sources used in this work is true and complete in every respect. All sources that have been quoted or paraphrased have been marked accordingly.

Additionally, I affirm that any text passages written with the help of AI-supported technology are marked as such, including a reference to the AI-supported program used.
This paper may be checked for plagiarism and use of AI-supported technology using the appropriate software. I understand that unethical conduct may lead to a grade of 1 or “fail” or expulsion from the study program.

Place, Date: ____________________ Student: ____________________

Will this work, or parts of it, be published?

No

Yes. With my signature I confirm that I agree to a publication of the work (print/digital) in the library, on the research database of the University of Basel and/or on the document server of the department. Likewise, I agree to the bibliographic reference in the catalog SLSP (Swiss Library Service Platform). (cross out as applicable)

Publication as of: ______________________________

Place, Date: ____________________ Student: ____________________

Place, Date: ____________________ Assessor: ____________________

Please enclose a completed and signed copy of this declaration in your Bachelor’s or Master’s thesis.

September 2023