



Björn Kirschner



Technische Universität München

GETINTUM - DOOR ACCESS SYSTEM

Idea Report for Android Practical Course (F13)

May 18, 2015

Contents

1	Introduction	2
2	Selective Symbolic Execution	2
3	bla3	3
4	Design Alternatives	3
5	Project Team	3
6	Action Planning and Scheduling	3
7	Action Planning and Scheduling	3
8	Action Planning and Scheduling	3
9	Action Planning and Scheduling	3

1 Introduction

2 Selective Symbolic Execution

Symbolic execution is an advanced analysis technique particularly suited for automated software testing [\[1\]](#) and malware analysis [\[2\]](#). Instead of concrete input (7, “string”, ...) symbolic execution uses symbolic values (λ , β , ...) when processing code. Assignments in the program path have impacts on these symbolic values. The integer calculation $x = x - 2$, for instance, would update the symbolic expression representing the input x to $\lambda - 2$. Conditional statements (if <condition> then ... else ...) fork program execution into two new paths. Both paths are then constrained by an additional condition, the ‘then’ branch with the if-condition and the ‘else’ branch with the negated if-condition respectively.

Following this procedure results in a tree-like structure of constrained symbolic expressions. A constraint solver can now take all these constraints along one execution path as input and find one concrete input (e.g., $\lambda = 5$) which would lead to the program following just this path. This alleviates writing reproducible test cases [\[3\]](#).

cite!

cite

cite

- 3 bla3
- 4 Design Alternatives
- 5 Project Team
- 6 Action Planning and Scheduling
- 7 Action Planning and Scheduling
- 8 Action Planning and Scheduling
- 9 Action Planning and Scheduling