정적 분석기를 골탕 먹이는 입력 프로그램 합성 기술

(Program Synthesis for Attacking Static Analyzers)

지도교수 : 이광근

이 논문을 공학학사 학위 논문으로 제출함.

2025년 10월 29일

서울대학교 공과대학

컴퓨터공학부

정원준

2026년 2월

정적 분석기를 골탕 먹이는 입력 프로그램 합성 기술

(Program Synthesis for Attacking Static Analyzers)

지도교수 : 이광근

이 논문을 공학학사 학위 논문으로 제출함.

2025년 10월 29일

서울대학교 공과대학

컴퓨터공학부

정원준

2026년 2월

국문초록

프로그래밍 언어와 그 실행 의미, 정적 분석기가 주어졌을 때 정적 분석기가 무의미한 결론을 내리게 하는 입력 프로그램을 합성한다. 무의미한 결론이라 함은, 프로그램의 특정 위치에서 특정 변수가 가질 수 있는 값에 대한 정보가 아무것도 없는 것을 의미한다. 본 연구에서는 적은 종류의 규칙을 가지는 G 언어와 그에 해당하는 한 정적 분석기를 고정하여 분석기에 대한 공격을 시도한다. 이러한 공격이 왜 항상 가능한지 증명하고, 전탐색에 기반한 방법으로 실제 공격 예시를 제시한다. 이후 프로그램 합성 기법을 이용하여 유의미한 실행 시간 안에 공격을 시도한다. 이 공격의 결과 통해 정적 분석기의 불완전한 부분을 파악하여 분석기를 보완하는데 도움을 줄 수 있다. 실행 의미는 같지만 정적 분석기가 다른 결론을 내리는 프로그램을 만들어 난독화 기법으로 사용될 수 있다.

주요어: 정적 분석, 프로그램 합성, 요약 해석, 난독화

목 차

제 1 장 배경 지식

제 1 절 정적 분석

제 2 절 안전함

제 3 절 Rice의 정리

제 4 절 프로그램 합성

제 2 장 문제 정의

제 3 장 가능성

제 4 장 기대효과

제 5 장 G 언어

제 1 절 문법

G 언어의 문법은 다음과 같다:

$$
\begin{array}{rcl}
P & \to & C \\
\\
C & \to & x := E \\
 & | & C; C \\
 & | & \text{ifp } E\ C\ C \\
 & | & \text{while } E\ C \\
\\
E & \to & n \quad (n \in \mathbb{Z}) \\
 & | & x \\
 & | & E{+}E \\
 & | & E{*}E \\
 & | & {-}E
\end{array}
$$

이 때 P는 프로그램, C는 커맨드, E는 식을 의미한다. 모든 식은 정수값을 갖는다. (3) **Rice의 정리**에서 사용한 전략들을 적용하기 용이한 언어로 구성하였다. while 문을 통해서 튜링 완전을 확보하였다. 이 내용은 G 언어의 실행의미를 정의한 후에 더 자세히 다루겠다.

제 2 절 실행의미

G 언어의 실행 의미는 무엇 의미구조$_{denotational\ semantics}$로 정의한다. 실행의미가 정의되는 규칙은 다음과 같다:

$$
\begin{aligned}
Val &= \mathbb{Z} \\
n &\in Val \\
\sigma &\in Env = Var \xrightarrow{\text{fin}} Val \\
x &\in Var \\
P, C &\in Cmd \\
E &\in Exp \\
[\![\cdot]\!] &: Cmd \to Env \to Env \\
&\quad + Exp \to Env \to Val
\end{aligned}
$$

$$
\frac{}{[\![n]\!]\sigma = n} \qquad \frac{}{[\![x]\!]\sigma = \sigma(x)} x \in \mathrm{dom}(\sigma)
$$

$$
\frac{[\![E_1]\!]\sigma = n_1 \quad [\![E_2]\!]\sigma = n_2}{[\![E_1 + E_2]\!]\sigma = n_1 + n_2} \qquad \frac{[\![E_1]\!]\sigma = n_1 \quad [\![E_2]\!]\sigma = n_2}{[\![E_1 * E_2]\!]\sigma = n_1 \times n_2}
$$

$$
\frac{[\![E]\!]\sigma = n}{[\![-E]\!]\sigma = -n}
$$

이 때, 전체 프로그램의 실행의미는 $[\![P]\!]\emptyset$으로 정의된다.

제 6 장 분석기 고정

제 7 장 문제 고정

제 8 장 전탐색

제 1 절 구현

제 9 장 결과

제 10 장 결론

Abstract

Given a programming language, its operational semantics, and a static analyzer, we synthesize an input program that causes the static analyzer to yield a meaningless conclusion. A meaningless conclusion refers to a state where there is no information about the possible values of a specific variable at a specific location in the program. In this study, we fix the G language, which has a small set of rules, and a corresponding static analyzer, to attempt an attack on the analyzer. We prove why such an attack is always possible and present a concrete example of the attack using a full-search-based method. Subsequently, we attempt the attack within a meaningful execution time using program synthesis techniques. The results of this attack can help identify incomplete aspects of the static analyzer, thereby aiding in its refinement. By creating programs that have the same operational semantics but lead to different conclusions by the static analyzer, they can be used as an obfuscation technique.

Keywords: Static Analysis, Program Synthesis, Abstract Interpretation, Obfuscation