P R O J E C T D O C U M E N T

기존 난독화 도구와의 비교

Test Result Document

Project Name	코드 난독화 도구 제작

15 조

202002562 조인우
202002508 손지웅
201902686 노형우

지도교수: 조은선 교수님 (서명)

Table of Contents

1. Introduction

1.1. Objective

이 문서는 본 연구에서 개발한 코드 난독화 도구에 대해 수행한 정량적 실험 결과를 중심으로 구성되었으며, GPT-4o와 같은 최신 LLM을 대상으로 한 복원 저항성 평가, 전통적인 난독화 도구(Tigress)와의 성능 비교, 사이클릭 복잡도 및 실행 시간 증가율 분석, 역난독화 성공 여부 판별 등 실험 설계와 결과 해석에 초점을 맞추었다. 다양한 조건에서의 테스트를 통해 난독화 기법의 효과를 정량적 · 정성적으로 검증하고, 이를 바탕으로 성능 한계와 향후 개선 방향을 제시한다.

2. Experiment Result Report

1. 서론

1.1 실험 개요

- 개발한 난독화 도구가 LLM 기반 역난독화(GPT-4o)에 대해 높은 복원 저항성을 가지는지 검증하고, 기존 도구(Tigress)와 비교하여 구조적 난독화의 효과를 분석한다.
- 입력 데이터: 난독화 도구의 주요 기법이 원활히 적용될 수 있도록, 중첩 조건문 및 반복문을 포함한 miniC 코드 샘플 1개
- 실험 환경: gcc v13.3.0(컴파일), Ubuntu 22.04 LTS(코드 실행 및 시간 측정), Tigress v4.0(기존 난독화 도구), GPT-4o(역난독화 도구), Lizard 라이브러리(파이썬, 사이클릭 복잡도 계산)

1.2 실험 방법

1. 동일한 원본 코드에 대해서 여러 난독화 옵션을 각각 적용
2. 난독화된 코드를 컴파일 후 1000회 평균 실행 시간 측정
3. Lizard 라이브러리를 이용해서 사이클릭 복잡도 측정
4. LLM을 통해 역난독화

평가기준

1. 사이클릭 복잡도: 코드 제어 흐름 복잡도 지표, 난독화 전후의 복잡도 증가를 정량적으로 계산
2. 실행 시간: 성능 오버헤드 평가
3. LLM 역난독화 여부: GPT-4o가 의미를 복원했는지 여부로 판단, 난독화를 조작된 코드가 남아있는지에 따라서 O/X

2. 테스트 결과 상세

2.1 테스트 결과 개요

기법 \ 평가지표		실행 시간(ms)	사이클릭 복잡도	LLM 역난독화 여부
원본		13.80ms	4	-
ObfusTree		21.28 (+54.20%)	15 (+275%)	O
Tigress	Flatten	93.15 (+574.28%)	39 (+875%)	O
	AddOpaque	21.66 (+56.96%)	22 (+450%)	O
	Split	19.36 (+40.29%)	16 (+300%)	O
Obfus Tree + Tigress	Flatten	136.22 (+887.10%)	70 (+1650%)	X
	AddOpaque	20.87 (+51.23%)	34 (+750%)	X
	Split	25.40 (+84.06%)	28 (+600%)	X

ObfusTree -> 개발한 난독화 도구

2.2 테스트 결과 상세 분석

- 개발한 도구와 기존 난독화 도구 단독 적용은 모두 LLM에 의해 역난독화가 성공적으로 수행되었

는데, 이는 개발한 도구의 경우 실험에 사용된 코드가 상대적으로 짧고 구조가 단순하여 LLM이 의미를 쉽게 추론할 수 있었던 반면, 기존 난독화 도구의 경우 CTF 문제나 공개 리포지토리 등을 통해 유사한 난독화 패턴이 이미 학습되었을 가능성이 높기 때문으로 해석된다.

- 기존 난독화 도구 + AddOpaque 조합은 예상과 달리 실행 시간이 단독 적용보다 오히려 감소하였다. 이는 컴파일러 최적화나 제어 흐름 간소화 등의 영향으로 보이며, 데이터 오류보다는 구조적 상호작용에 의한 결과로 해석된다.
- 기존 난독화 도구 + Flatten 조합은 실행 시간이 과도하게 증가(+887%)하였는데, 이는 두 기법이 복잡한 흐름을 중복 삽입하면서 성능 비효율이 발생했기 때문으로 보인다.
- LLM 역난독화 실패는 모두 병합된 난독화 기법에서 발생하였으며, 이는 구조가 지나치게 복잡해져 GPT-4o가 의미를 복원하지 못했음을 의미한다.

2.3 실험 결과의 한계와 위협 요인

- miniC 기반 코드만 사용되어, 복잡한 실무 코드나 다른 언어에 대한 일반화는 아직 힘든 것으로 보인다.
- 정적 파스 트리 기반 구조 변형만 수행하도록 설계되어 있으며, 동적 실행 흐름이나 런타임 정보에 따라 변형되는 코드는 다루지 않는다. 즉, 동적 코드 난독화에는 적용할 수 없다는 제약이 존재한다.
- 재현 환경이 갖는 한계: GPT-4o 응답은 비결정적이며, date 명령어 기반 실행 시간 측정은 시스템 부하에 따라 편차가 발생할 수 있다.

3. 결론

본 연구를 통해 개발한 도구는 LLM 기반 역난독화에 대해 단독 적용 시에는 취약하지만, 기존 도구와 병합 시 복원 저항성이 크게 향상됨을 확인하였다. 구조적 난독화 기법은 균형적인 실행 오버헤드와 복잡도 증가 효과를 제공하며, LLM이 구조를 추론하는 데 어려움을 유발하는 것으로 나타났다.

향후에는 포인터, 함수 호출 등 고급 구문 확장, 다양한 언어 지원, LLM 프롬프트 변화 대응 난독화 설계 등을 통해 실용성과 범용성을 높일 수 있다.

3.AI 도구 활용 정보

사용 도구	GPT-4o
사용 목적	어휘 선택 및 문장 흐름 정리
프롬프트	● 이 글을 다듬어줘
반영 위치	문서 전체
수작업 수정	있음(의도하지 않게 추가된 내용 삭제)