Enumeration Synthesis of Instruction Leakage Enhanced by Large-Language Models

Antonio Napoli, PhD student, KTH Royal Institute of technology
Dept. of Computer Science, Division of Theoretical computer science
Supervisors: Prof. Roberto Guanciale, Prof. Musard Balliu

Motivation & Research Goals

Introduction

Instruction leakage in processors poses a critical challenge for secure computing. Observational equivalence models, like constant-time execution, often miss subtle side-channel vulnerabilities.
For example, assembly instructions such as:

```
mul     x7, x14, x8 # Potential leakage: value of x14, x8
add     x7, x18, x2
ldr     x1, [x23, x12] # Potential leakage: address x23+x12, or part of it x23+x12>>6 / x23+x12>>4
```

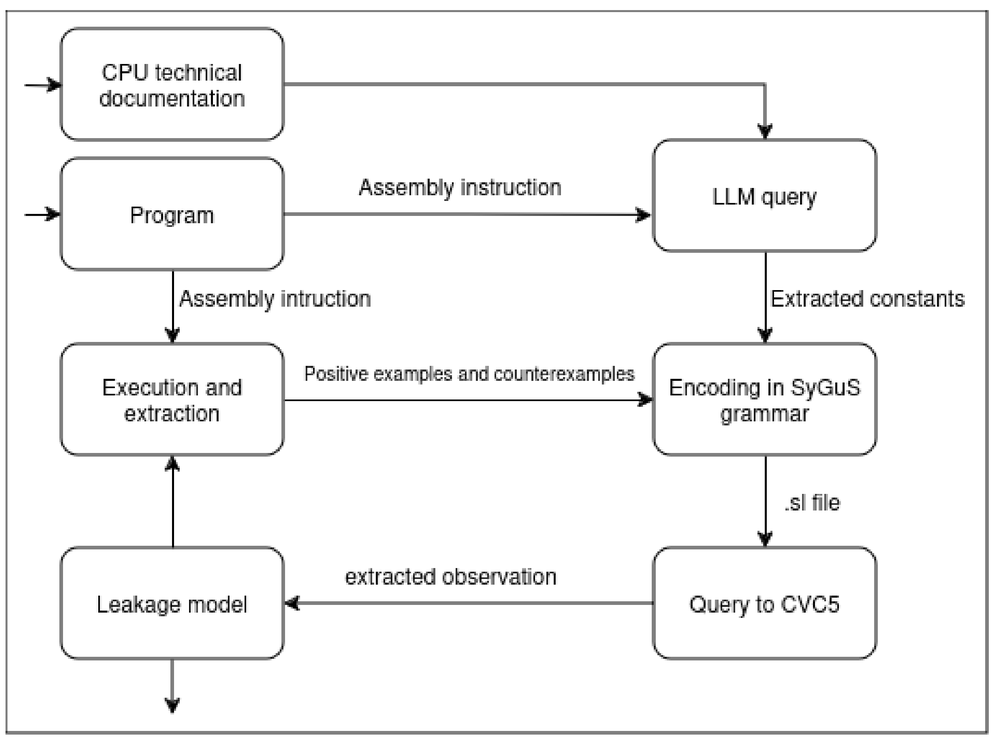
may leak information through cache or execution time, making it hard to pinpoint the source.
This project aims to address these gaps by:

- Analyzing assembly snippets with varied initial states.
- Using synthesis-by-example to uncover leakage patterns.
- Leveraging LLMs to speed up the synthesis process.

Goal: Identify the leakage pattern and refine the leakage models incorporating the observation found.

Methods

- Experiment Setup: Execute a target assembly snippet multiple times with varying initial register states, capturing traces of machine states after each instruction.
- Leakage Analysis: Compare traces to detect observational equivalence or leakage, marking examples as valid or counterexamples.
- Synthesis-by-Example: Use SyGuS and CVC5 to synthesize leakage patterns, leveraging bitvector grammars to model registers.
- LLM Integration: Extract instruction details from processor documentation to enhance synthesis efficiency.



Selected Results

The synthesis process experienced delays due to the overhead of constant generation.

```
ARM7TDMI technical reference
For the multiply instructions, the number of cycles needed
depends on the operand value.
    - m equals 1 if bits [32:8] are all zero or one
    - m equals 2 if bits [32:16] are all zero or one
    - m equals 3 if bits [32:24] are all zero or one
    - m equals 4 otherwise
```

Preliminary experiments demonstrated: Identification of previously unmodeled leakages and Enhanced synthesis speed and accuracy with LLM assistance

```
...
mul     x7, x14, x8
add     x7, x18, x2
ldr     x1, [x23, x12]
```

- Data Generation: Execute the code snippet to create 40 positive examples and 20 counterexamples.
- LLM Query: Use ChatGPT ("o1 mini") to retrieve specific hexadecimal constants, refining the synthesis grammar.
- SyGuS File Creation: Generate a SyGuS file using the examples and enhanced grammar, treating all registers as Bit-Vectors.
- Experiment Setup: Test three grammar configurations: Full Grammar, LLM Enhanced Grammar, LLM Enhanced Grammar.

Grammar	Execution Time	Leakage Detected
Full	timeout	None
LLM Enhanced	timeout	None
LLM Enhance /constant	9.746s	$(x23 + x12) >> 6$

Table 1: Example results from experiments.

References

[1] Enhanced Enumeration Techniques for Syntax-Guided Synthesis of Bit-Vector Manipulations
Ding, Yuantian and Qiu, Xiaokang Årzén
Proceedings of the ACM on Programming Languages, Volume 8, Issue POPL

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM