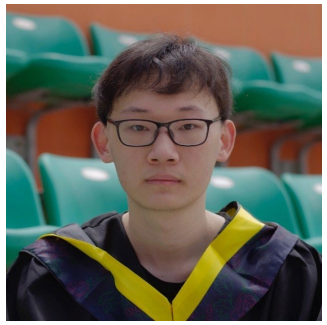
MANCHESTER 1824
The University of Manchester

VeriExploit: Automatic Bug Reproduction in Smart Contracts via LLMs and Formal Methods

Chenfeng Wei
Chenfeng.wei@machester.ac.uk
The University of Manchester

Co-Authors



Mr.
Chenfeng Wei

Mr.
Shiyu Cai

Mr.
Yiannis Charalambous

Mr.
Tong Wu

Dr.
Sangharatna Godboley

Dr.
Lucas C. Cordeiro

MANCHESTER 1824
The University of Manchester

Motivation

Reproducing found bugs can be hard

- Bug finders (e.g. verifiers) detect issues but rarely provide executable reproductions.
- Many bugs are cross-contract and multi-step (e.g. reentrancy)
  - ➢ hand-crafting attacker contracts + call sequences is slow and error-prone.
- In Solidity smart contracts, reproduction is more challenging
  - ➢ requires an attacker contract and a clear exploit path/trace.

MANCHESTER 1824
The University of Manchester

Why Reproducing Bugs?

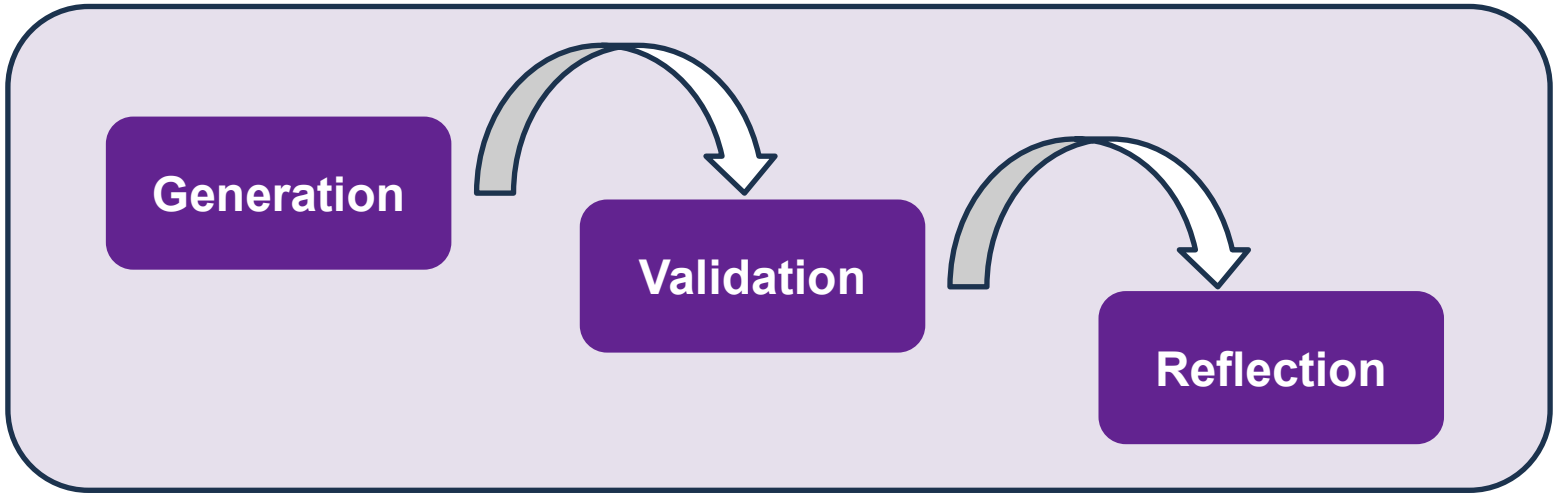
- **Understanding & Fixing**: Generate attacker contracts to help developers understand vulnerabilities.
  - ➢ (Ballesteros et al., 2022), (Wu et al., 2024).
- **Detection & Forecasting**: Pair adversarial with vulnerable contracts to detect/anticipate attacks.
  - ➢ (Wang et al., 2024)
- **Vulnerability Discovery**: Generate exploit PoCs to validate the (LLM-based) bug findings.
  - ➢ (Gervais & Zhou, 2025), (Xiao et al., 2025)

Our primary goal is *Understanding*, but the generated contracts are also suitable for *Forecasting*

MANCHESTER 1824
The University of Manchester

Our Solution: VeriExploit

- LLM-guided generation of attacker contracts.
- Bounded cross-contract verification (BCCV).
- Automatically validates exploits and reports executable steps.
- Generalizes across multiple vulnerability types.



MANCHESTER 1824
The University of Manchester

Algorithm Overview

Given:

- Vulnerable and attacker contracts.
- A verifier that supports unbounded reasoning.

VeriExploit checks:

- If an attacker path reproduces the reported bug, we provably find it.
- Otherwise, we formally diagnose why: **unreachable** or **incorrect parameters**. (args, caller, value, pre-state)

Key Numbers

Dataset: **100** samples from open-sourced contacts, covering **six** bug types.

85.6%
Overall success (ESBMC)
Validated exploits across all benchmarks

64.6%
Overall success (SolCMC)
Performance on the same dataset

11.8%
LLM-only baseline
Without verifier feedback

+20.6 pp
Self-reflection gain
From 65.0% to 85.6% (ESBMC)

90.9%
AdvScanner comparison
vs 88.5% (AdvScanner)

Evaluation: 5 trials per case, GPT-4o, reflection limit = 5, per-run timeout = 120 s.

MANCHESTER 1824
The University of Manchester

Thank you!