STATIC ANALYISIS OF SOLIDITY SMART CONTRACTS

Nikhil Naik & Dr Naipeng Dong

Introduction

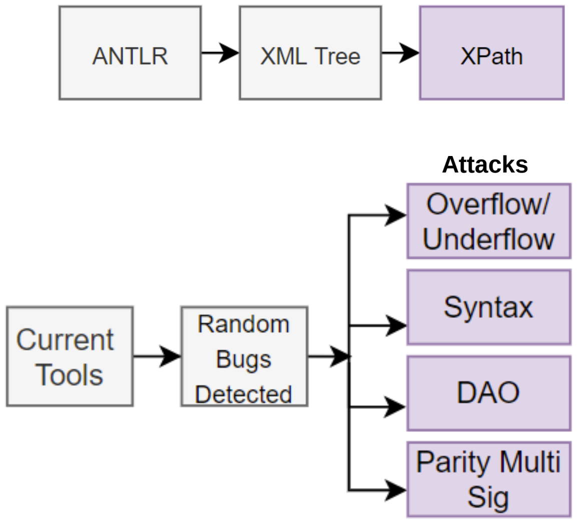
- ► Static Program Analysis Detects Bugs/Vulnerabilities in code
- ► Solidity Smart Contracts deployed on Ethereum Blockchain
- ► Re-Deploying Buggy Smart Contracts is expensive
- ► Solidity is a new programming language with new Attacks and Vulnerabilities being discovered rapidly

Background

► Current static analysis tools use Parse Trees for detection logic with an ANTLR parser

Limitations:

- Random bugs detected poor attack coverage
- Poor System compatibility and high dependencies
- No bug solution and poor UI/UX to view analysis results



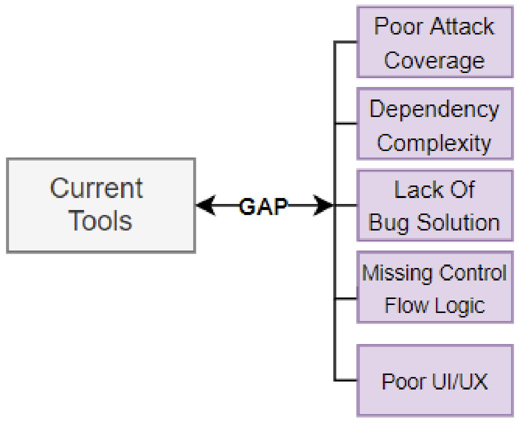
Project Goals & Aims

► Build Python based Static Analysis tool overcome limitation of current tools

► Construct Control Flow Logic for bug detection of bugs detected

► Evaluate current & project tools performance on deployed contracts

Attacks Coverage:
Overflow/Underflow, Syntax
DAO

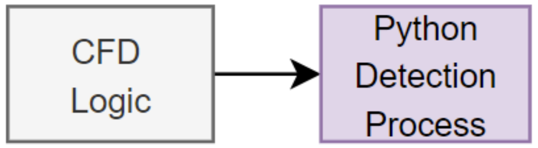


Approach & Execution

► Control Flow Diagram **(CFD)** Logic informs code functions steps

► Check bug patterns or missing precaution measures

► Develop Solidity code variations of each bug for testing



► 6 Overflow/Underflow, 24 Syntax and 5 DAO attack theme bugs detected in GUI program

► **35 Python bug detect functions process:**

- 1st Scan contract store variable names, types and data structures
- 2nd Check if bug condition violates **(CFD)** conditions
- Log bug name, Line in code, solution, impact and accuracy
- Produce contract safety rating (%) based on impact and accuracy

Testing

► Comparative Analysis of existing tools:

- **Linux OS Virtual Machine:** Solint, Solidity Scan & sFuzz
- **High Dependency Tools:** Smart Check & Remix
- **PySolSweep*** (Projects Proposed Tool)

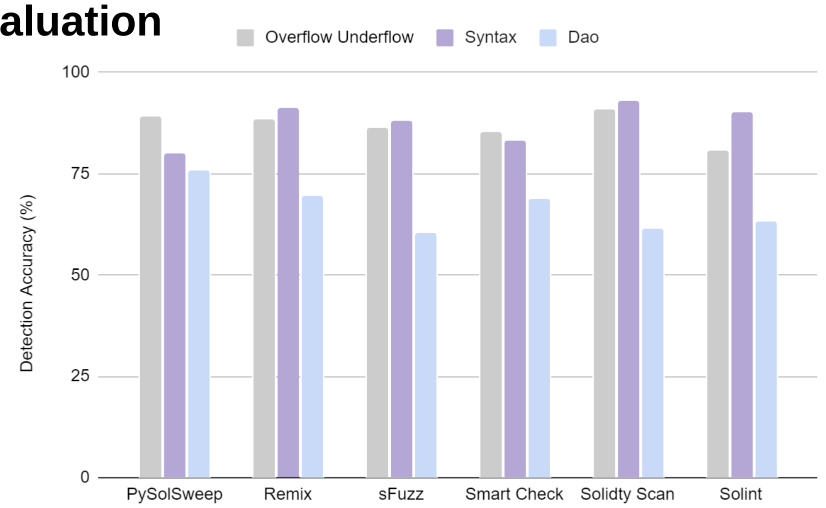
► **100** General, **50** DAO Withdraw Function Deployed on Etherscan Smart Contracts code using academic papers **Benchmark Criteria**:

- Minimum 200 LOC, Multiple Complex Functions, numerical operations, loops, gas usage, data structures and libraries

► **Independent Variable:** Deployed Test Contracts

► **Dependent Variable:** Number of Bugs, Bug Attack Theme and False Positive Rate (CFD Logic Crosscheck)

Evaluation



Tool	Overflow/Underflow	Syntax	Dao	Total Verified Bugs
PySolSweep*	*1859*	*2024*	*416*	*4299*
Remix	*489*	*1090*	*61*	*1640*
sFuzz	*1282*	*1860*	*62*	*3204*
Smart Check	*316*	*1076*	*12*	*1404*
Solidity Scan	*129*	*527*	*5*	*661*
Solint	*148*	*690*	*49*	*887*

► **Projects Tool achieves proposed aims**:

- **Limitation** Slightly Lower Accuracy
- Increased Overflow/Underflow, Syntax and DAO bug coverage

► Existing tools only detected 1 DAO bug variant

► PySolSweep tool to detect check-effect-interact and block reentrancy modifier violations for DAO attack

► High volume of compromised deployed smart contracts

Conclusion

► Control Flow Diagrams logic for bug detection

► Python based minimal dependency Static Analysis tool

- UI/UX GUI Design
- Store Static Analysis results in a file
- Suggestions to resolve bug/vulnerability in code

► Systematic Approach of ***Coverage*** Against Attacks Themes

► Analysis of existing static analysis tools on deployed smart contracts

► DAO bugs poorly covered in existing tools

► Future work integrate attack theme coverage of bugs to ANTLR Parser analysis for increased bug detection accuracy

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

School of Information Technology & Electrical Engineering

INNOVATION EXPO