Automated Attack Synthesis for Constant Product Market Makers

Sujin Han, Jinseo Kim, Sung-Ju Lee, Insu Yun

ACM SIGSOFT ISSTA

June 27, 2025

KAIST

Decentralized Finance (DeFi)

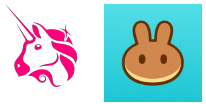
- Financial ecosystem powered by blockchains
- Scope: DeFi on EVM-based blockchains
  - Ethereum
  - Binance Smart Chain

- No central intermediaries → trust is encoded in smart contract code
- Provides new financial services not possible in traditional centralized finance

Building Blocks of DeFi

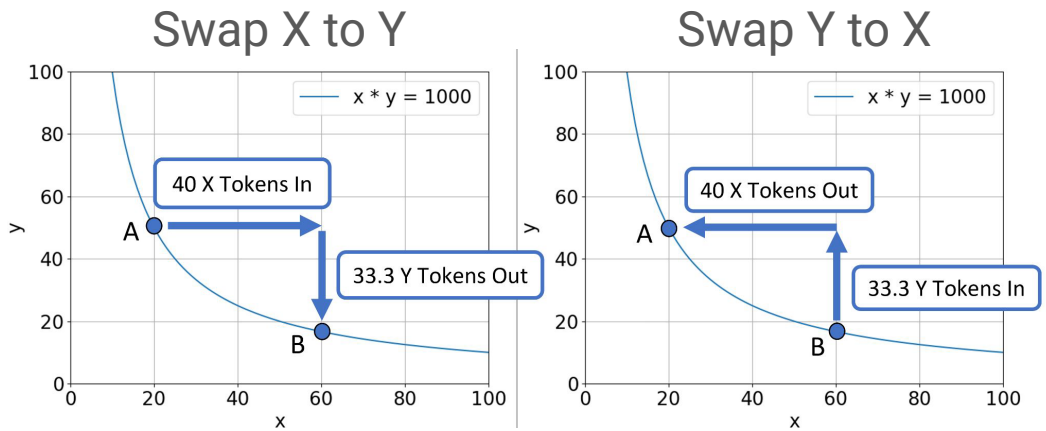
ERC20 Tokens 

- Standard for fungible tokens on EVM-based chains
- Implements interfaces like `transfer()` and `balanceOf()`
- Enables interoperability between various tokens with financial services (ex. DEX)

Decentralized Exchanges (DEX) 

- Smart-contract-based market for exchanging tokens
- Popular model: **Constant Product Market Maker (CPMM)**

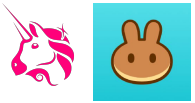
Constant Product Market Makers (CPMM)



- Model adopted by major DEXes (ex. Uniswap , PancakeSwap)
- Maintain reserves of two tokens: **X** and **Y**
- Enforce invariant: $x \cdot y = k$

CPMM DEXes are often targets of exploits

- CPMM DEXes store billions in assets (as of June 2025)
  - Uniswap TVL: **$5.2 billion**
  - PancakeSwap TVL: **$1.8 billion**



- Token behavior incompatible with the CPMM model can be exploited to drain tokens from DEX
- Real-world incidents are **frequent** and **severe**:
  - **138 CPMM DEX exploits** reported by BlockSec in Feb 2023 alone
  - ThoreumFi-BNB (January 2023) - **$580K** stolen

Many tools have been suggested to detect DeFi bugs

General-Purpose Vulnerability Detectors

- Echidna (Grieco et al.)
  - Grammar-based fuzzing
  - Uses static analysis (Slither) to guide input generation
- ItyFuzz (Shou et al.)
  - Snapshot-based fuzzing
  - Leverages dataflow and comparison waypoints

Specialized Tools

- DeFiTainter (Kong et al.)
  - Taint analysis
  - Focused on price manipulation vulnerabilities

Detecting CPMM-incompatible behavior is challenging

Motivating Example: ANCH Exploit

Loss: ~$20K

Incompatible Behavior

- ANCH rewarded 0.05% of transfer amount every time when transferring to/from DEX

Exploit

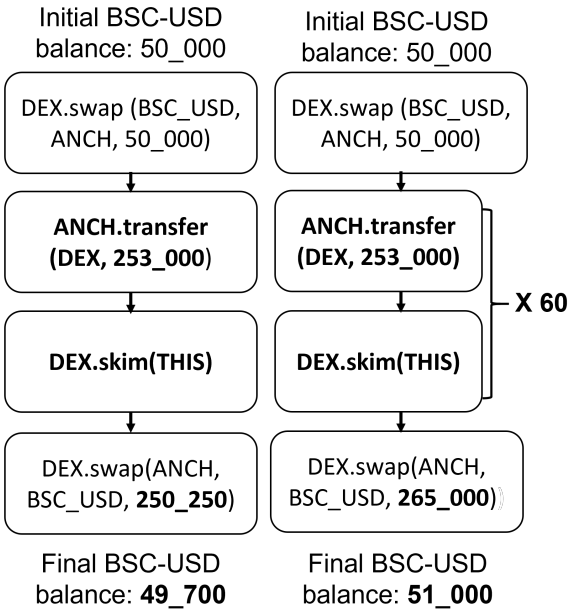
- Loop transfers to gain a significant amount of ANCH reward tokens for free
- Drain stablecoin from CPMM DEX trading ANCH tokens

```
uint public rewardRate = 5;
uint public percent = 10000;
uint public minAmount = 10000 * 1e18;
function giveReward(address receiver, uint amount)
      private {
  if (amount > minAmount) {
    rewardAmount = amount * rewardRate / percent;
    balances[receiver] += rewardAmount;
  }
}
function transfer(address sender, address receiver,
      uint amount) public {
  if (sender == DEX_ADDR) {
    giveReward(receiver, amount);
  } else if (receiver == DEX_ADDR) {
    giveReward(sender, amount);
  }
  balances[sender] -= amount;
  balances[receiver] += amount;
}
```

Simplified ANCH code

Detecting CPMM-incompatible behavior is challenging

- Not all reward mechanisms break CPMM → hard to design a one-size-fits-all oracle
- Need concrete exploit traces (i.e., profitable transaction) to confirm issues
- Tools like Echidna and ItyFuzz aim to maximize coverage → likely to miss exploits that require repetitions



ANCH exploit without repetition (left) and with repetition (right)

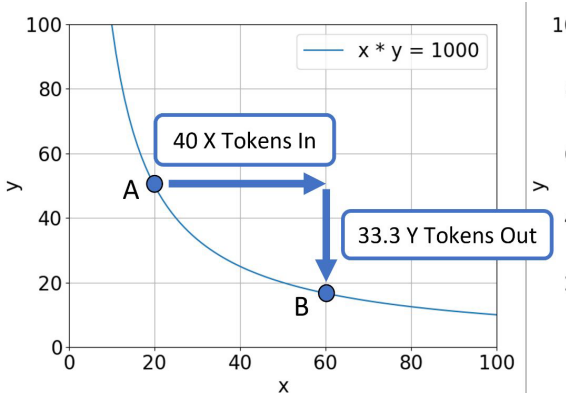
How can we reliably detect CPMM bugs on blockchains?

- Formally defined CPMM composability bugs
- Identify two invariants that, when broken, can lead to token drain
- Propose *CPMMX* that can automatically detect CPMM composability bugs
  - Employs a novel approach *shallow-then-deep search* to efficiently identify vulnerable contracts across entire blockchains
  - Generates end-to-end exploits (no false positives!)

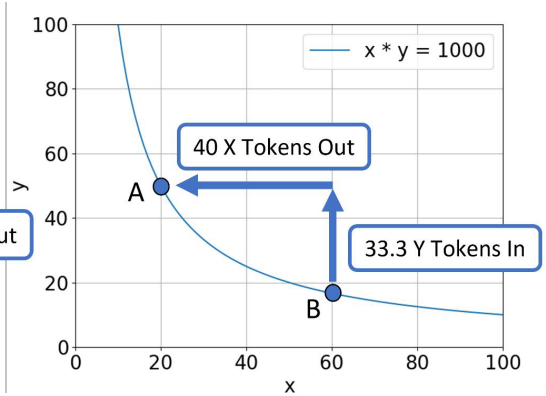
Defining CPMM Composability Bugs (CPMM Bugs)

- Consider **X token**, **Y token** and a **CPMM** trading them
- A *CPMM composability bug* is a flaw in the **Y token** contract that lets an attacker illegitimately extract **X tokens** from the **CPMM**

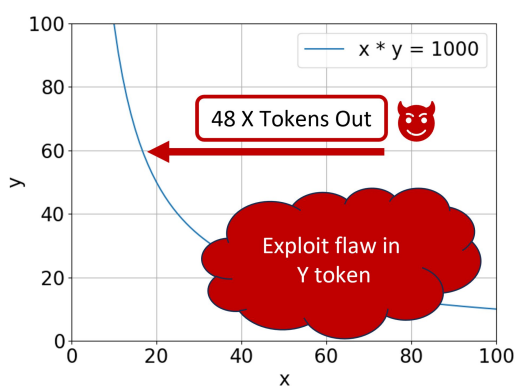
Benign: Swap X to Y



Benign: Swap Y to X

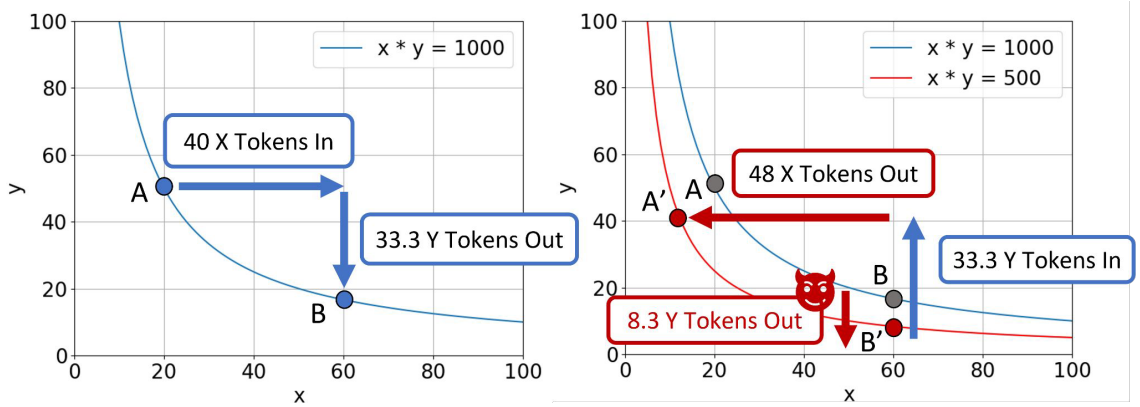


Malicious: Swap Y to X



→ Such flaws can be categorized into **two types**

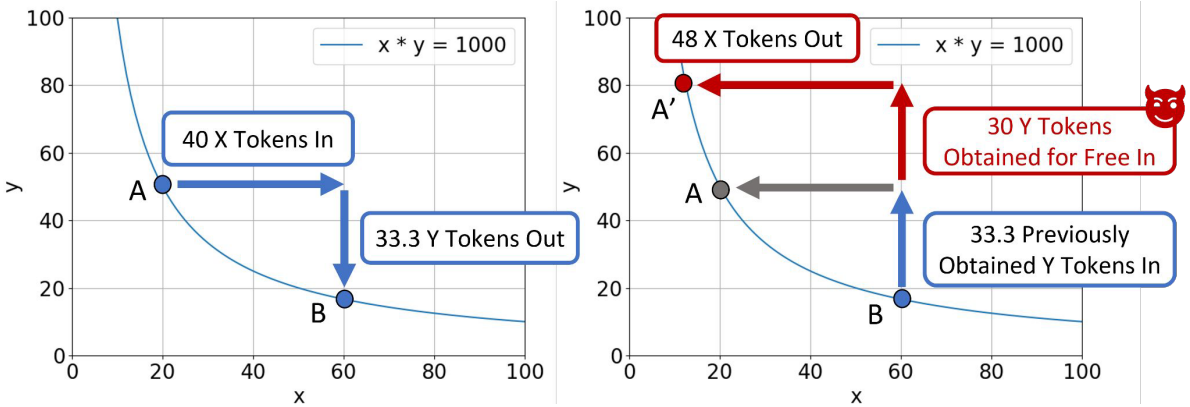
Type 1: DEX Y Token Balance Decrease



- Flaw in Y token lets attacker remove Y tokens from CPMM
- This shifts the curve inward → Y price rises
- Attacker gets **more X tokens** for the same amount of Y tokens

Invariant 1. Users should not be able to remove tokens in CPMM.

Type 2: Attacker Y Token Balance Increase

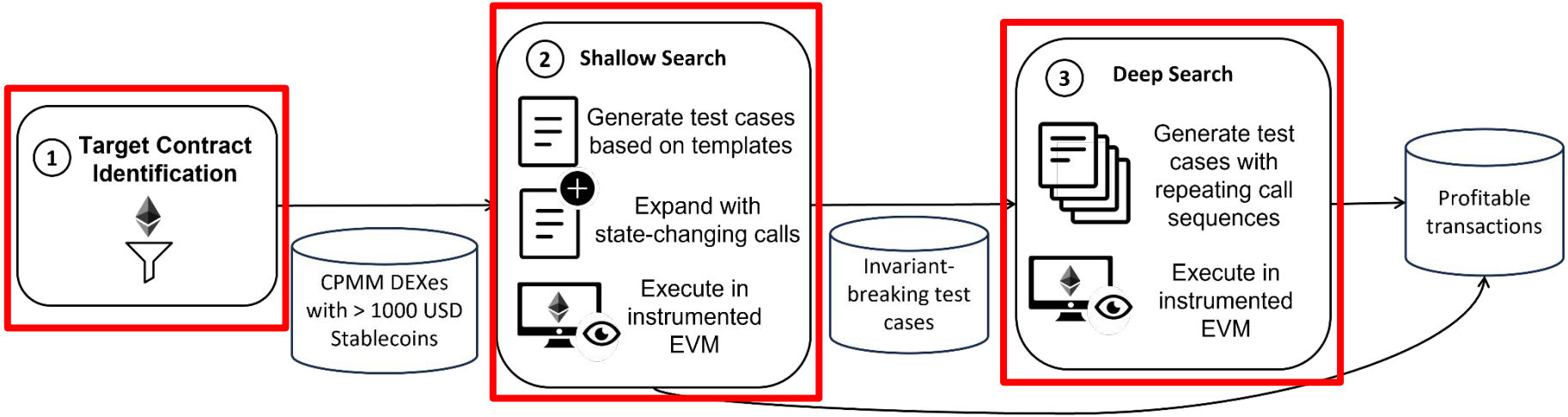


- Flaw in Y token lets attacker obtain Y tokens for free
- Attacker uses Y tokens to drain more X tokens from CPMM

Invariant 2. Users should not be able to obtain tokens traded in CPMM for free.

CPMMX: CPMM eXploiter

- Based on previous findings, we built a tool that automatically detects CPMM bugs across entire blockchains



① Target Contract Identification

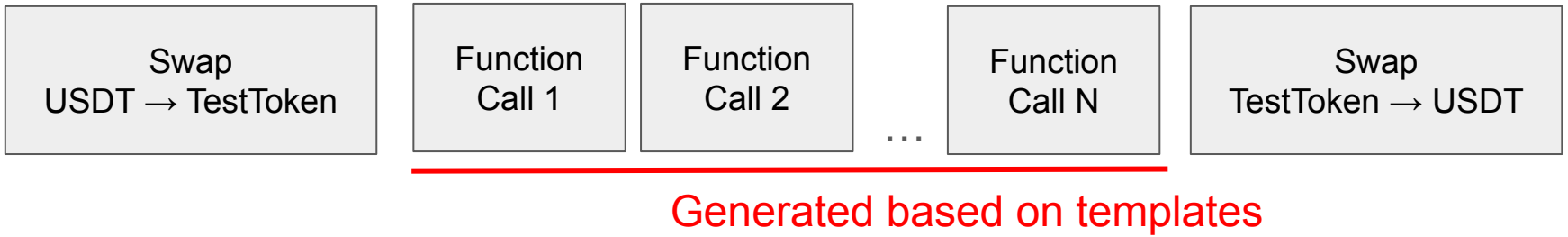
- Given a list of CPMM contracts, *CPMMX* filters contracts that meet the following criteria:
  - One of the traded tokens is a well-known coin (ex. WETH, USDT)
  - CPMM contains over $1,000 worth of well-known coin

- Ex. CPMM contract trading USDT and TestToken
  - Balance: (10,000 USDT, 200,000 TestToken)

② Shallow Search for Finding Invariant Breaking Transactions

- Generates test cases likely to break invariants

Test Case Format



Generating Test Cases in Shallow Search

- Observed that many CPMM-incompatible behavior stem from incentive mechanisms in tokens
  - Rewards
  - Fees
  - Burn (remove tokens from circulation)
- Designed templates likely to trigger such mechanisms
  - **Cross-trading**: circular token transfers that may activate incentive logic
  - **Burn**: function calls with `burn` in the name

Type	Testcase	Description
Cross-trading	**`transfer(this, transferAmount)`**	Send tokens to ourselves
	`transfer(DEX, transferAmount)` **`DEX.skim(this)`**	Send tokens back and forth between DEX and ours

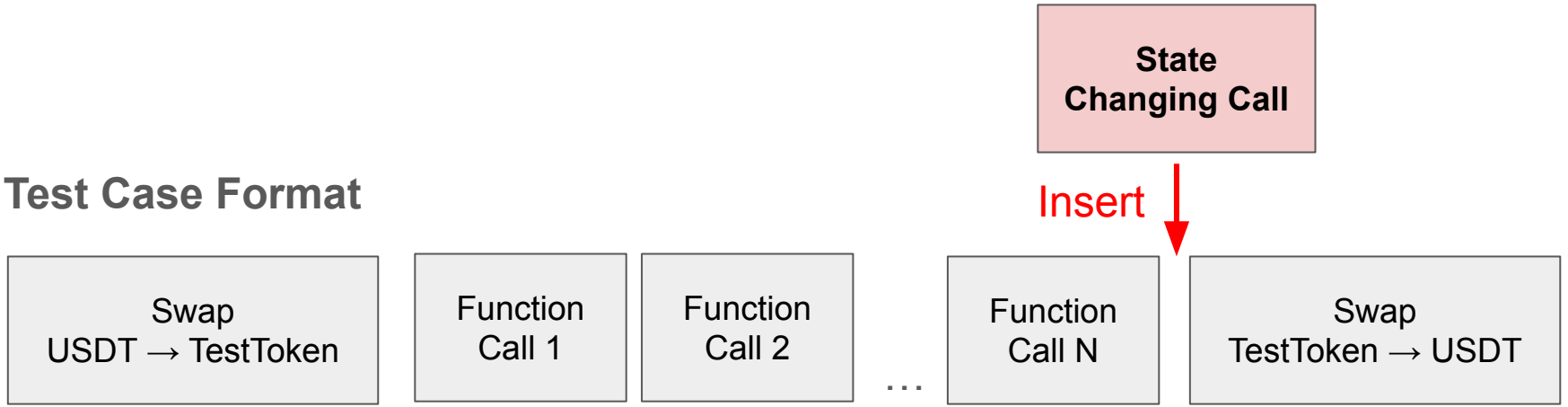
	`transfer(DEX, transferAmount)`
	`DEX.skim(DEX)`
	`pair.skim(this)`
		Send tokens to DEX, DEX sends tokens to itself, DEX sends tokens to ours

	`transfer(DEX, transferAmount)` **`DEX.skim(DEX)`**	Send tokens to DEX, DEX sends tokens to itself
Burn	**`burn(burnAmount)`**	Remove tokens from circulation
	`burn(DEX, burnAmount)`	Remove tokens from DEX

Templates for generation test cases

Generating Test Cases in Shallow Search

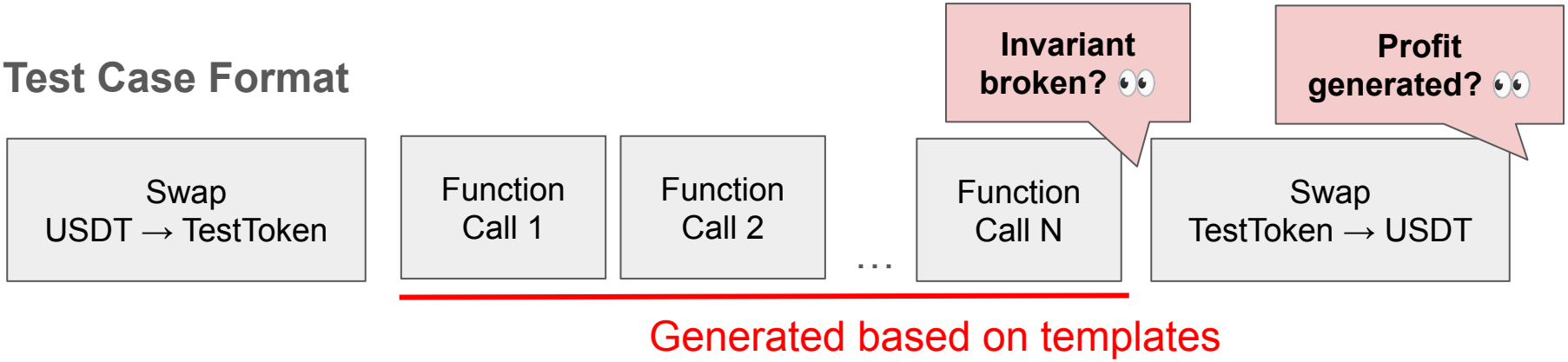
- If no invariant-breaking transaction found, integrate **state changing calls**:
  - small amount transfers (ex. `transfer(1)`)
  - non-view, no-argument function calls (ex. `distributeFee()`)



Shallow Search for Finding Invariant Breaking Transactions

- Run test cases in instrumented EVM to check for invariant violations & profit
- Three possible outcomes:
  1. Profit → early termination (vulnerable)
  2. Invariant violation but no profit → proceed to *deep search*
  3. No invariant violation and no profit → early termination (not vulnerable)

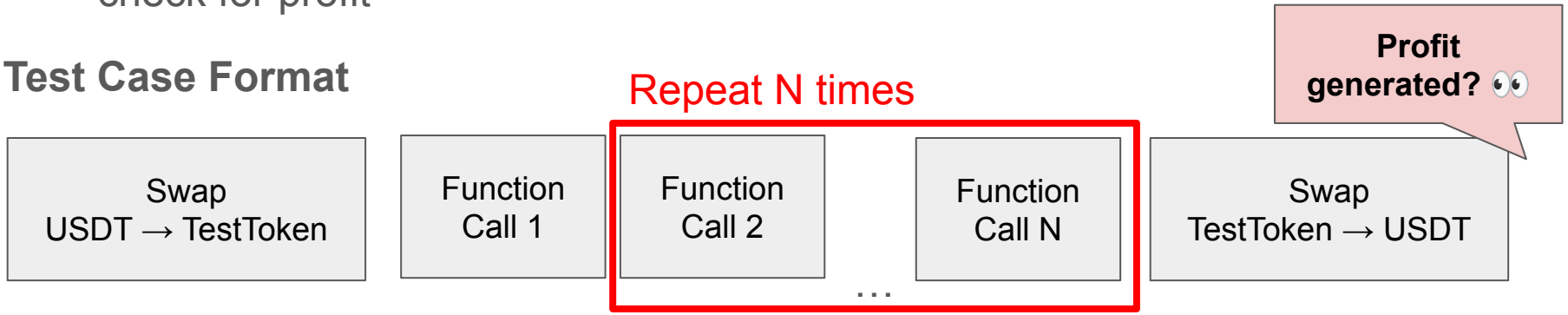
Test Case Format



③ Deep Search for Generating Profitable Transactions

- Enter deep search phase if *CPMMX* found only invariant-breaking test cases
- Generates test cases by repeating invariant-breaking call sequences and check for profit

Test Case Format



- Decide on which test case to prioritize based on final stablecoin balance
  1. Balance increases → keep increasing repetitions
  2. Balance decreases → limit repetitions, but keep test case
  3. No change after increasing repetitions 3 times → discard test case

Evaluation

RQ1. How **effective** is *CPMMX* at detecting CPMM bugs? ←

RQ2. How **efficient** is *CPMMX* at detecting CPMM bugs? ←

RQ3. How significant are the techniques applied to *CPMMX*?

RQ4. How effective is *CPMMX* at detecting undiscovered CPMM bugs in the real world? ←

Experimental Setup

Baselines

Tool	Multi-Contract Support	Methodology
ItyFuzz	✓	Fuzzing
Echidna	✓	Fuzzing
DeFiTainter*	✓	Taint analysis
Mythril	✗	Symbolic execution
Slither	✗	Static analysis

*DeFiTainter targets price manipulation bugs while other tools provide more general/configurable oracles

Experimental Setup

Datasets

1. **DeFiHackLabs**

   23 CPMM bugs manually selected from DeFiHackLabs

2. **BlockSec**

   124 CPMM bugs reported by BlockSec (Invariant 1 violations)

3. **RealWorld-BSC**

   122* vulnerable tokens (BlockSec) + 122 benign tokens

   *Excluded two tokens on Ethereum

RQ1. **Effectiveness** in Detecting CPMM Bugs

Table 1. CPMM composability bug detection rate of *CPMMX* and baselines on the DeFiHackLabs and BlockSec datasets.

	DeFiTainter	Echidna	ItyFuzz	Mythril	Slither	Ours
DeFiHackLabs Total	1/19	0/23	8.33/23	0/23	0/19	**21/23**
DeFiHackLabs Recall	0.05	0.00	0.36	0.00	0.00	**0.91**
BlockSec Total	1/123	9/124	74/124	0/124	0/123	**109/124**
BlockSec Recall	0.01	0.07	0.60	0.00	0.00	**0.88**

- Outperformed baselines in recall by **2.5x** for DeFiHackLabs dataset by **1.5x** for BlockSec dataset

RQ2. **Efficiency** in Detecting CPMM Bugs

- Achieved **top F1 score of 0.97** with zero false positives
- Detected **most bugs in the least time** at only ~14% of the next best time – thanks to **fewest number of timeouts**

→ Highlights the scalability benefits of a bug-specific detection tool

Table 2. Performance metrics and running times of CPMMX and baseline on RealWorld-BSC

	Echidna	ItyFuzz	Ours
Precision	0.80	**1.00**	**1.00**
Recall	0.09	0.49	**0.93**
F1 Score	0.16	0.66	**0.97**
Vulnerable Time (min)	2290	1707	**150**
Benign Time (min)	2425	2440	**447**
Overall Time (min)	4715	4147	**597**
Vulnerable Timeout #	111.33	62	**6**
Benign Timeout #	119.33	122	**16**
Overall Timeout #	230.66	184	**22**

*Ran with 20 minute timeout

RQ4. Detecting CPMM Bugs in the Real World

- Deployed *CPMMX* on Ethereum and Binance Smart Chain
  - Analyzed all UniswapV2 and PancakeSwap contracts
  - As of Oct 2024: ~371K (ETH) / ~1.7M (BSC) contracts
  - After coin-balance filtering: ~19K / ~29K tested via *shallow-then-deep search*
  - Large scale evaluation enabled by **early termination**
- Discovered **26 new bugs** that can extract **$15.7K** worth of stablecoins
  - Top 3 potential losses: $4,796 / $4,359 / $3,576
  - Average loss: $603
- Tried to contact token maintainers but got no response

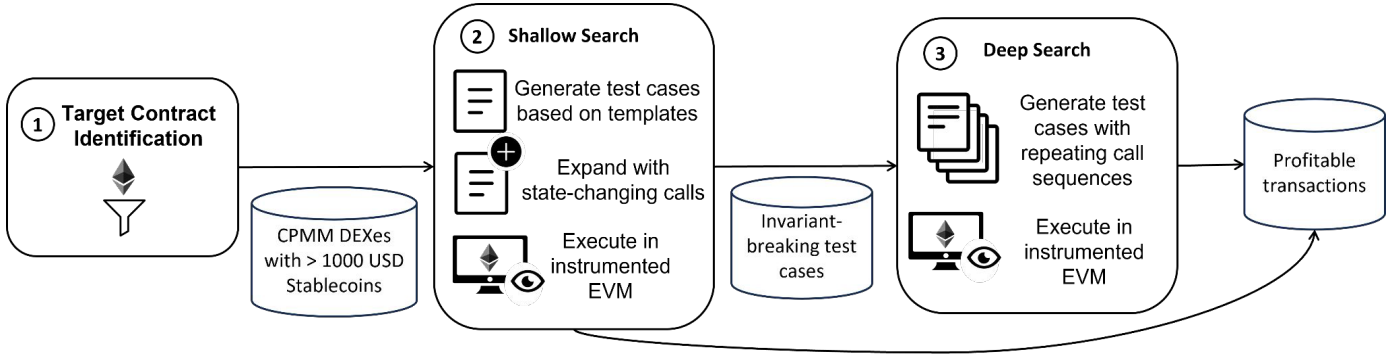
Discussion

- Lack of a safe and ethical way to report vulnerabilities in smart contracts if maintainers are uncooperative
  - Public disclosure can cause direct financial harm to token holders

- Efficiency vs. generalizability tradeoff
  - *CPMMX* is effective but does not detect other vulnerabilities
  - *CPMMX* also misses some CPMM bugs that do not conform to its templates

Summary

- Formalized **CPMM composability bugs** and identified two safety invariants
- Propose ***CPMMX*** that can automatically detect CPMM composability bugs across entire blockchains without false positives
- ***CPMMX*** identified **26** previously undiscovered vulnerabilities with total loss **$15.7K**



Thank you! Please reach out if you have any questions sujinhan@kaist.ac.kr
Code available at https://github.com/kaist-hacking/CPMMX