black hat USA 2024

AUGUST 7-8, 2024

BRIEFINGS

Use Your Spell Against You: A Proactive Threat Prevention of Smart Contract Exploit

Yajin Zhou

BlockSec & Zhejiang University

This work is a team effort of researchers from Zhejiang University and BlockSec.

Hailin Wang, Jianfeng Zhu, Hang Feng, Youwen Hu, Runhuai Li, Sheng Yu, Lei Wu, Yajin Zhou

About Me

- Co-founder of BlockSec and Professor of Zhejiang University
- Research interests
  - DeFi security, Blockchain system security
- Publish: 60+ papers with 9,000+ citations
- Hack and build systems
- Read more: https://yajin.org

Security Matters in Web3

Despite the bull and bear cycles in the crypto market, losses caused by exploits and scams have been growing at a rapid pace.

Security Matters in Web3

Explorer > Security Incidents

https://app.blocksec.com/explorer/security-incidents

Security Incidents

The attack incidents causing losses exceeding $100K will be documented.

Search For Project Name or Tx Hash of Attack Tx

All | Bookmark

Project	Loss	Chain	Vulnerability	Date		
Ronin Bridge	~ $11.8 M		Misconfiguration	2024/08/06	Root Cause	Share
TokenStake (Unknown)	~ $578 K		Vulnerable Price Dependency	2024/08/05	Root Cause	Share
Convergence	~ $210 K		Unverified User Input	2024/08/01	Root Cause	Share
Spectra	~ $550 K		Arbitrary Call	2024/07/23	Root Cause	Share
DeltaPrime	~ $1 M		Misconfiguration	2024/07/22	Root Cause	Share
UPS	~ $521 K		Business Logic Flaw	2024/07/21	Root Cause	Share
WazirX	~ $207 M		Compromised Private Key	2024/07/18	Root Cause	Share
LI.FI	~ $11 M		Arbitrary Call	2024/07/16	Root Cause	Share
Minterest Finance	~ $1.5 M		Reentrancy	2024/07/14	Root Cause	Share

DeFi Security Incidents Dashboard

Why Security Incidents are Prevalent

- Economical incentive
  - Hackers can get “paid”. Think about a house full of gold but without a good security system
- Less security-qualified developers
  - Developers are not trained well in security concepts
- DeFi composability: creates more attack vectors

Why Security Incidents are Prevalent

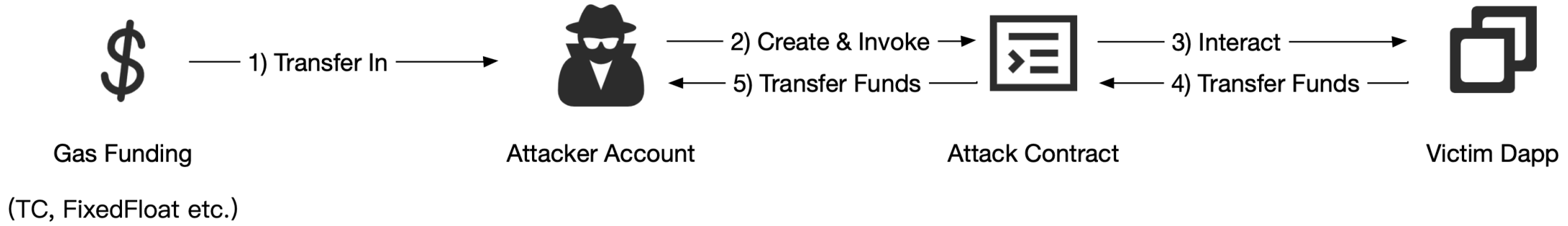
- Openness: everyone can see the code on the chain, and everyone can issue an attack tx if a vulnerability exists
- Anonymity: it’s hard to trace (not impossible) the attacker if he/she is smart enough to hide
- Flashloan: enlarge the money that an attacker can use

Existing Approach

- Pre-launch
  - Code auditing
  - Fuzzing testing
  - Formal verification
- Post-launch
  - We think there should be an effective approach to detect and **block** hacks after the protocol is deployed

Steps of a DeFi Attack

- Prepare the attack
- **Launch the attack**
- Launder the profit

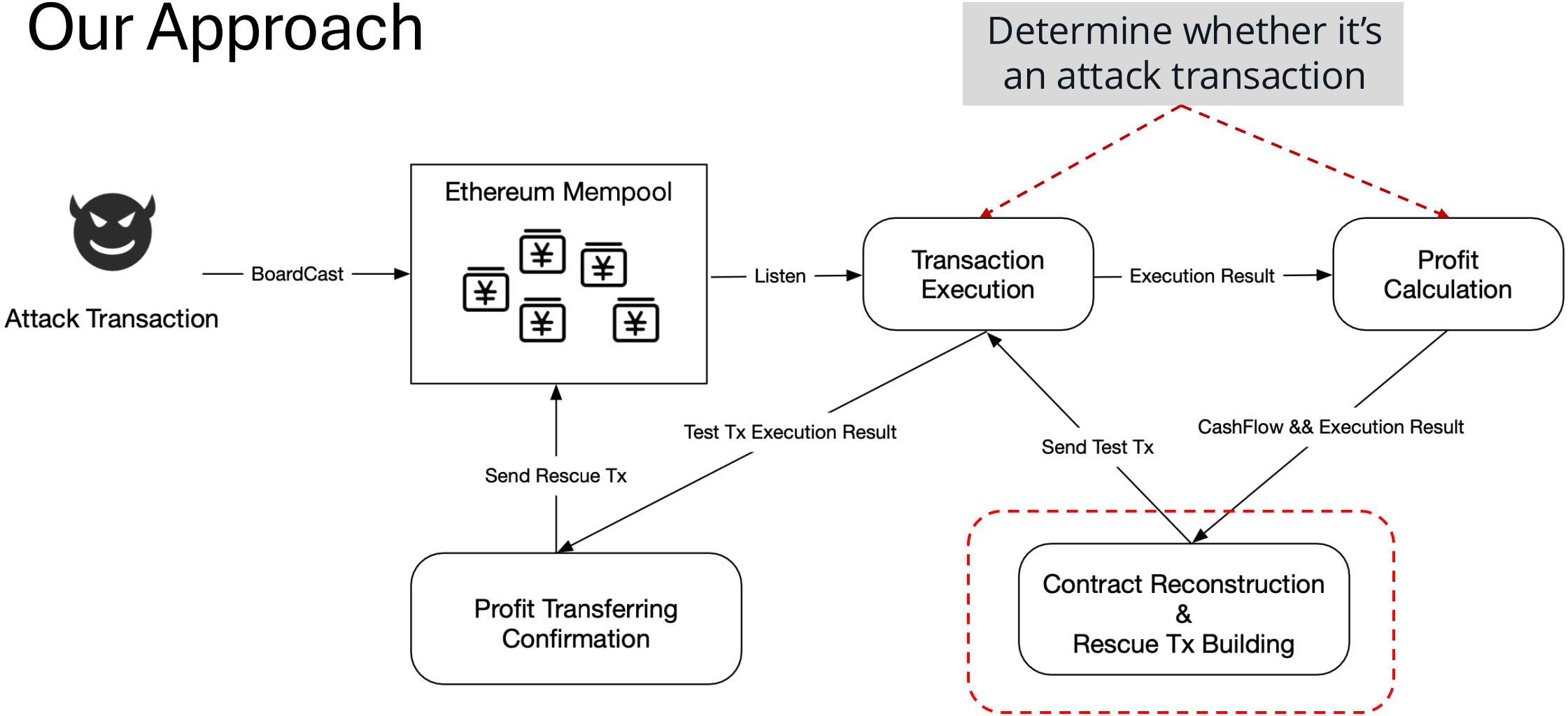


Our Approach

- The process of a transaction is confirmed
  - T1: A transaction is broadcasted to the p2p network
  - T2: Every node on the network can listen to pending transactions
  - T3: Validators confirm the transaction

Can we **detect and block** the attack transaction during the time window of T1 and T3 (typically 12 seconds in Ethereum)

Our Approach

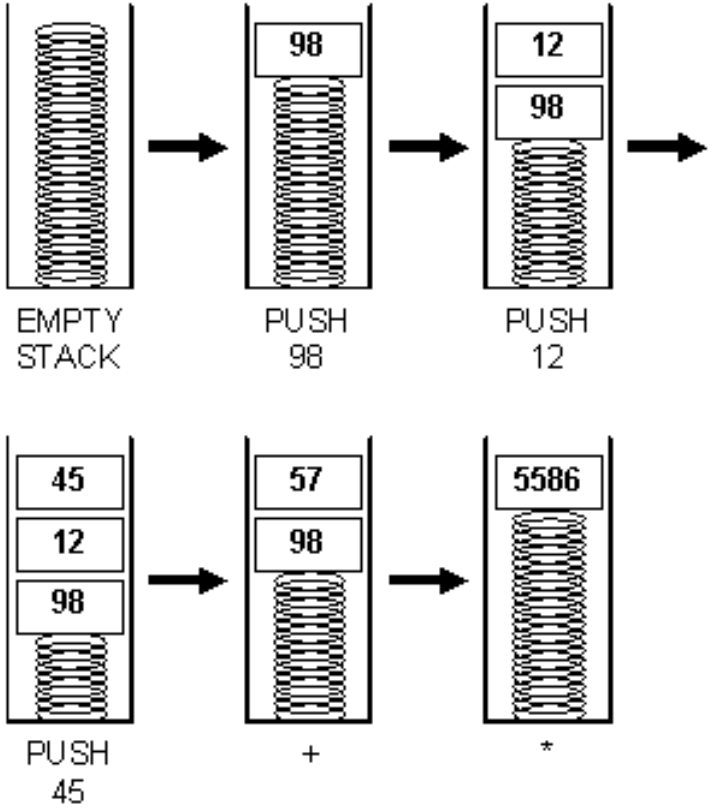


Background: Smart Contracts

- Can be called through **transactions**
- Contain code (bytecode)
- Are executed in EVM

Background: EVM

- Stack-based VM



https://www.preethikasireddy.com/post/how-does-ethereum-work-anyway

Challenges and Solutions

- How to extract attack logic from the attack contract
  - The attack contract contains verification logic (of the caller) – in a simple way or in an obfuscated way
  - Callback functions
- Solutions: freezing conditional JUMP
  - Leverage execution trace

Challenges and Solutions

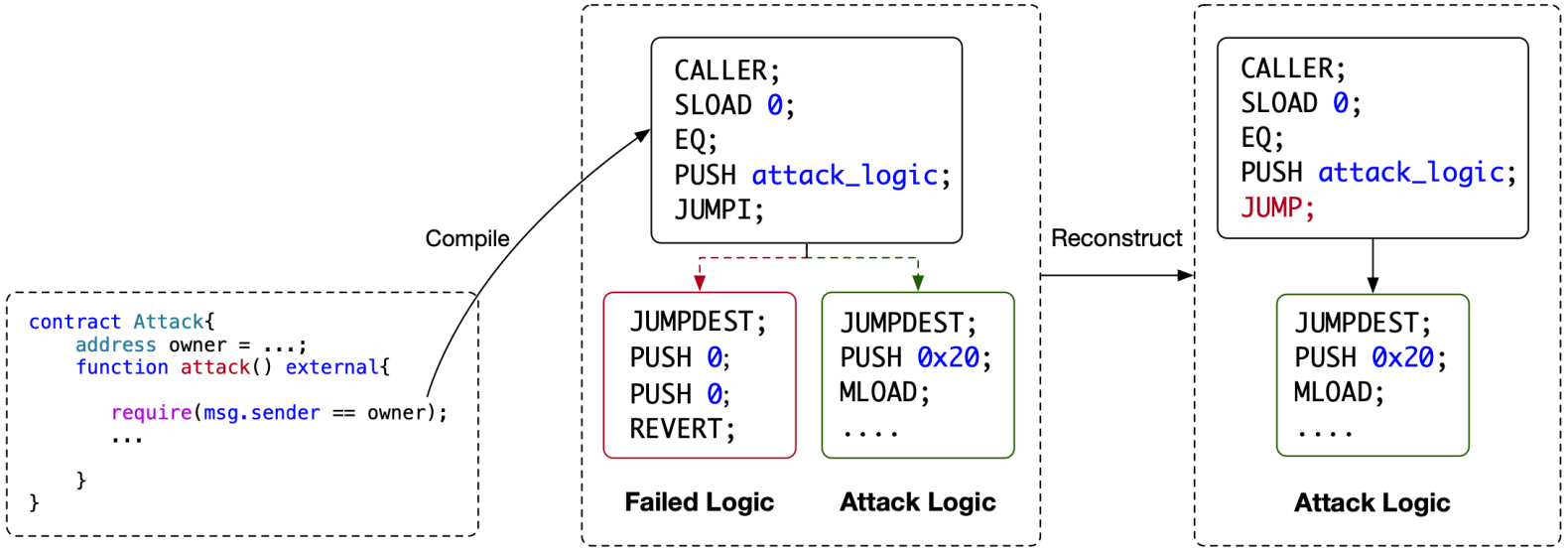
- How to locate and replace revenue addresses
  - Revenue addresses are the ones that get profit during the attack
  - We need to replace them with our ones
- Solutions: balance change table
  - Build the balance change table
  - Replace the addresses when they are pushed on the stack

Challenges and Solutions

- How to identify the pre-conditions of an attack
  - Auxiliary contracts
  - Multiple transactions to launch an attack
- Solutions
  - Dependency analysis if multiple transactions/contracts are involved

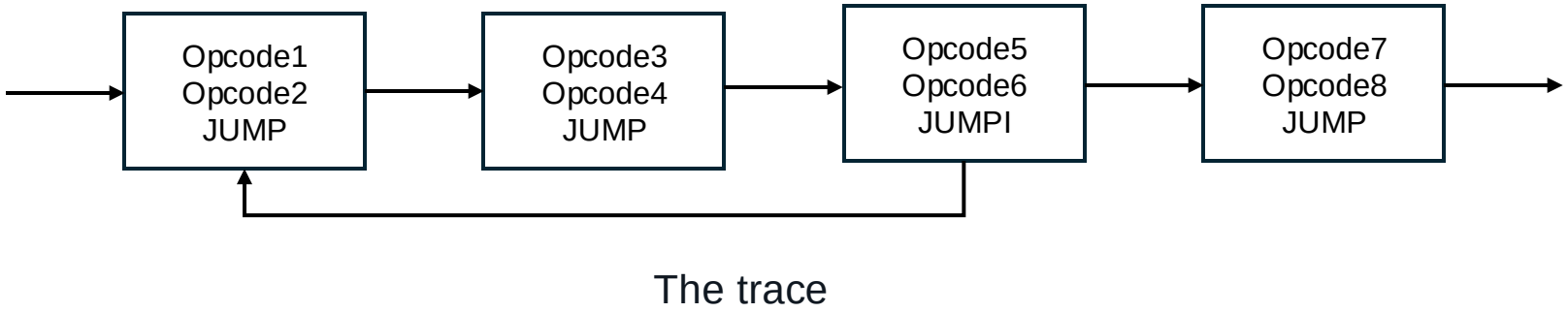
Task One: Extract and Construct Attack Logic

- Automatically reconstruct a smart contract with the attack logic
  - Attacks usually use some logic to protect their attack contacts



Task One: Extract and Construct Attack Logic

- Option one: connect the opcode trace of an attack transaction
  - Input: opcode trace of an attack transaction
  - Output: a new smart contract with the trace



Generated contract

Opcode1
Opcode2
Opcode3
Opcode4
Opcode5
Opcode6
Opcode1
Opcode2
Opcode3
Opcode4
Opcode5
Opcode6
Opcode7
Opcode8

Task One: Extract and Construct Attack Logic

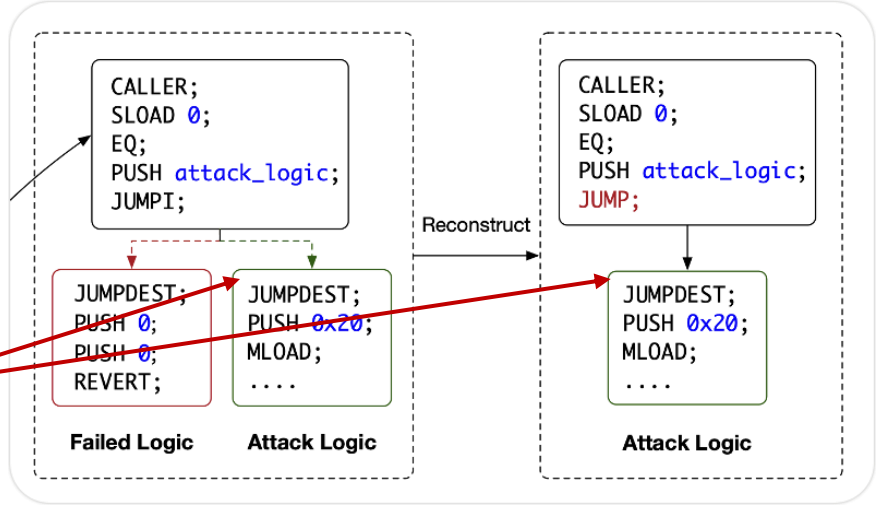
- Option one: connect the opcode trace of an attack transaction
  - It works theoretically but does not work in practice
  - Contract size is limited – it will generate a large contract if the attack has a loop

Opcode1
Opcode2
Opcode3
Opcode4
Opcode5
Opcode6
Opcode1
Opcode2
Opcode3
Opcode4
Opcode5
Opcode6
Opcode7
Opcode8

Task One: Extract and Construct Attack Logic

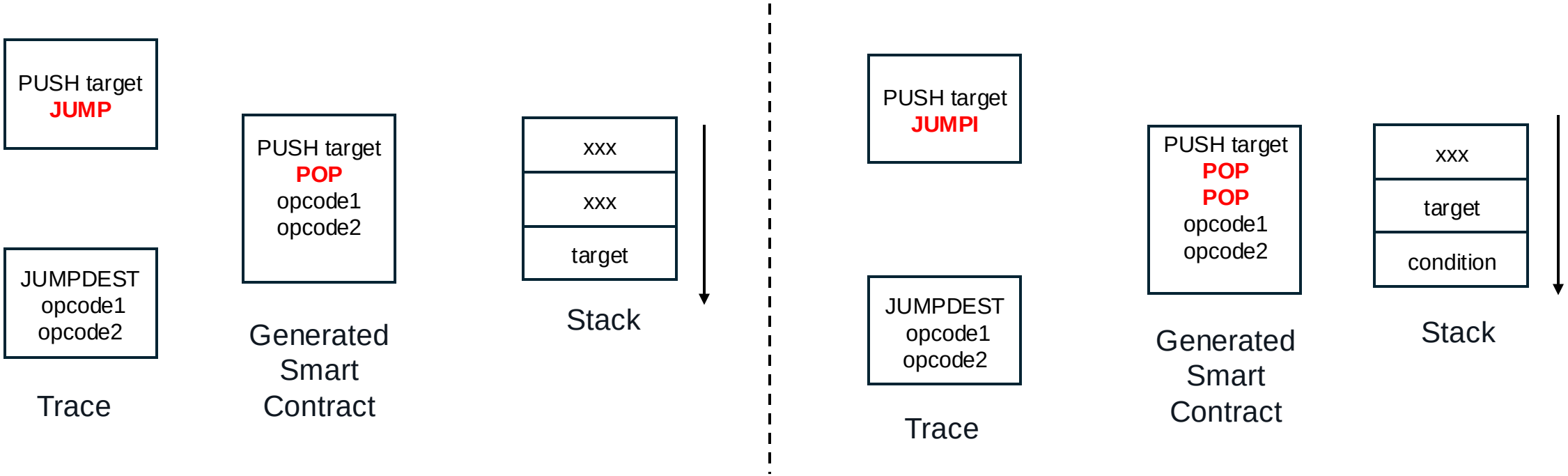
- Option two: reuse the basic block as much as possible
  - It works theoretically and in practice
  - But the process is more complicated than the previous one
    - We need to handle the JUMP instruction and fixup the offsets in the trace and our generated contract

Offsets are different



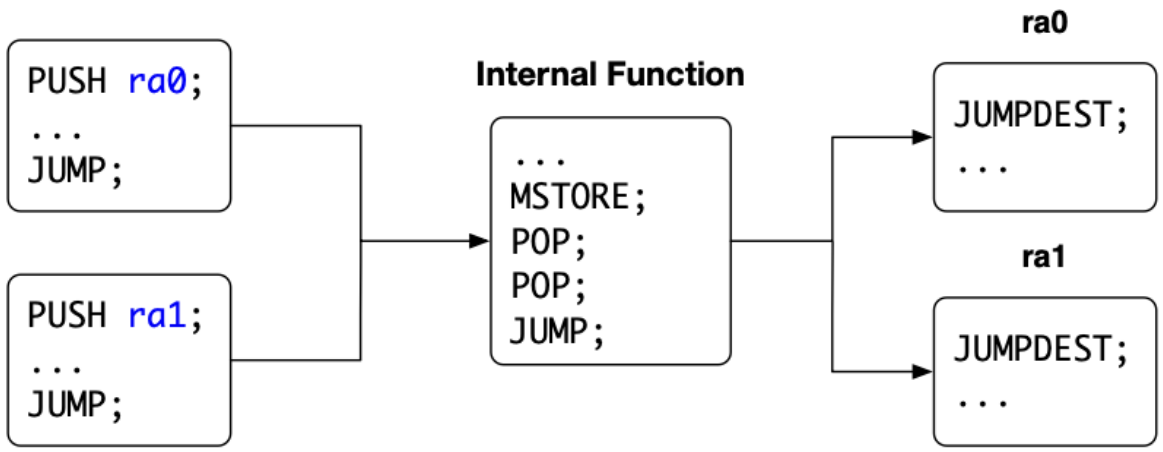
Freezing Conditional JUMP

- For simple JUMP and JUMPI -> replace with POP



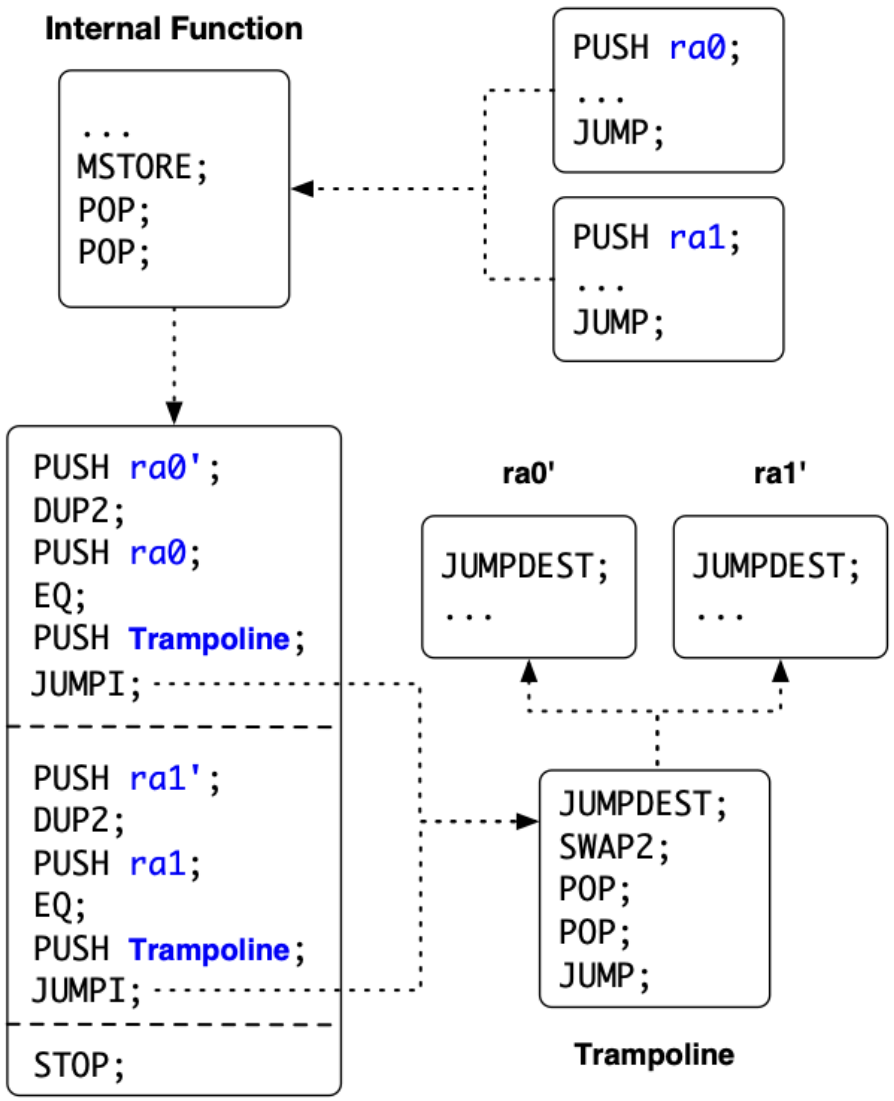
Freezing Conditional JUMP

- For JUMP with multiple targets
  - The real target is determined in the runtime -> from which basic block



Freezing Conditional JUMP

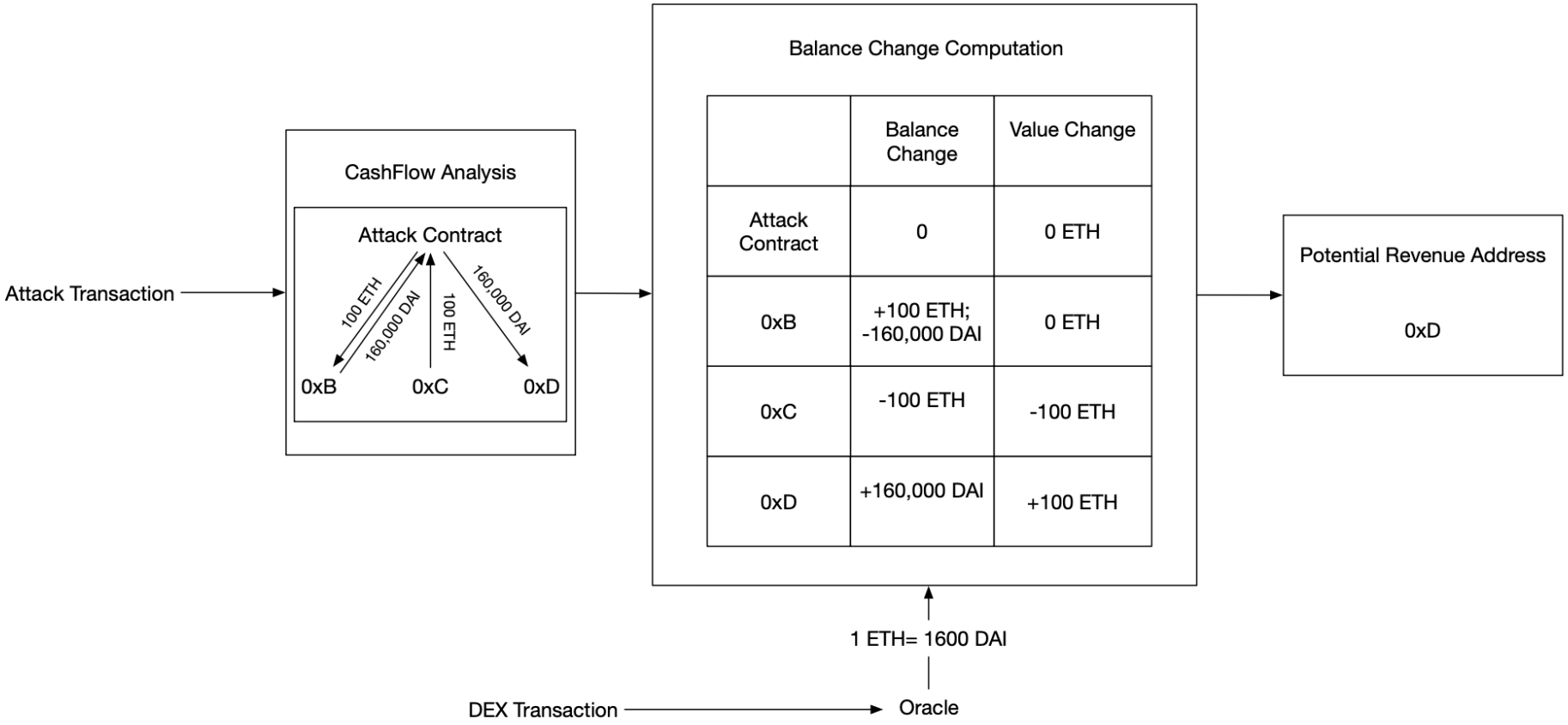
- We compare the basic block and use a trampoline to relocate the destination block
  - **ra0 -> ra0'**
  - **ra1 -> ra1'**
- Also, we need to make the stack balance after the trampoline



Task Two: Identify and Replace Revenue Addresses

- Revenue addresses
  - The ones that balance changes positively, *and replacing them does not affect the attack logic*
- How to identify them: try-and-catch
  - Calculate the balance changes
  - Replace the ones with positive balance change to see what happens
- When: Replace them when they are pushed on the stack

Task Two: Identify and Replace Revenue Addresses

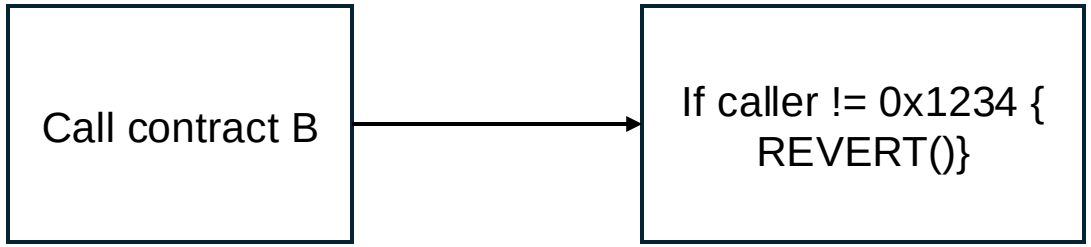


Task Three: Identify Pre-conditions

- Auxiliary contracts
  - Contracts created by the attacker before or during the attack process – they also contain the attack logic
- Multiple transactions
  - An attack can consist of multiple transactions: T1, T2 prepare the attack and T3 launches the attack

Identify Auxiliary contracts

- Created during the attack: find CREATE/CREATE2 instruction
- Created before the attack: use REVERT to locate them
  - The auxiliary will revert during the interaction (since we change the caller when invoking the auxiliary contract)
  - We track all the reverted internal calls to identify the auxiliary contract



Auxiliary contract

Identify Multiple Transactions

- Attackers can split the attack transaction into multiple ones

EGD Finance Attack Incident

Identify Multiple Transactions

- Our method
  - We retrieve the state dependency search method to locate all preparation transactions
  - State-dependent: a transaction is using a storage changed by a previous transaction
  - Optimizations: cache state change called by a tx in (24 hours)

Evaluation

- Effectiveness
  - If our system can successfully synthesize the attack tx and contract with replaced revenue addresses, then our system is effective in blocking that hack
- Dataset
  - Historical attacks in DeFiHackLabs: June 2020 to Februray 2023
  - Real attacks

Evaluation

- Dataset: 117 attacks in total, we use 87 of them
  - The attack crosses different blocks
  - They require some special tokens
- Among 87 attacks, we can successfully generate the attack contracts

Feasibility	Category	Success	Total
Not Feasible	Unable to finish in a single block	0	29
	Require special capital	0	4
Feasible	Common attack	47	50
	Involve anti-front-running strategy	31	34

Evaluation

- Reasons for failed cases
  - Attack tx was too complicated with many auxiliary contracts – the generated contract was too big (bigger than the maximum allowed contract size)

Real Cases

- Blocked hacks and rescued more than 20 million USDs
  - Representative ones: ParaSpace: 5 Million, Saddle Finance: 3.8 million

Saddle @saddlefinance · Apr 30, 2022
The team is investigating a possible exploit and is pausing pool withdrawals
14 37 95

Saddle @saddlefinance · Apr 30, 2022
Correction: Only metapools are paused. Single-asset withdrawals are currently restricted, but balanced pool withdrawals are always possible
1 11

Saddle
@saddlefinance

White hat hackers @BlockSecTeam were able to secure $3.8m. The team is in contact with them to return the funds

6:42 PM · Apr 30, 2022

Parallel Network
@ParallelFi

We alongside @BlockSecTeam have identified the cause of the exploit that occurred earlier on the ParaSpace protocol, and we are relieved to share that all user funds and assets on ParaSpace are safe and secure. No NFTs were compromised and financial losses to the protocol are minimal.

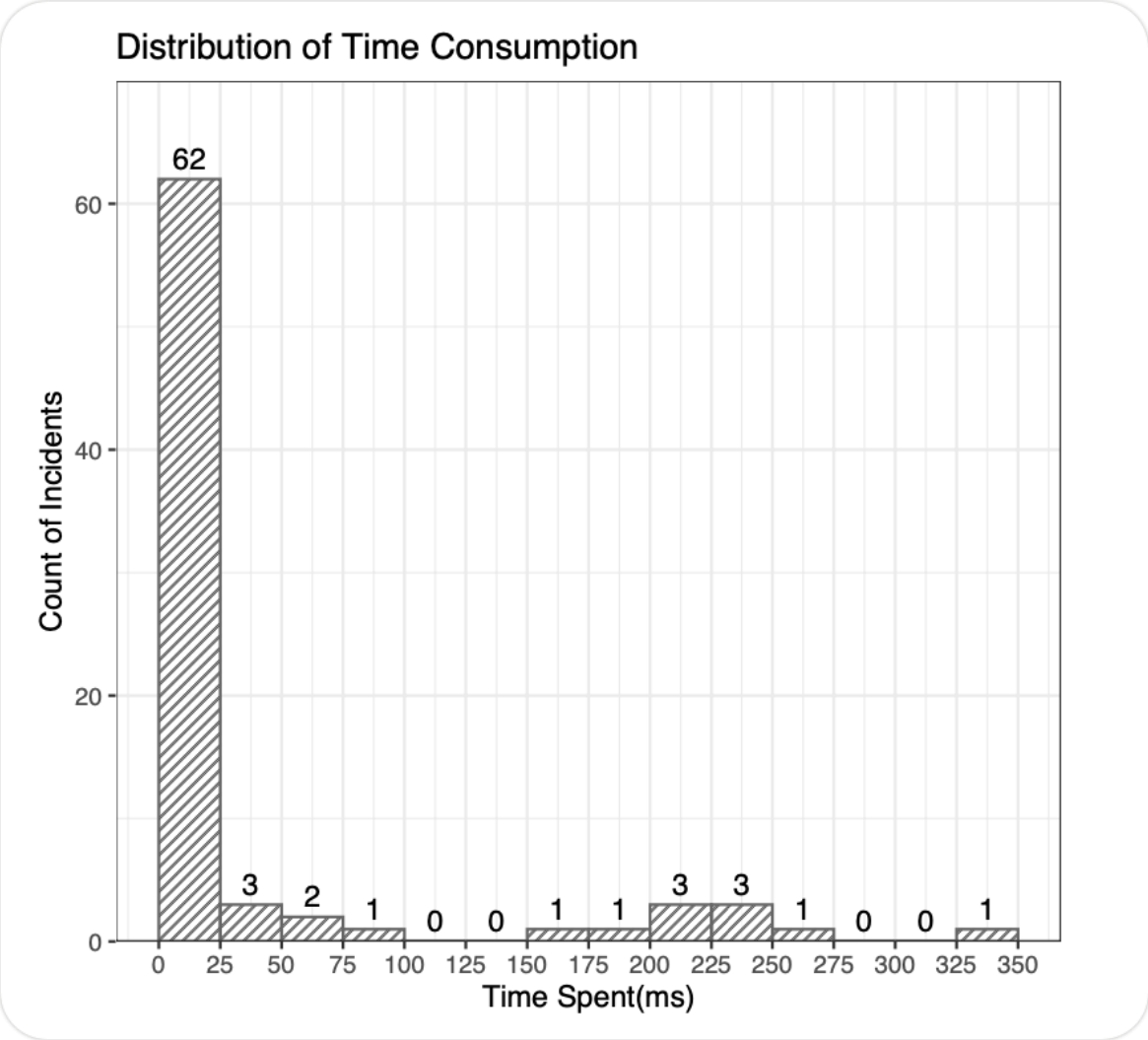
We thank you for your patience while we calculated the financial impact of the exploit.

Thanks to @BlockSecTeam's technology and the swift actions taken by the @BlockSecTeam team, losses from the protocol are minimal and we were able to rescue the 2,909 ETH that the exploiter was attempting to withdraw from the protocol.

https://blocksec.com/blog/lead-in-phalcon-s-hack-blocking-saga

Evaluation

- Efficiency
  - How quick to reconstruct the attack contract
    - More than 80% cases are finished in less than 25 ms



Take Away Messages

Phalcon

A Platform to Monitor and Block Hacks

Help users, protocol operators, traders, and everyone to perceive suspicious transactions, get instant alerts and take automatic actions

- DeFi hack is still a serious threat
- We propose a post-launch security measure to detect and block hacks
  - It automatically synthesizes a similar attack tx, but with replaced revenue addresses in synthesized contracts
- We have rescued more than 20 million USDs
- The technique has been commercialized in Phalcon
  - https://blocksec.com/phalcon