

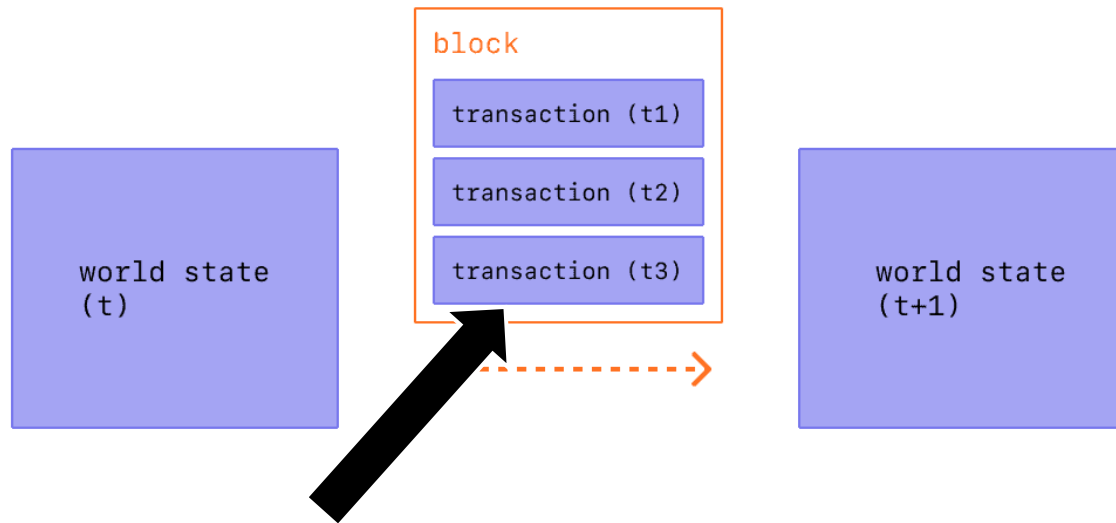
The Blockchain Imitation Game

Kaihua Qin , Stefanos Chaliasos , Liyi Zhou ,
Benjamin Livshits , Dawn Song, Arthur Gervais

2023 USENIX SECURITY SYMPOSIUM

Background

- **Smart Contract:** implement a set of rules for managing digital assets in Ethereum accounts.



$$\text{Transaction fee} = \text{Gas Price} * \text{Gas}$$

Transaction Hash: 0x6b72d7b8da5aa4120ec6104902df54690092a0c77d3081dfc8fb875070fa3317

Status: Success

Block: 17574324 | 3 Block Confirmations

Timestamp: 42 secs ago (Jun-28-2023 12:43:23 AM +UTC) | Confirmed within 1 sec

Sponsored: **CHANCER**
Don't Miss the Presale [Buy Now](#)

From: 0x90aaB4fd14708047b44A8D9A1fC795dA99eb95D1

To: 0x618da4ec3ED58736fE05CCF87C63fb955Db953cd

Value: 405 wei (< \$0.000001)

Transaction Fee: 0.000279103703592 ETH (\$0.52)

Gas Price: 13.290652552 Gwei (0.000000013290652552 ETH)

Gas Limit & Usage by Txn: 22,050 | 21,000 (95.24%)

Gas Fees: Base: 13.290652551 Gwei | Max: 15.943770075 Gwei | Max Priority: 0.000000001 Gwei

Burnt & Txn Savings Fees: Burnt: 0.000279103703571 ETH (\$0.52) Txn Savings: 0.000055715467983 ETH (\$0.10)

Background

Taint Analysis

tracks information flow originating from taint sources (e.g., untrusted input) as a program executes.

- Taint Introduction
- Taint Propagation
- Taint Checking

```
1. [...]  
2. scanf("%d", &x);  
3. [...]  
4. y = x + k;  
5. [...]  
6. x = 0;  
7. [...]  
8. while (i < y)
```

```
1. if (x > 0)  
2.     y = 1;  
3. else  
4.     y = 0;
```

Background

Imitation attack

Naïve string-replace imitation method

This algorithm takes as input a victim transaction, and simply replaces the transaction's sender address with an adversarial address in the transaction sender and data fields.

```
1  contract GANGSINU {  
2    function increaseAllowance(  
3      address spender,  
4      uint256 addedValue  
5    ) public virtual returns (bool) {  
6      _approve(  
7        _msgSender(),  
8        spender,  
9        _allowances[_msgSender()][spender] +  
10         addedValue  
11      );  
12      _mint(spender, addedValue);  
13      return true;  
14    }  
}
```


```
1  contract CustomizedLiquidationContract {  
2    function printMoney(...) public payable {  
3      require(0x53d8...0d81 == msg.sender);  
4      //liquidation logic omitted  
5    }  
6  }
```

Transaction Hash: 0xe58214cfb38650089ce6bace5669a58e03557935ab8480467ae511df69ca40db

Status: Success

Block: 12978074 4591624 Block Confirmations

Timestamp: 688 days 20 hrs ago (Aug-07-2021 01:02:04 PM +UTC)

Sponsored: 

From: 0x5B1B0349B3a668c75cc868801A39430684e3f36A

Interacted With (To): 0x9796Bcece6b6032deB6f097b6F1cc180aE974feC

ERC-20 Tokens Transferred: From Null: 0x000...000 To 0x5B1B03...84e3f36A For 1,000,000,000,000,000
Gangster Inu... (GANGSI...)

Value: 0 ETH (\$0.00)

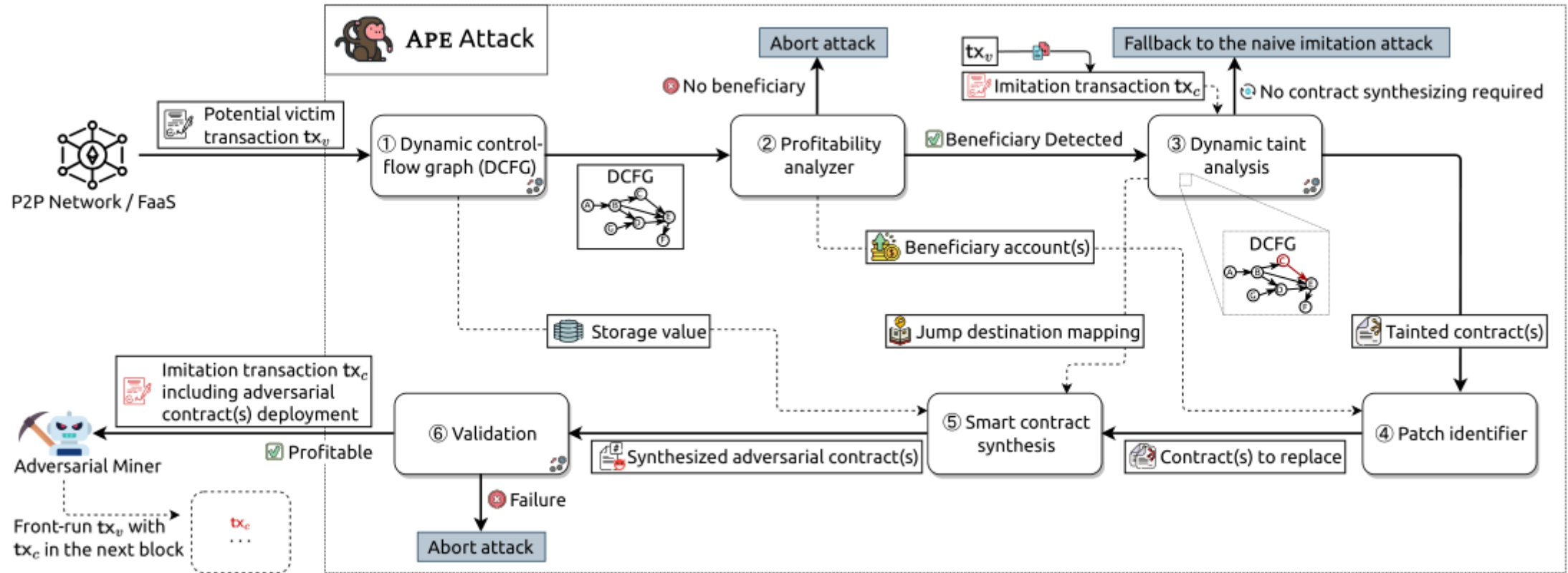
Transaction Fee: 0.004608912663122914 ETH \$8.64

Gas Price: 107.491491082 Gwei (0.000000107491491082 ETH)

Problem & Challenge

- ✓ short front-running time-window(real-time)
- ✓ recursively identify and replace the victim contract
- ✓ synthesized contracts are invoked and executed correctly

APE : a generalized imitation tool



1、 Dynamic Control-Flow Graph

~~CFG(static information)~~

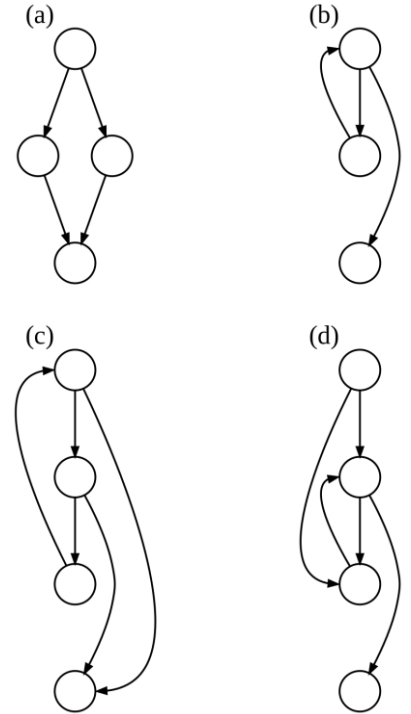
DCFG(dynamic information)



execute tx locally

record **condition value of JUMPI**(for 3)

identify contract calls and track the executions of all smart contract



2、 Profitability Analyzer

extract the asset transfers from the DCFG constructed through analyzing the EVM logs defined in asset implementation standards

Beneficiary Account

- If the sender is a beneficiary account, other accounts are irrelevant to the profitability analyzer.
- Otherwise, if the sender is not a beneficiary account, the collective profit of other beneficiary accounts, minus the potential loss of the sender account must remain positive.

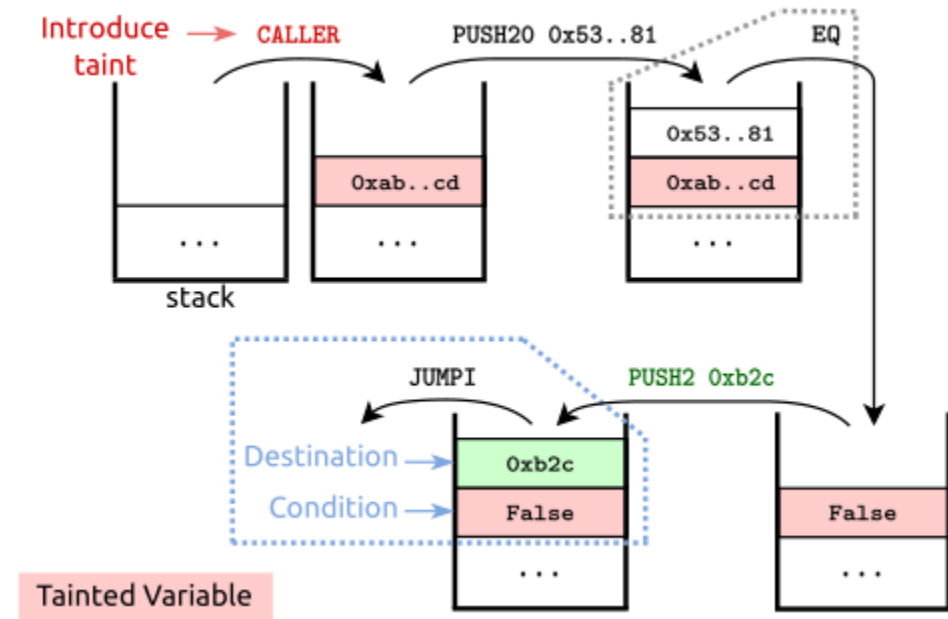
3、 Dynamic Taint Analysis

track where and how tx_c 's execution fails

```
1 contract CustomizedLiquidationContract {  
2   function printMoney(...) public payable {  
3     require(0x53d8...0d81 == msg.sender);  
4     //liquidation logic omitted  
5   }  
6 }
```

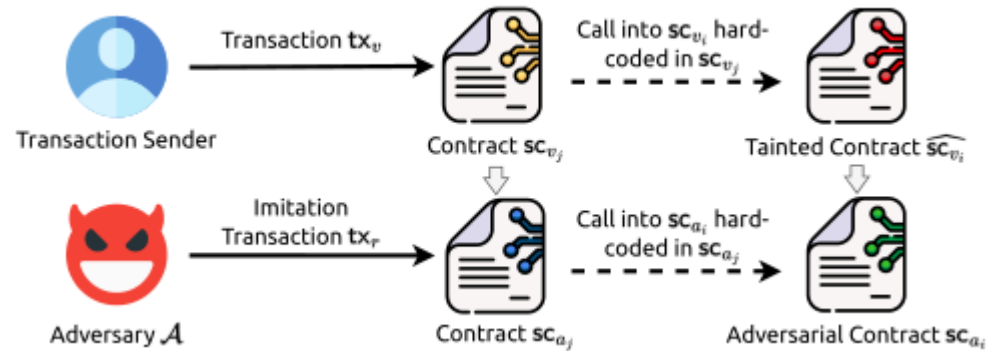
PC: Disassembled Code

0xb0c:	JUMPDEST	
0xb0d:	CALLER	
0xb0e:	PUSH20	0x53d8...0d81
0xb23:	EQ	
0xb24:	PUSH2	0xb2c
0xb27:	JUMPI	
0xb28:	PUSH1	0x0
0xb2a:	DUP1	
0xb2b:	REVERT	



4、 Patch Identifier

- Invocation from a transaction
- Invocation from a contract



5、 Smart Contract Synthesis

tainted contract: **replaces JUMPI with JUMP**

modify **hard-coded account**(contract)



modify contract bytecode

6、 Validation

deploys every sc_{ai} and executes tx_c on the latest
blockchain state locally

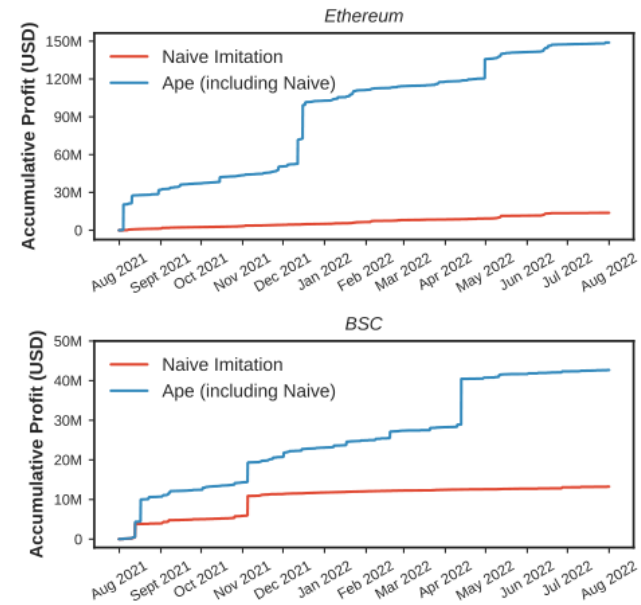


The revenue **covers all transaction fees
including the smart contract(s) deployment fees.**

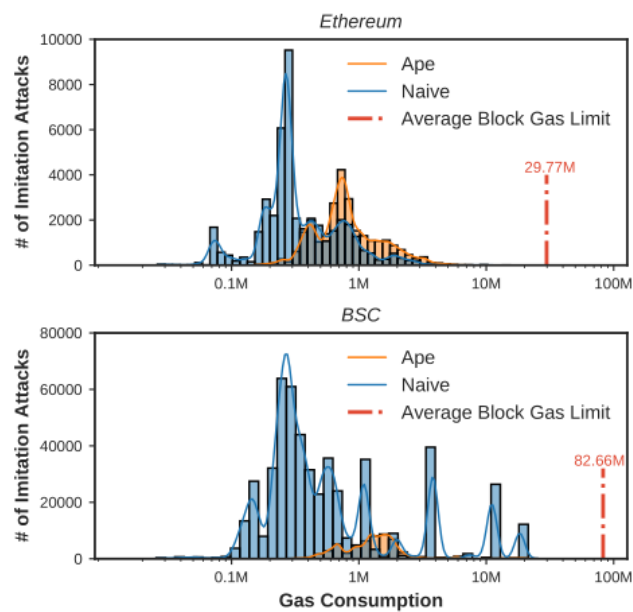
Overall Evaluation Results

Chain	Attack	Transactions	Contracts	Overall Profit (USD)	Average Profit (USD)
Ethereum	Naive	43,979	NA	13.87M	315.48 ± 4.73K
	APE	26,127	665	135.08M	5.17K ± 227.22K
BSC	Naive	516,128	NA	13.25M	25.67 ± 1.78K
	APE	52,799	1,193	29.45M	557.75 ± 55.88K

Overall attack statistics



Attack profit



Gas consumption

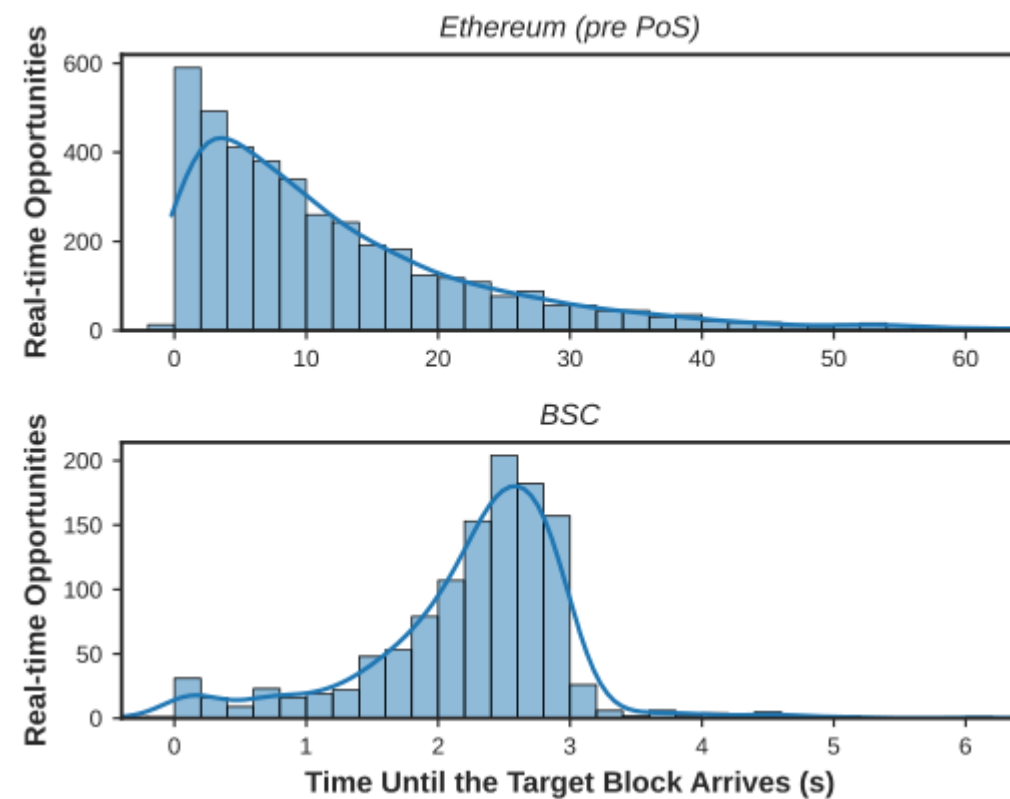
		Mean	Std.	Max	Min
Ethereum	Adversarial Contract Number	1.02	0.15	3	1
	Contract Size Reduction	60.16%	19.19%	98.63%	-295.56%
BSC	Adversarial Contract Number	1.05	0.23	3	1
	Contract Size Reduction	57.59%	18.69%	99.46%	-613.33%

Adversarial contract

APE Real-Time Performance

	Mean (s)	Std. (s)	Max (s)	Min (s)
Step ① DCFG	0.02	0.03	0.36	3×10^{-4}
Step ② Profitability Analyzer	2×10^{-3}	5×10^{-3}	0.10	2×10^{-5}
Step ③ Dynamic Taint Analysis	0.04	0.06	1.39	3×10^{-4}
Step ④ Patch Identifier	2×10^{-5}	5×10^{-5}	2×10^{-3}	1×10^{-6}
Step ⑤ Smart Contract Synthesis	5×10^{-4}	2×10^{-3}	0.09	2×10^{-5}
Step ⑥ Validation	7×10^{-3}	0.02	0.96	2×10^{-4}
Overall time cost of APE	0.07	0.10	1.59	9×10^{-4}
Overall time cost Naive Imitation	0.01	0.01	0.11	2×10^{-4}

Single transaction performance



Mempool performance

APE Countermeasures

- Imitation as a Defence Tool (Whitehat hacking)
- Breaking Atomicity
- Front-running Mitigation
- Code Obfuscation

Conclusion

- generalized blockchain imitation game
- APE: a generalized imitation tool for EVM-based blockchains
- the first to show that dynamic program analysis techniques can realize an imitation attack, posing a substantial threat to blockchain users