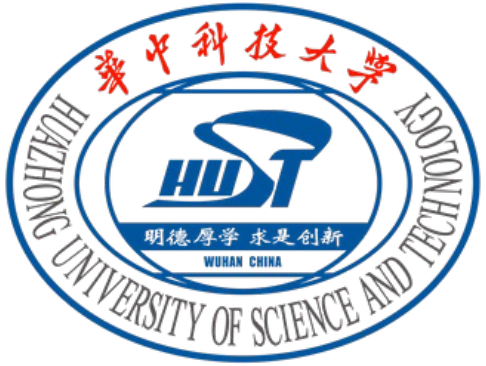
All Your Tokens are Belong to Us: Demystifying Address Verification Vulnerabilities in Solidity Smart Contracts

Tianle Sun[1], Ningyu He[2], Jiang Xiao[1], Yinliang Yue[3], Xiapu Luo[4]
Haoyu Wang[1]

[1]Huazhong university of Science and Technology
[2]Peking University
[3]Zhongguancun Laboratory
[4]The Hong Kong Polytechnic University



Motivating Example

Vulnerable Contract

```
function migrateStake(address oldStaking, uint256 amount) external {
    // no whitelist verification
    StaxLPStaking(oldStaking).migrateWithdraw(msg.sender, amount);
    _applyStake(msg.sender, amount);
}

function _applyStake(address _for, uint256 _amount) internal {
    _totalSupply += _amount;
    _balances[_for] += _amount;
}
```

//minting valuable tokens

Attacker Contract

```
contract Attacker {
    function migrateWithdraw(address from, uint256 amount) external {}

    function attack() {
      ...
    }
}
```

P1. The vulnerable function takes an address as a parameter, and performs insufficient authorization examination on that address

P2. The address in P1 is taken as the target of an external call.

P3. On-chain states that are control-flow dependent on the return value mentioned in P2 are updated.

1. Input attack contract.
2. Execute attack contract.
3. Change state.
4. Withdraw all tokens and take profit.

Background

- Blockchain has been hacked for large amounts of money every year since 2021

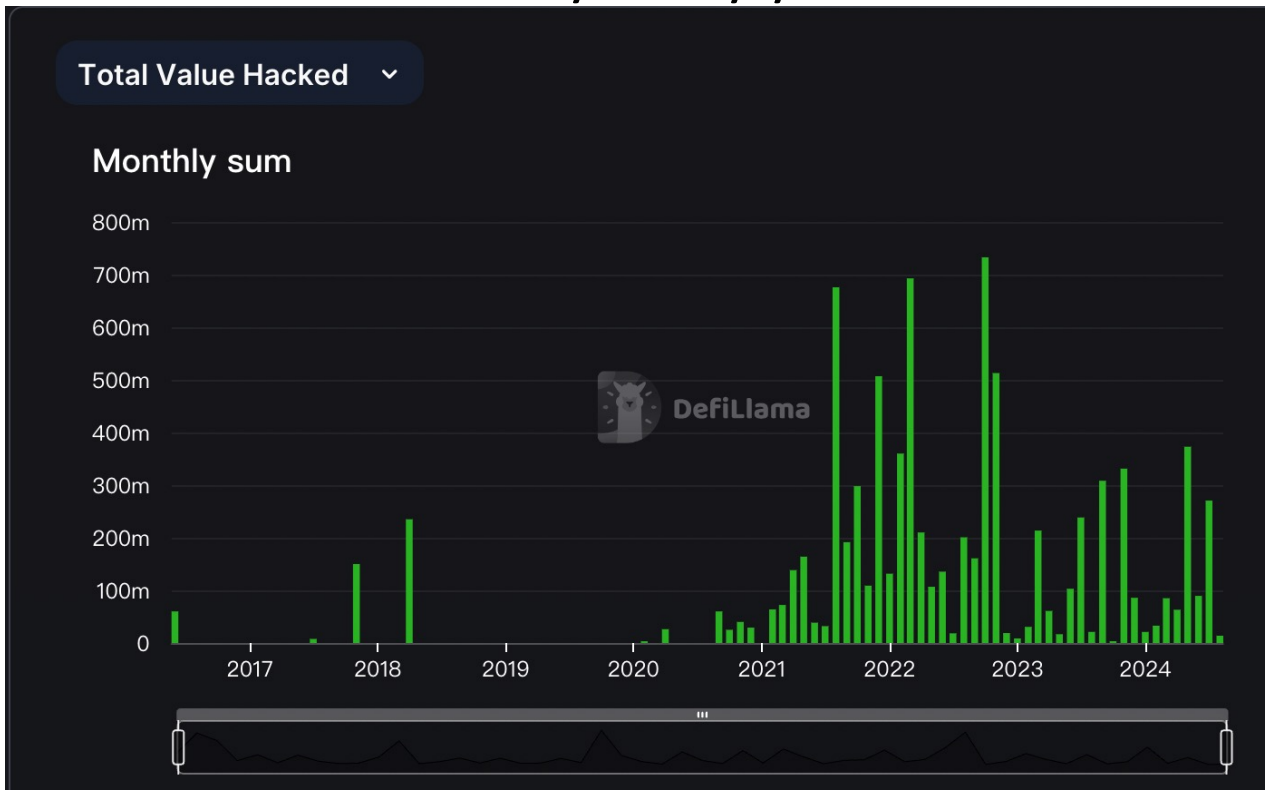


Fig 1. Total Value hacked from 2017 to 2024.

- Address verification is crucial for secure contract execution

$8.2M Lost as Visor Finance Suffers Latest DeFi Hack 2021

by Chris Williams Dec. 21, 2021

Multichain Hack Worsens as Loss of Funds Reaches $3M: Report 2022

Over $7 Million Drained from Exactly Protocol on Optimism Layer 2 Network

by BSC News August 18, 2023

2023

address verification as a major issue in security incidents

Common Address Verification Methods

List 1. Hard-encoded comparison

1	addr_1
2	…
3	…
…	…
n	…

List 2. Address enumeration

1	addr_1
2	addr_2
3	addr_3
4	addr_4
5	addr_5
…	…
n	addr_n

List 3. Mapping verification

sha3(add_1)	True
…	…
sha3(add_2)	True
…	…
…	…
sha3(add_n)	True
…	…

Challenges and Limitation

Challenges

- **Lack of semantics.**
- **Inter-procedural analysis on control flow and data flow.**

Limitations of Existing Tools

- **Pattern-based Matching** (May miss complex or novel vulnerability patterns)
- **Symbolic Execution & Model Checking** (Path explosion and performance bottlenecks)
- **Taint Analysis** (Not optimized for address verification vulnerabilities)

Design of AVVERIFIER

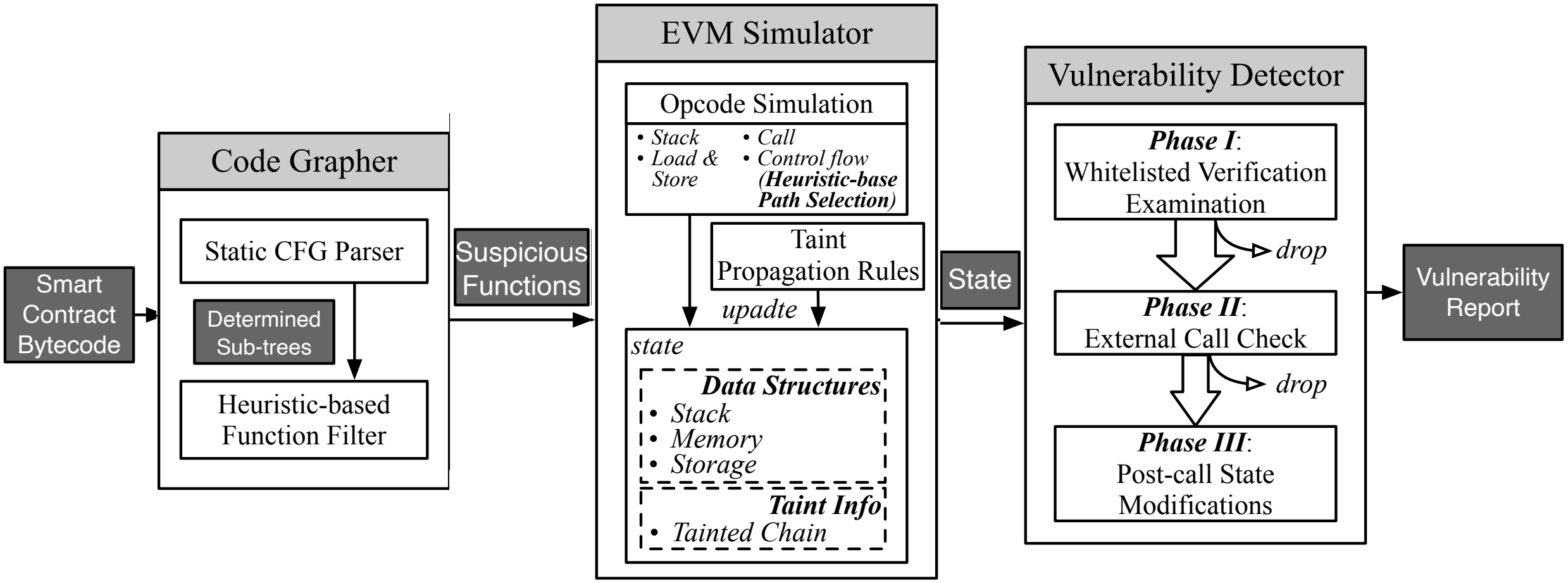


Fig 2. The workflow and architecture of AVVERIFIER.

Research Question

- RQ1: Is AVVERIFIER efficient and effective in identifying the address verification vulnerability?
- RQ2: How many smart contracts are vulnerable in the wild and what are their characteristics?
- RQ3: Can AVVERIFIER be deployed as a real-time detection system?

RQ 1: efficient and effective

Metrics	AVVERIFIER				Mythril*				Ethainter*				Jackal*				ETHBMC*			
	P	$\overline{P}$	N	$\overline{N}$	P	$\overline{P}$	N	$\overline{N}$	P	$\overline{P}$	N	$\overline{N}$	P	$\overline{P}$	N	$\overline{N}$	P	$\overline{P}$	N	$\overline{N}$
Avg. Time (s)	7.98	6.85	6.74	7.72	24.36	31.62	29.39	28.47	9.74	10.32	12.36	12.15	20.21	18.40	18.05	20.04	0.33	0.35	1.64	1.52
# Timeout	0	0	0	0	2	2	2	2	1	1	1	1	2	2	1	1	0	0	0	0
True Positives	6	-	-	4	2	-	-	1	4	-	-	3	4	-	-	3	0	-	-	0
True Negatives	-	6	4	-	-	4	2	-	-	5	3	-	-	4	3	-	-	6	4	-
False Positives	-	0	0	-	-	0	0	-	-	0	0	-	-	0	0	-	-	0	0	-
False Negatives	0	-	-	0	2	-	-	1	1	-	-	0	0	-	-	0	6	-	-	4
Precision	100%				100%				100%				100%				0%			
Recall	100%				50%				87.5%				100%				0%			

*The address vulnerability detector is implemented by ourselves.

Table 1. Performance comparison among Avverifier, Mythril, Ethainter, Jackal, and ETHBMC on the benchmark.

Avverifier performs best with well-constructed benchmark.

RQ 1: efficient and effective

Metrics	AVVERIFIER	Mythril*	Ethainter*	Jackal*	ETHBMC*
Avg. Time(s)	6.34	33.69	8.74	29.96	5.43
# Timeout	0	42	6	60	164
True Positives	348	16	147	172	2
True Negative	0	2	4	11	21
False Positives	21	8	17	10	0
False Negatives	0	301	195	116	182
Precision	94.3%	66.7%	89.6%	94.5%	100%
Recall	100%	5.1%	43.0%	59.7%	1.1%

*The address vulnerability detector is implemented by ourselves.

Table 2. Performance comparison among Avverifier, Mythril, Ethainter, Jackal, and ETHBMC on real-world contracts.

Avverifier performs best in a dataset of over 5 million contracts.

RQ 1: efficient and effective

- RQ1: Is AVVERIFIER efficient and effective in identifying the address verification vulnerability?

Answer:

- AVVERIFIER can improve the efficiency 2 to 5 times.
- AVVERIFIER can achieve at least 94% precision and 100% recall on the well-constructed benchmark and real-world contracts.

RQ 2: Characteristics of Vulnerable Contracts

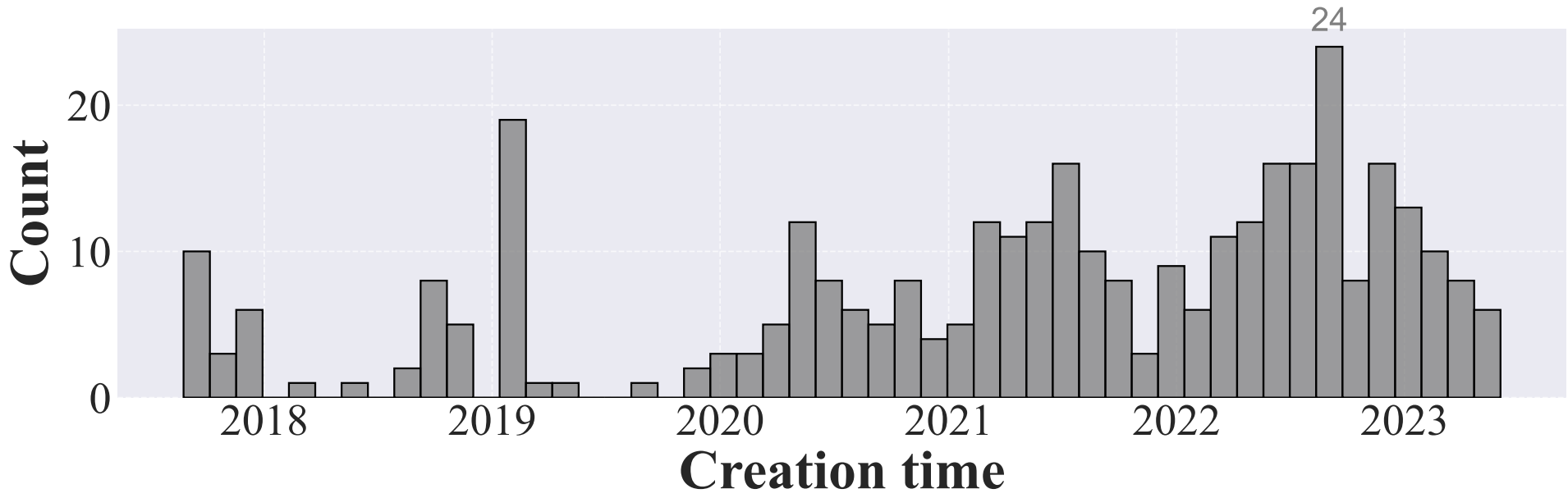


Fig 3. Distribution of vulnerable ones by creation time.

Most vulnerable contracts are found in 2022-2023

RQ 2: Characteristics of Vulnerable Contracts

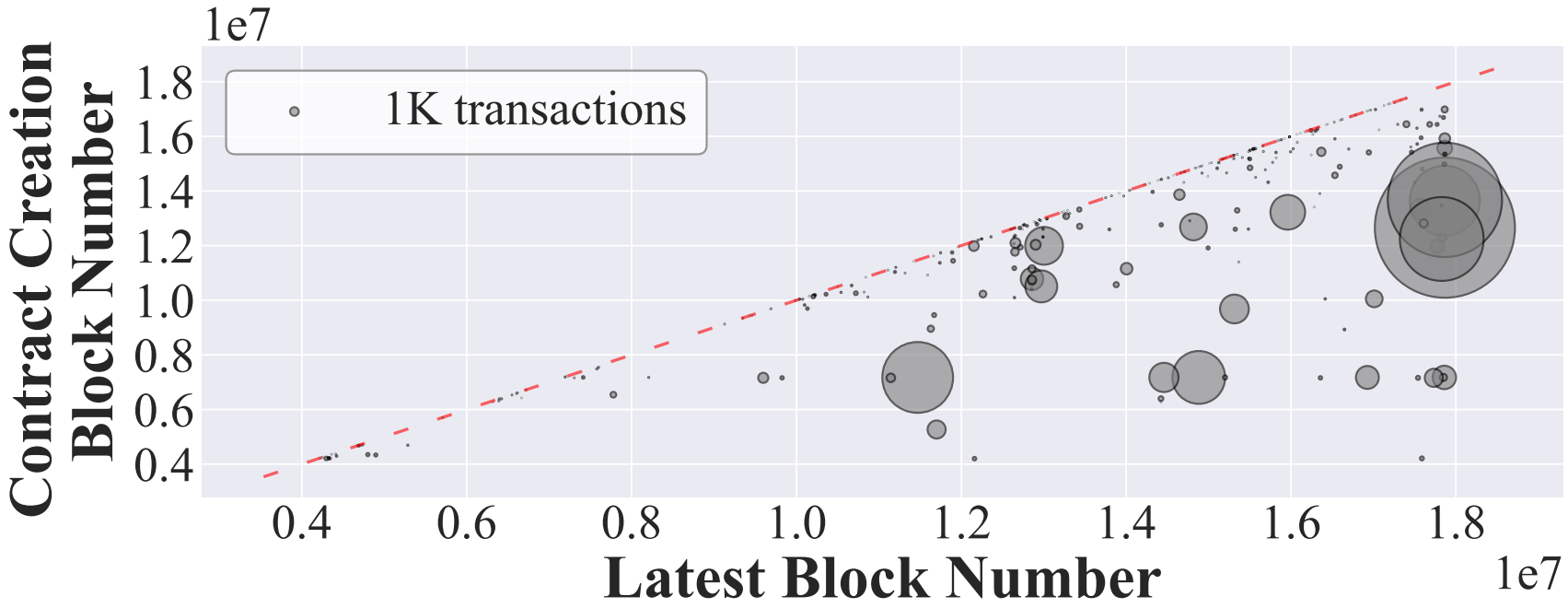


Fig 4. Relations between creation time and lifespan for vulnerable contracts.

A significant number of contracts had a very short lifespan.

RQ 2: Characteristics of Vulnerable Contracts

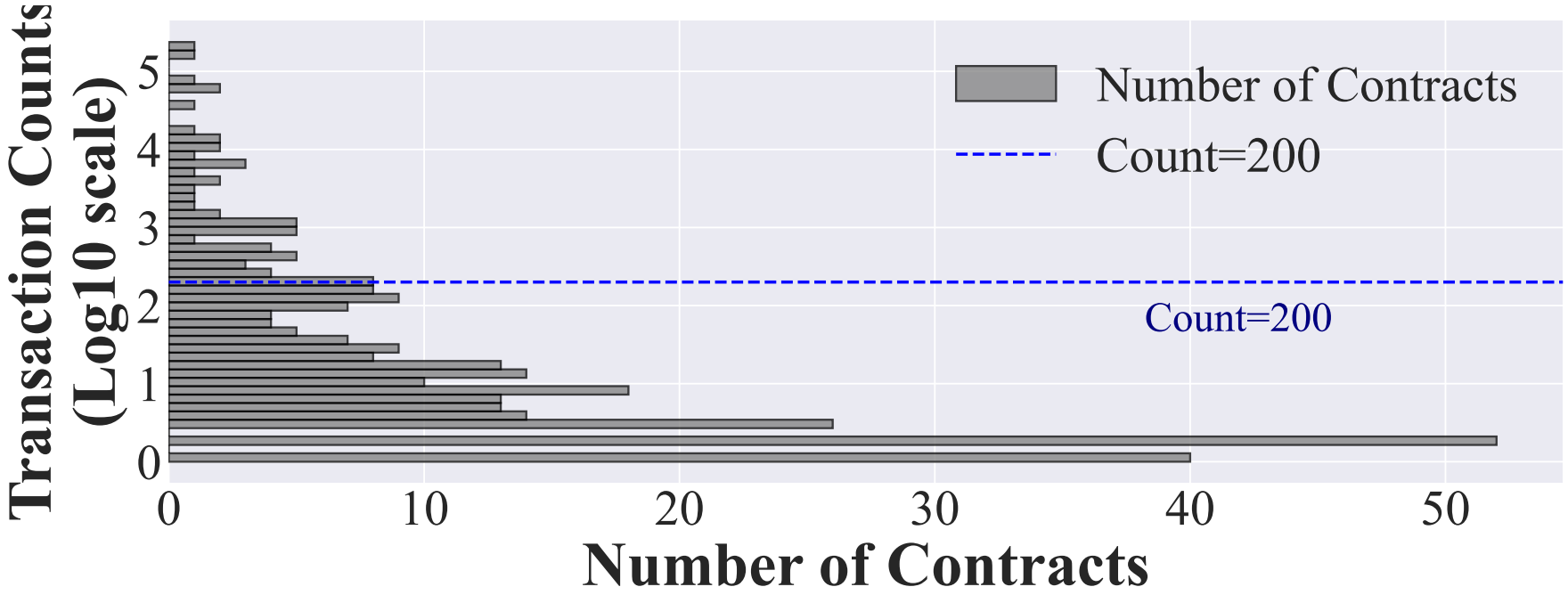


Fig 5. Distribution of the number of transactions involved in vulnerable contracts.

Most vulnerable contracts have fewer transactions.

RQ 2: Characteristics of Vulnerable Contracts

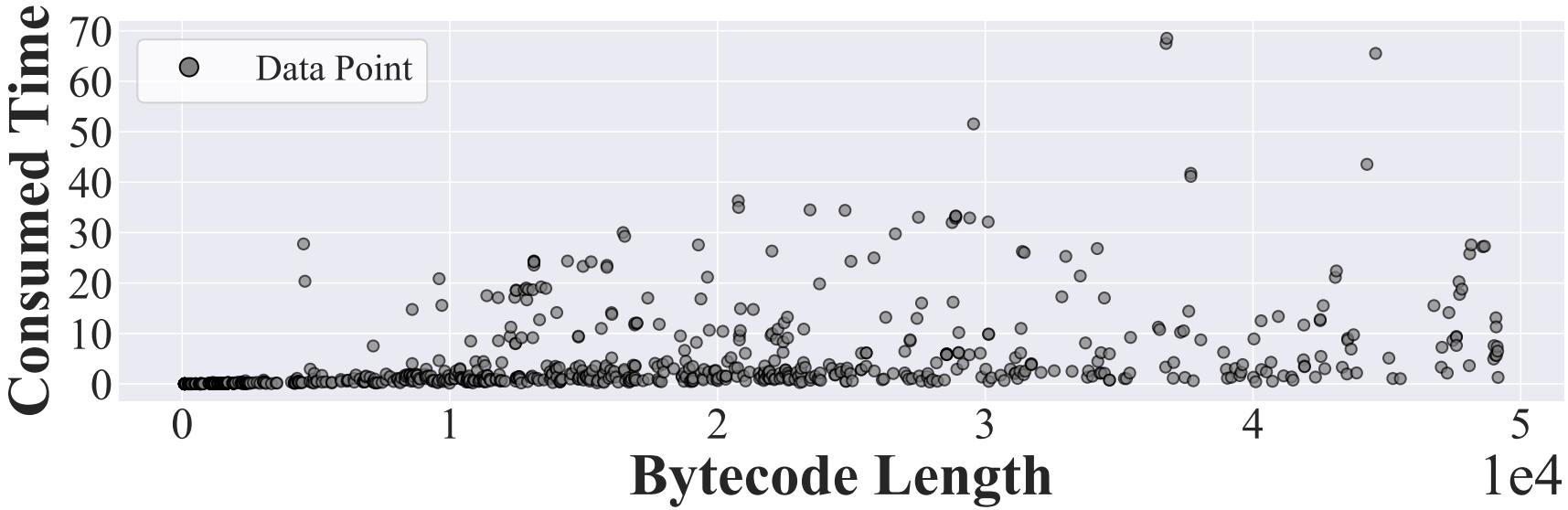


Fig 6. The relationship between the bytecode length and the time consumed on each case.

The detection time does not grow exponentially with the length of the bytecode.

RQ 2: Characteristics of Vulnerable Contracts

- RQ2: How many smart contracts are vulnerable in the wild and what are their characteristics?

Answer:

- Around 68.4% vulnerable contracts are ERC-20 or ERC-721 tokens.
- Attackers tend to launch attacks dozens of days after the deployment for greater benefits.

RQ 3: Real-time Detection

- Chain: Ethereum, BSC
- Vulnerable Contracts: 9
- Time Span: 2023.4.15 – 2023.10.30

```
function 0xbc423deb(uint256 varg0, uint256 varg1, uint256 varg2) public payable {
    require((address(varg2)).code.size);
    v0 = address(varg2).slip(_ilk, msg.sender, 0 - varg1).gas(msg.gas);
    require(v0);
    v1, v2 = _gem.transfer(address(varg0), varg1).gas(msg.gas);
    require(v1);
    emit Exit(address(varg0), varg1);
}
```

External Call Check

No Whitelist Verfication

Post-call State Modification

RQ 3: Real-time Detection

- RQ3: Can AVVERIFIER be deployed as a real-time detection system?

Answer:

- AVVERIFIER has the ability to monitor in real-time.
- AVVERIFIER monitors a vulnerable contract with $30,000.

Takeaway Message

- New taint analysis method for detecting address verification vulnerabilities in Ethereum contracts.
- Outperforms existing tools in detecting address verification issues.
- Real-time detection of address verification vulnerabilities on blockchain platforms.

Contact us:
stl_hust@hust.edu.cn

Github QR Code