



北京大學
PEKING UNIVERSITY



北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY



NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE



西安交通大學
XI'AN JIAOTONG UNIVERSITY



THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

EventSpec: Defining and Detecting Event-Semantic Issues in Blockchain Ecosystems

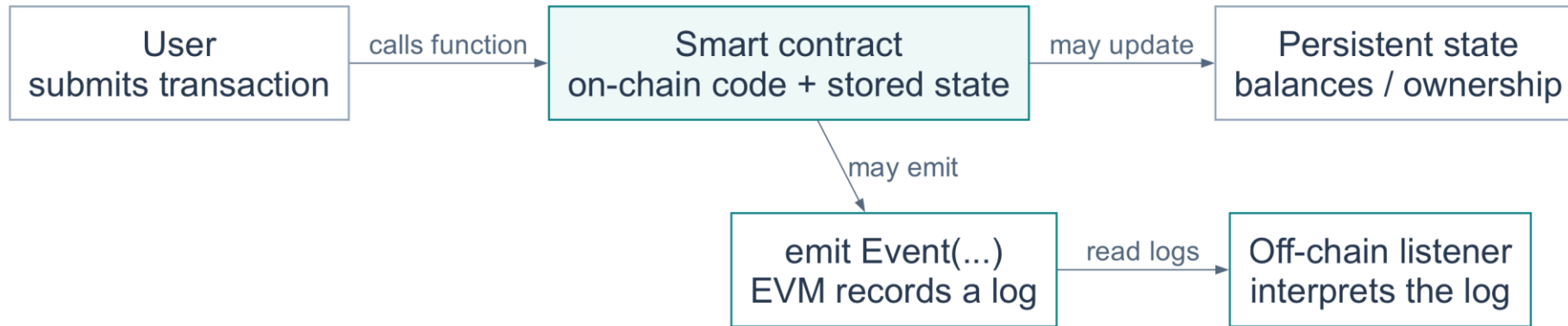
ISSTA 2026

Yixuan Liu¹, Yuxin Dong², Ye Liu³, Yin Wu⁴, Chengxuan Zhang¹, Xiapu Luo⁵, Yi Li¹

1 Nanyang Technological University, Singapore 2 Peking University 3 Beijing Institute of Technology
4 Xi'an Jiaotong University, China 5 Hong Kong Polytechnic University

Background: Smart Contracts and Events

A smart contract is a program stored on a blockchain. Transactions trigger its execution, which can update persistent state or transfer assets.



Real example: a USDC Transfer log

[illegible]

The log identifies the emitting contract and encodes event parameters.

Applications interpret these logs to display transfers or trigger actions.

Nearly 98% of successful contract-involved transactions emitted logs in our measurement.

Sources: EventSpec §2; Blockscout tx 0xe479...cbbd. Address and topics/data are cropped from the same log.

Background: Token Standards

Logs carry an emitter address, topics, and data.

A shared event signature does not guarantee the same layout or meaning.

ERC-20 · fungible tokens

```
event Transfer(  
    address indexed from,  
    address indexed to,  
    uint256 value  
);
```

3 topics + 1 data word

ERC-721 · non-fungible tokens

```
event Transfer(  
    address indexed from,  
    address indexed to,  
    uint256 indexed tokenId  
);
```

4 topics + 0 data words

A listener must validate the emitter, layout and meaning of each event.

Sources: EventSpec, Section 2.2; ERC-20 and ERC-721 specifications. Topic counts include the event signature.

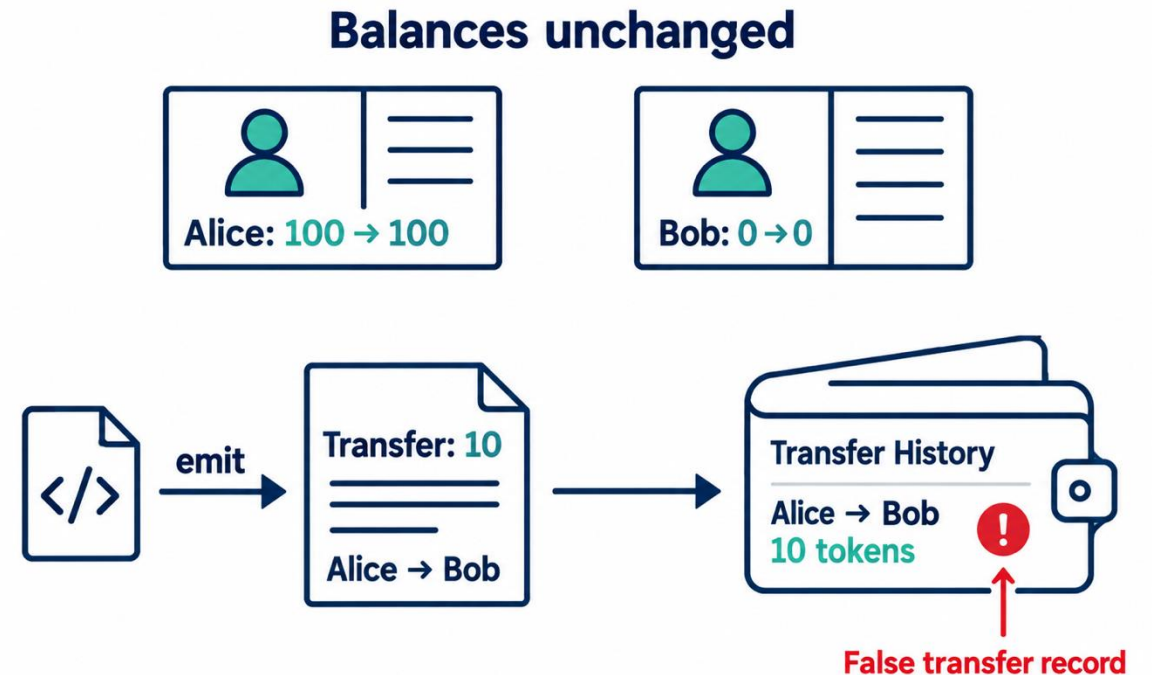
Motivation: Event–State Inconsistency

A misleading Transfer log can create a false transfer record in a wallet or indexer.

Contract code

```
function transfer(address to,  
                  uint value) external {  
    // missing update to balances  
    emit Transfer(msg.sender, to, value);  
}
```

Missing balance updates



Impact: a wallet may display a transfer that never occurred.

Source: EventSpec, Fig. 3 and §3.3.2. Conceptual illustration; balances and the 10-token amount are illustrative.

Empirical Study: Data and Analysis

Audit findings and security incidents provide evidence for the defect taxonomy.

Evidence source	Collected / keyword-filtered	Retained after review
Security-company reports	46 event/log-related incidents	11 incidents
FORGE audit reports	6,454 reports → 1,354 findings	1,294 findings in 1,020 reports

From evidence to five defect classes

- 1 Relevance review** Two independent reviewers; resolve disagreements ($\kappa = 0.89$).
- 2 Evidence cards** Record the issue, code evidence and underlying root cause.
- 3 Open card sorting** 40% jointly derive categories; 60% independently classify ($\kappa = 0.95$).

Source: EventSpec, §§3.1–3.2. Incident sources: SlowMist, BlockSec, PeckShield and CertiK.

Empirical Study: Defect Taxonomy

Five defect classes describe how event semantics can diverge from contract behavior.

Issue	Definition
Event Collision	Different execution paths emit indistinguishable logs with overlapping parameter domains.
State–Event Mismatch	Logs disagree with the state transition they claim to summarize.
Unauthorized Event Emission	Privileged events lack required access-control evidence.
Event Emission Mismatch	Events are missing, extra, duplicated, misordered or of the wrong type.
Event Parameter Mismatch	Event layout, parameter order, origin or value binding is inconsistent.

Source: EventSpec, Table 1 and Section 3.3. Event Emission Mismatch accounts for about 85% of retained audit findings.

Case Study: Event Collision

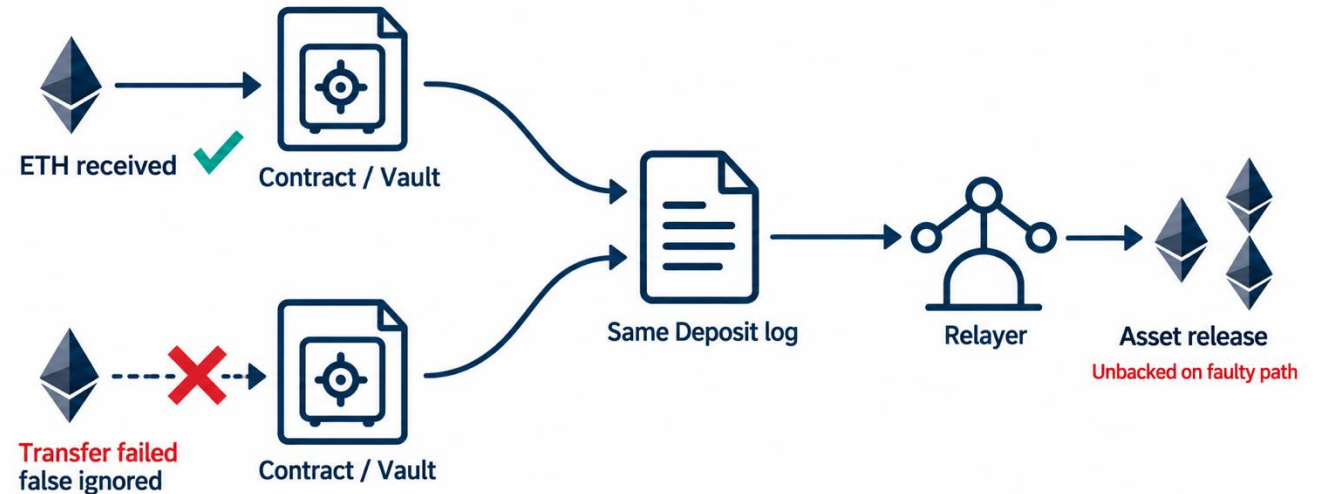
Two functions can emit the same Deposit log, even when only one receives ETH.

Genuine ETH deposit

```
function depositETH() external payable {  
    // ETH enters through msg.value  
    emit Deposit(msg.sender, address(0),  
                msg.value);  
}
```

Faulty token-deposit path

```
function deposit(address token,  
                uint amount) external {  
    // Missing: token != address(0)  
    // Returns false without reverting.  
    bool ok = _transferFrom(token, amount);  
    // BUG: ok ignored; still emits.  
    emit Deposit(msg.sender, token,  
                amount);  
}
```



Identical logs can make an unfunded deposit look genuine to a relayer.

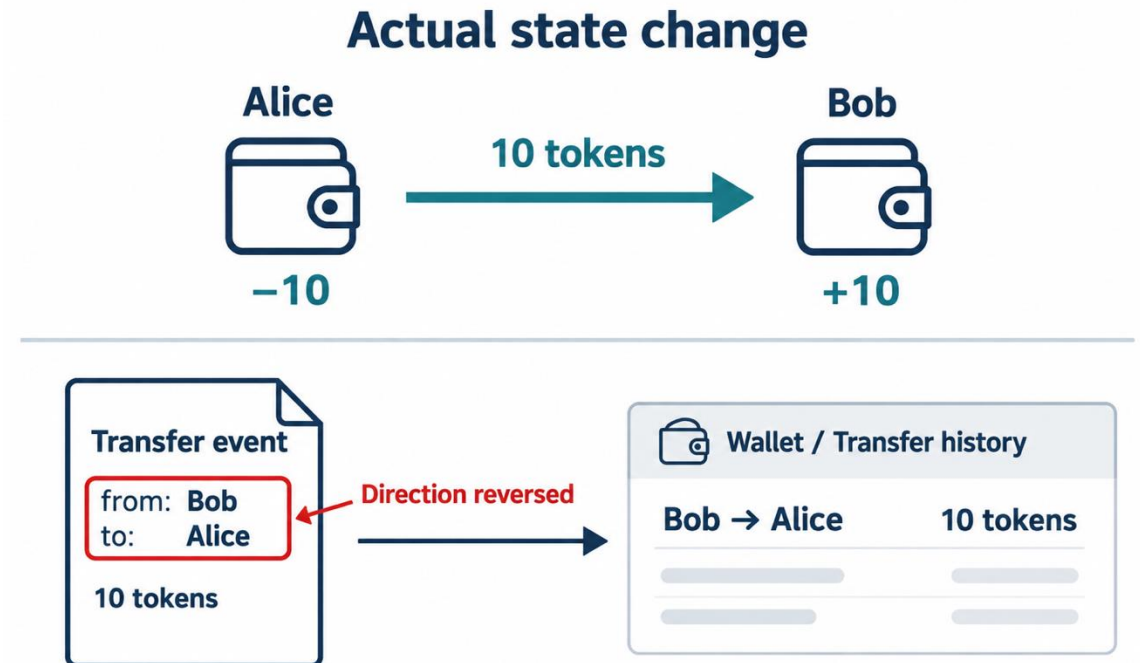
Case Study: Event Parameter Mismatch

The balances record Alice → Bob, but the Transfer event reports Bob → Alice.

Correct balances; swapped event fields

```
function transfer(address to,  
                  uint v) external {  
    balances[msg.sender] -= v;  
    balances[to] += v;  
    emit Transfer(to, msg.sender, v);  
}
```

Expected: `Transfer(msg.sender, to, v)`

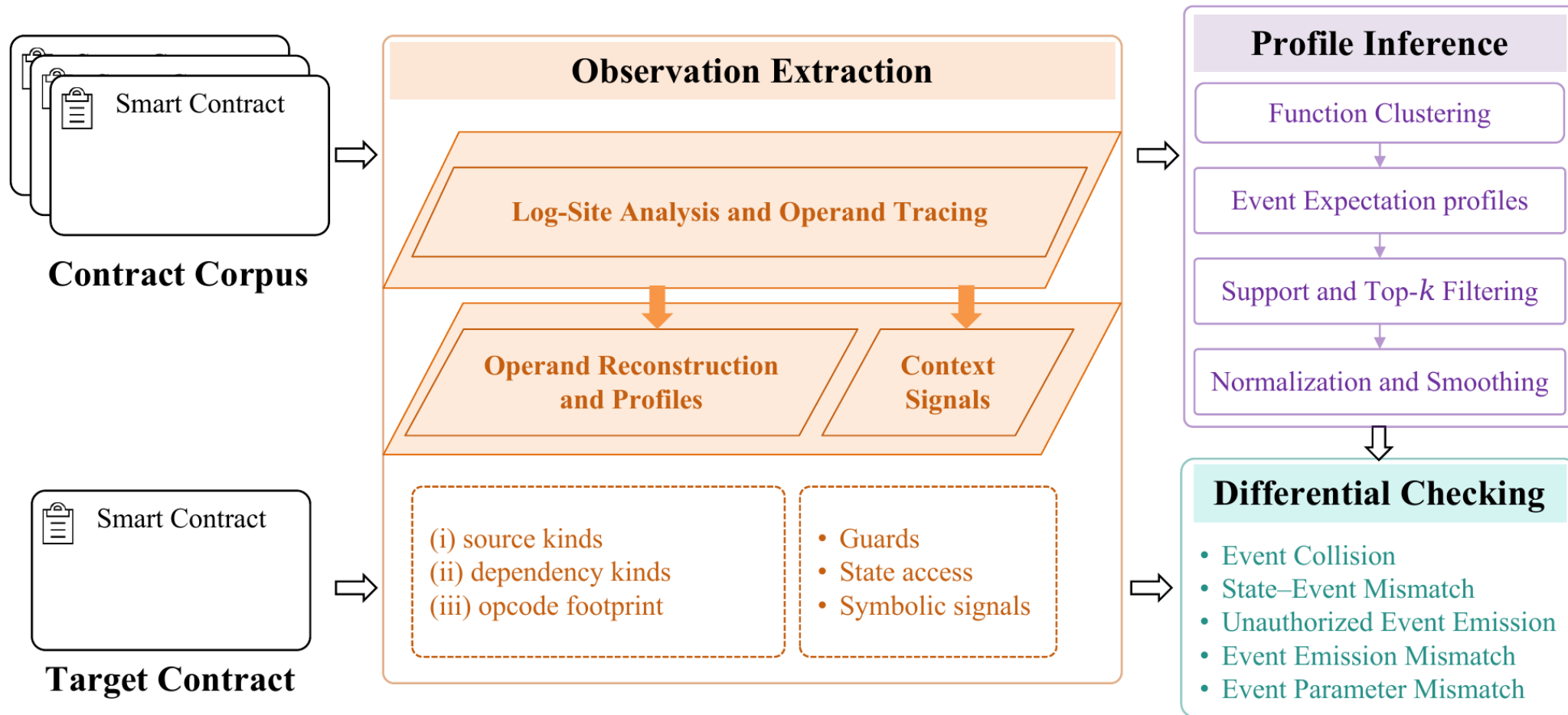


Impact: the event reverses the actual transfer direction.

Source: EventSpec, Fig. 6 and §3.3.5. Conceptual illustration; Alice, Bob, and 10 tokens explain the swapped parameters.

Methodology: EventSpec Overview

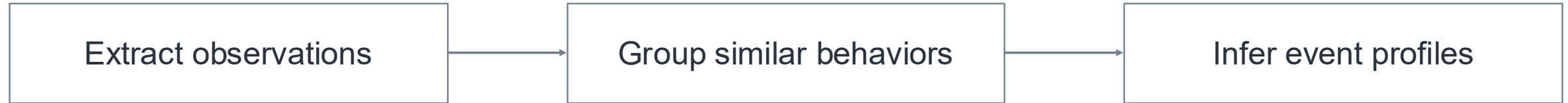
Extract observations, infer expected behavior, then check a target against the learned profiles.



Source: EventSpec, Fig. 8 (original paper figure). The off-chain testing harness is presented separately.

Methodology: Learning Event Profiles

Recurring contract behaviors provide empirical expectations for event emission.



Evidence from each function

Evidence	What we record
Parameter origins	Inputs, constants or stored values
Guards	Checks required to reach the event
State relations	Relevant reads and writes

What an event profile records

Layout

Topic and data positions

Semantics

Parameter origins, guards, state

Emission

Event types and counts

Layout describes log encoding; semantic constraints describe expected behavior.

Source: EventSpec, §§4.1–4.4. Learned profiles describe recurring behavior; they are not formal guarantees.

Methodology: Differential Checking

A matching event layout can still hide missing authorization.

Evidence	Inferred profile	Target log
Event signature	t	t
Layout	3 topics + 1 data word	3 topics + 1 data word
Operand origin	topic1 ← CONST	topic1 ← CONST
Guard evidence	owner	none

Matched profiles pass similarity, support and confidence gates.

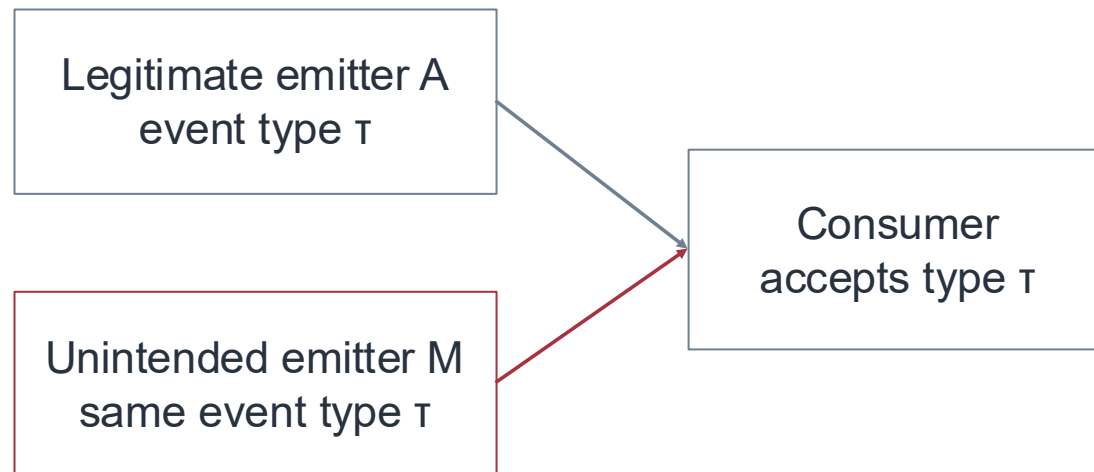
Finding: Unauthorized Event Emission

Source: EventSpec, Fig. 10, Table 2 and Section 4.5. Guard evidence is checked after bounded interprocedural attribution.

Methodology: Off-Chain Test Harness

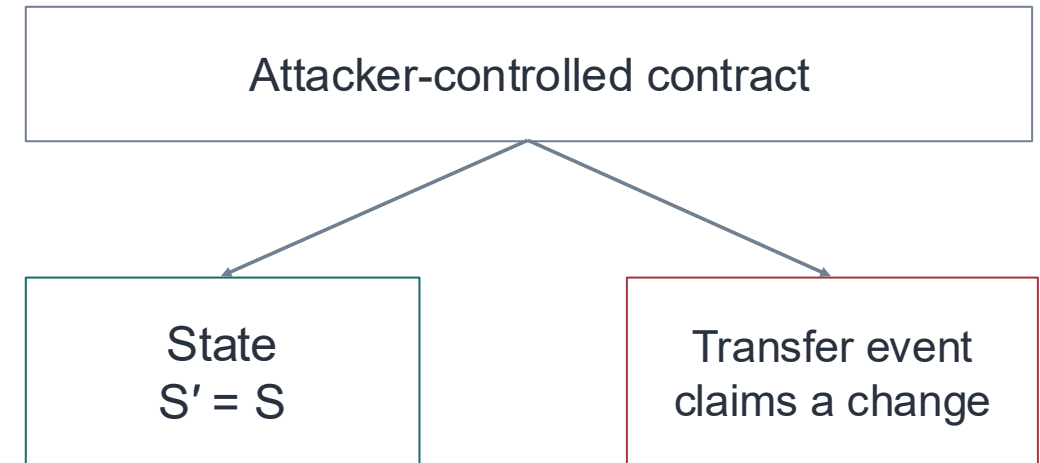
The harness tests emitter validation and consistency between emitted events and state.

(a) Event Origin Confusion (EOCA)



One transaction contains logs from A and M.
The consumer fails to bind each log to its emitter.

(b) Event–State Desynchronization (ESDA)



The consumer infers a transfer from the event
without verifying the unchanged state.

Evaluation: Datasets and Ground Truth

Each dataset answers a different question about inference, accuracy or detection yield.

Purpose	Dataset / unit	Source and selection
Learn profiles	212,072 contracts	Smart Contract Sanctuary; bytecode deduplication and decompilation filter
Validate profiles	14 ERCs / 26 event definitions	Final ERC standards; at least 50 implementations per ERC
Measure accuracy	95 labeled contracts	Random subset of the target pool; manual labels ($\kappa = 0.97$)
Measure detection yield	6,617 active contracts	NumScout-curated Sanctuary set; activity, balance and compilation filters

The 95-contract accuracy sample is part of the 6,617-contract target pool.

Source: EventSpec, §5.2. Event Collision accuracy additionally uses five known card-sorting cases.

Evaluation: Specification Inference

Reference: 14 Final ERC standards and 26 event definitions, derived from ERC documents.

Signature coverage

Layout agreement

Semantic agreement

80.77%

100%

90.48%

21 / 26 events

21 / 21 matched events

19 / 21 matched events

Different denominators

Layout and semantic agreement are conditional on a matched signature.

Semantic constraints include required checks and the sources of event parameters.

Source: EventSpec, Tables 3–4 and Section 5.3. Default $\alpha = 1$, minimum support = 20 and top-k = 3.

Evaluation: Detection Accuracy

Ground truth: 95 randomly sampled target contracts; manual labeling agreement $\kappa = 0.97$.

Issue	Precision (%)	Recall (%)	F1 (%)
Event Collision*	100.00	100.00	100.00
State–Event Mismatch	83.33	100.00	90.91
Unauthorized Event Emission	84.62	91.67	88.00
Event Emission Mismatch	98.90	98.90	98.90
Event Parameter Mismatch	66.67	90.00	76.60

Precision (comprehensive): 90.17%

Macro-average recall: 96.11%

Average F1 (%): EventSpec 90.88; Slither 17.92 under the paper’s category mapping.

Source: EventSpec, Table 5. 95 labeled contracts; *Event Collision effectiveness uses five additional card-sorting cases.

Evaluation: Detection Yield

Target pool: 6,617 active, verified contracts from the NumScout-curated Sanctuary set.

Issue	Contracts reported	Share of 6,617 (%)
Event Collision	0	0.00
State–Event Mismatch	4,159	62.85
Unauthorized Event Emission	405	6.12
Event Emission Mismatch	6,514	98.44
Event Parameter Mismatch	3,664	55.37

The 98.44% yield is dominated by no_event_emission.

Source: EventSpec, Table 5 and Section 5.4. Each issue is counted at most once per contract; categories overlap.

Evaluation: Off-Chain Case Studies

Targets: incident/bounty-listed consumers and 100 active Dune-listed contracts.

System	Vector	Paper evidence
Bridge relayers	EOCA	One known case reproduced; one additional relayer confirmed
Blockchain explorers	ESDA	At least five explorers displayed spoofed transfers as legitimate
Cryptocurrency wallets	ESDA	Six reports; four confirmed, including a \$600 bounty; two pending
NFT marketplaces	ESDA	Sleepminting reproduced on OpenSea and Rarible testnets

On-chain UEM audit

Three DeFi projects and one GameFi project allowed unauthorized privileged events.

Source: EventSpec, §5.5. Consumer selection also includes widely used explorers and open-source bridge relayers.

Discussion: Mitigation

Validate event semantics at both the emitting contract and the consuming service.

Contract-side safeguards

- Update state before emission
- Guard privileged events
- Match parameter order and indexing
- Disambiguate overlapping event meanings

Consumer-side safeguards

- Allowlist the intended emitter
- Validate log shape and execution status
- Verify state for high-risk transfers
- Bind events to intended calls

Source: EventSpec, Sections 5.4, 5.6 and mitigation discussion. Impact depends on how off-chain consumers interpret the logs.

Conclusion: Event-Semantic Security

Event-semantic security requires checking what a log claims about execution.

Define

Five defect classes connect audit evidence to recurring event-semantic failures.

Detect

Corpus-derived profiles enable explainable checks of emission behavior and constraints.

Validate

Off-chain case studies show why emitter and state verification matter.

Treat events as claims to verify.