

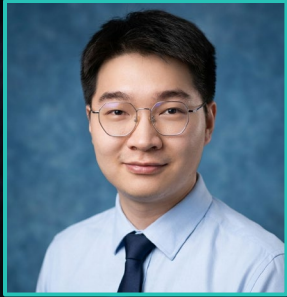


Fortifying the Foundation: LLM-Empowered Differential Testing for the Ethereum Infrastructure

Speakers: Jie Ma, Ningyu He

Contributors: Chiachih Wu, Haoyu Wang, Ying Gao, Yinliang Yue

About Us



Jie Ma

Eng.D Candidate, Beihang University
& Zhongguancun Laboratory



Ningyu He

Research Assistant Professor, PolyU
& Security Consultant, Amber Group



Chiachih Wu

Partner & Head of Web3 Security,
Amber Group



Haoyu Wang

Professor, Huazhong University
of Science and Technology



Ying Gao

Associate Professor, Beihang University
& Zhongguancun Laboratory



Yinliang Yue

Professor, Zhongguancun Laboratory

Agenda

- I. Overlooked Ethereum Client Security
- II. OpDiffer: Opcode-Level EVM Differential Testing
- III. APIDiffer: Ethereum Client API Differential Testing
- IV. Bugs Dissection
- V. Implication & Conclusion



Part I: Overlooked Ethereum Client Security

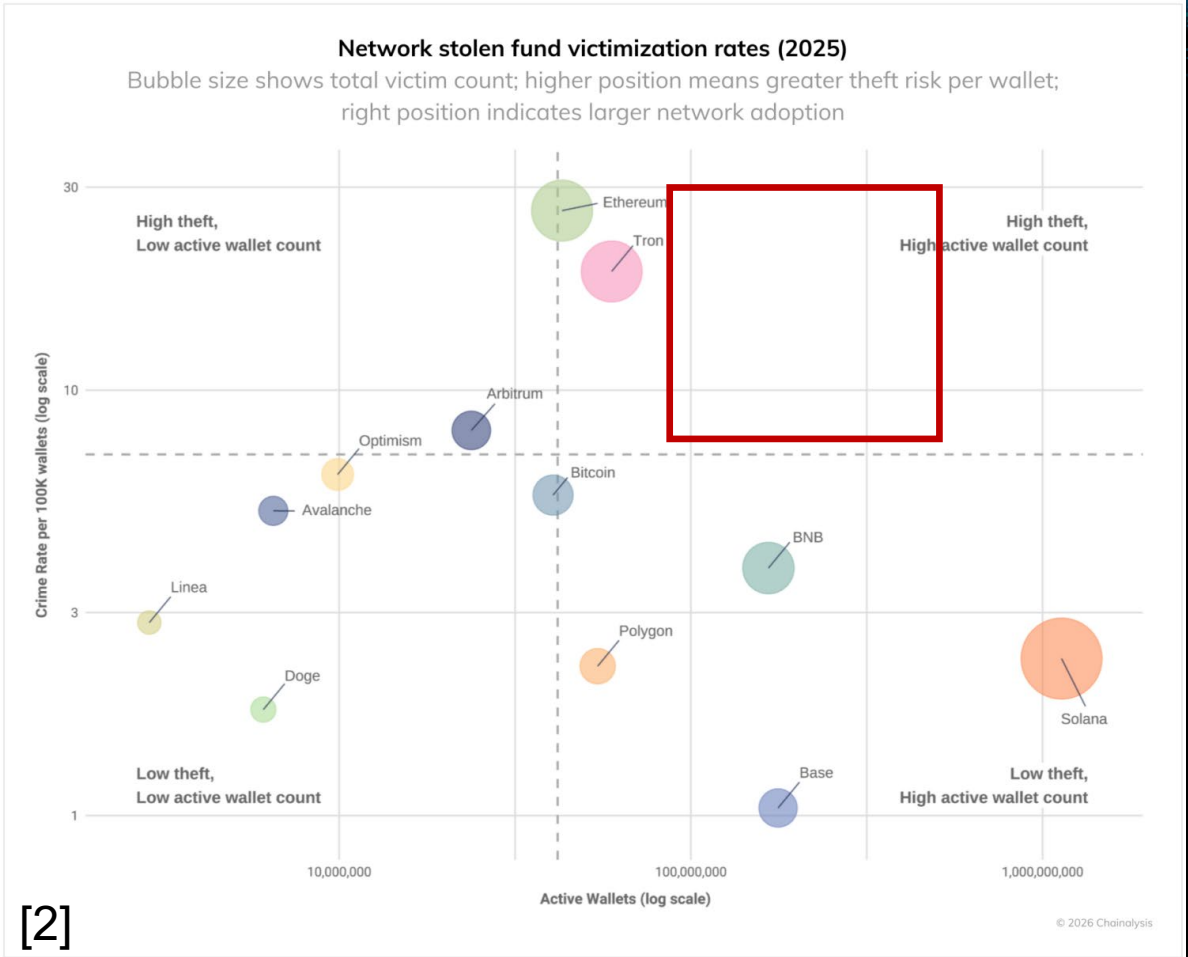
Ethereum

- Ethereum is the second largest blockchain platform, following Bitcoin
- Killer apps:
 - Smart contracts
 - 90M+ deployed on Ethereum
 - Decentralized Finance (DeFi)
 - \$52.8B+ total value locked (TVL) in DeFi protocols

#	Name ▾	Market Cap ▾
1	 Bitcoin BTC	\$1,435,472,973,915
2	 Ethereum ETH	\$263,789,979,757
3	 Tether USDT	\$184,134,381,562
4	 XRP XRP	\$82,285,462,784
5	 BNB BNB	\$81,950,278,266

Security Incidents in Ethereum

- Vulnerabilities in Ethereum = Possible money loss
- Security reality in 2025 [1]
 - 310 security incidents
 - \$1.7B in total value exploited
 - An average of \$5.7M stolen per incident



Security Incidents in Ethereum

ethereum security

About 235,000 results (0.07 sec)

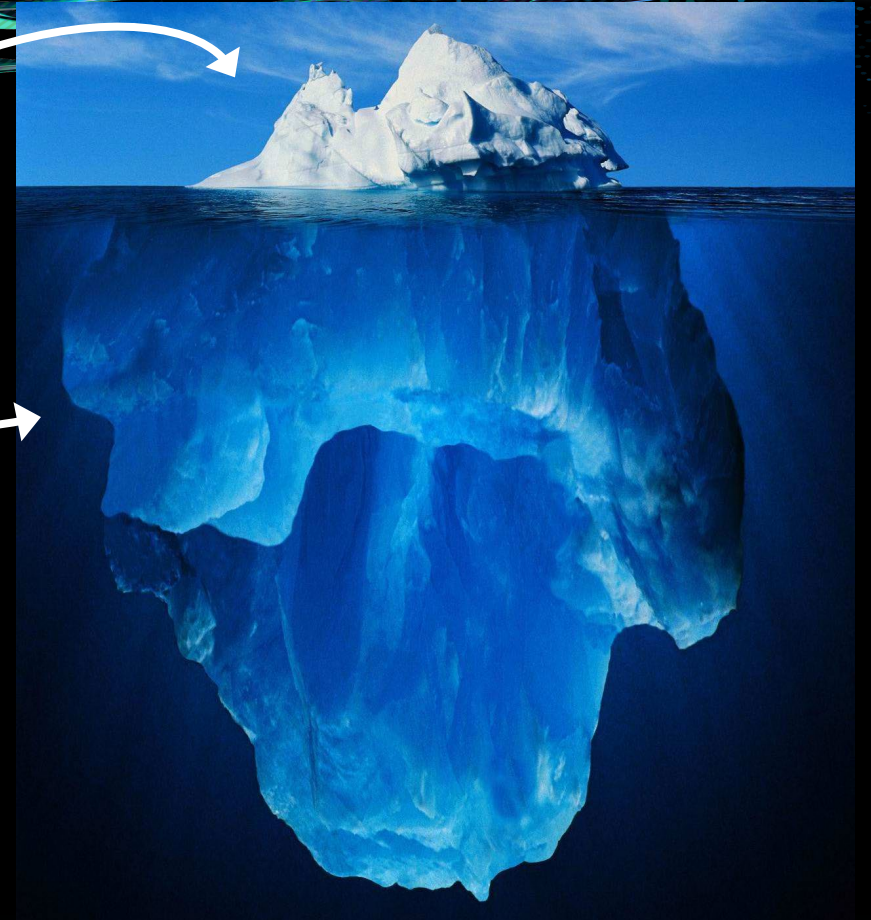
ethereum security contracts

About 137,000 results (0.16 sec)

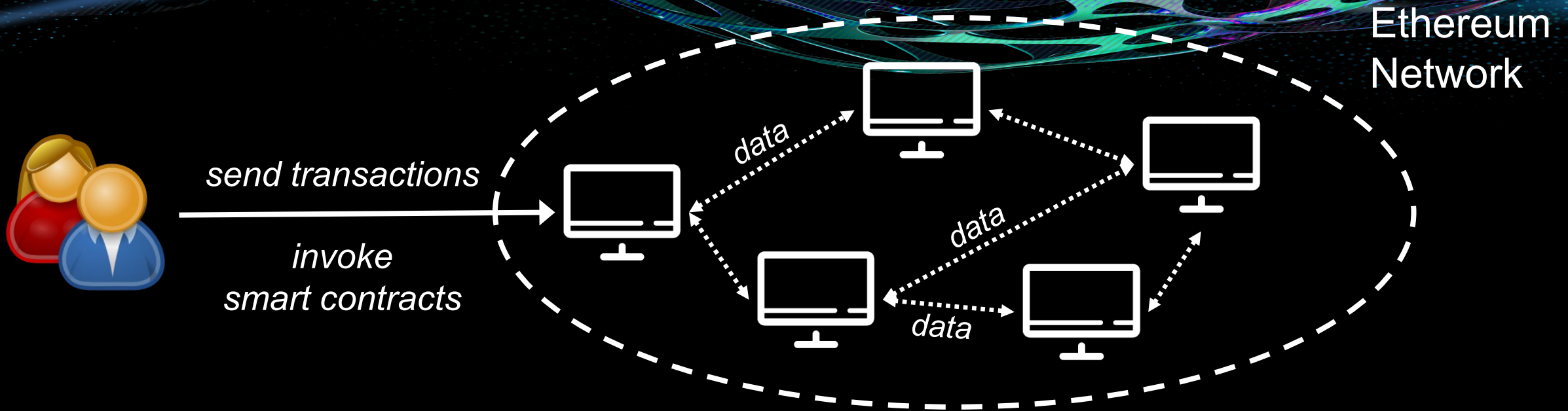
What users can
see and interact

Ethereum Infra
under the water

- ~58% of Ethereum security research is about smart contracts
- What about the security in infra?



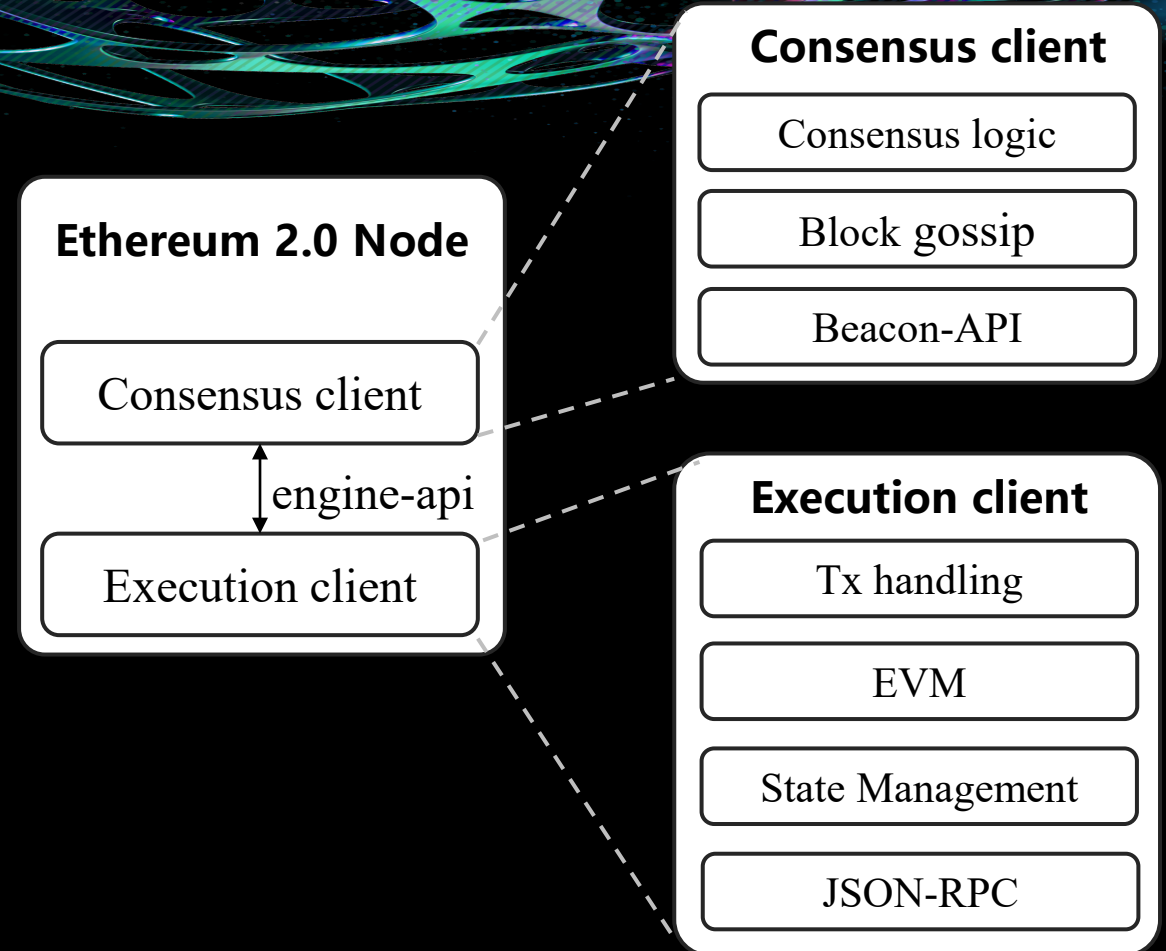
Ethereum Network Model



- Node is the bridge between users and Ethereum network

Ethereum Node Infrastructure

- Dual-engine architecture
 - Consensus Layer (CL) Client
 - Orders blocks and finalizes consensus
 - Beacon-API
 - Execution Layer (EL) Client
 - Executes transactions and updates state
 - Ethereum virtual machine (EVM) for contract execution
 - JSON-RPC
- Are there any bugs?



#1: Shallow Copy Bug (CVE-2020-26241)

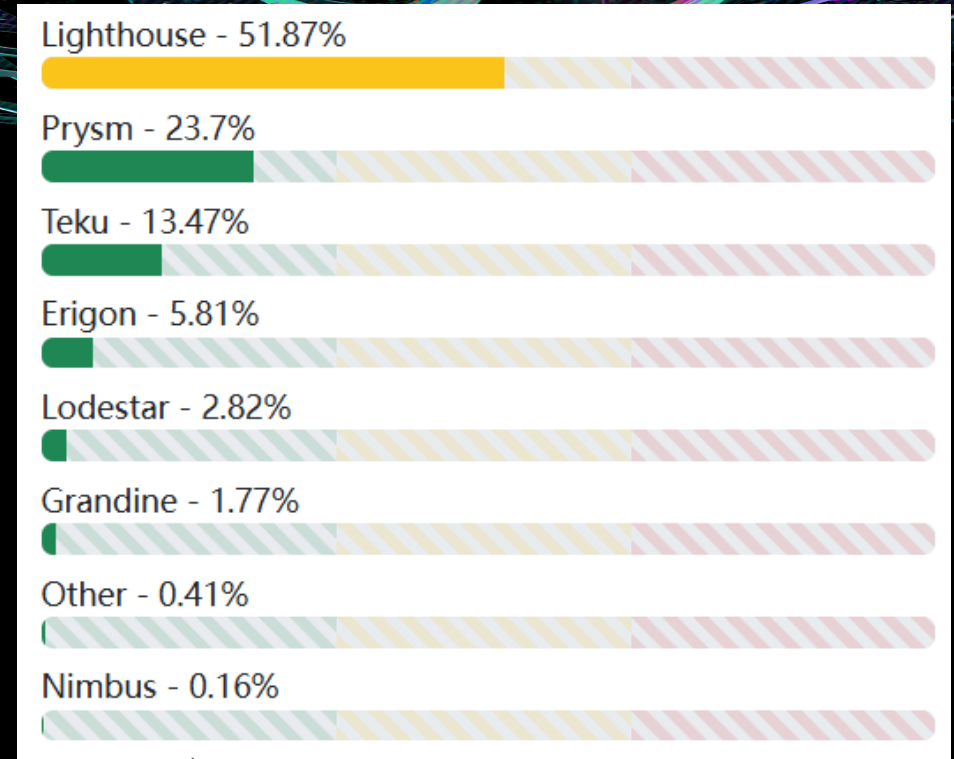
```
// if the operation clears the return data (e.g. it has returning data)
// set the last return to the result of the operation.
if operation.returns {
-   in.returnData = res
+   in.returnData = common.CopyBytes(res)
}
```



- Geth (the most popular EL client) unintentionally uses shallow copy instead of deep copy [3]
- Unintended aliasing between EVM execution contexts that should have been isolated
 - Chain fork -> took down the biggest service provider (Infura), and affected major downstream platforms (MetaMask and Binance)

#2: Prysm CL Client Bug

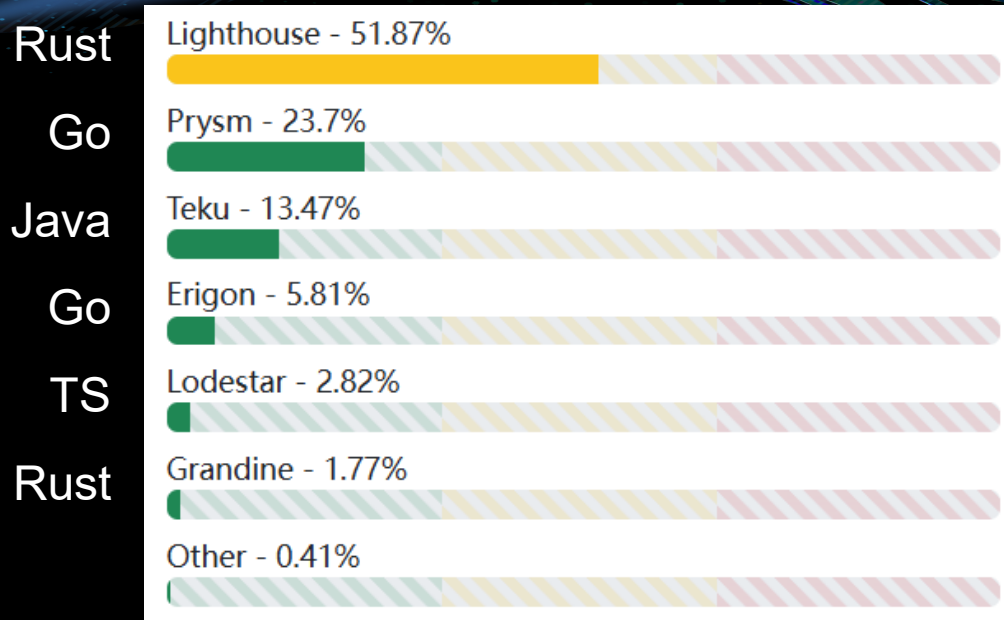
- 2025.12.04, Ethereum Fusaka Mainnet Upgrade [4]
- A bug in Prysm (CL client) could lead to repeatedly state replay and reconstruction
 - Resource exhaustion
 - Network participation drop to 75%, where 66.6% is the threshold to reach the **finality**



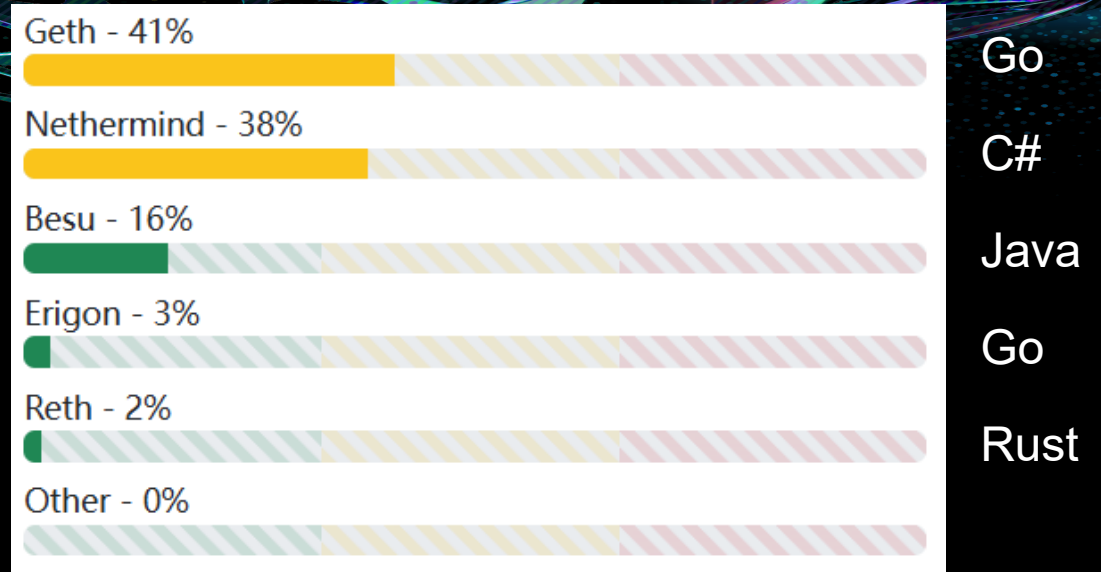
What if Lighthouse is took down?

Ethereum Client Diversity

CL Clients



EL Clients



[5]

- Client diversity reduces single points of failure across both execution and consensus layers
- They are developed according to specs, publicly maintained through open community processes

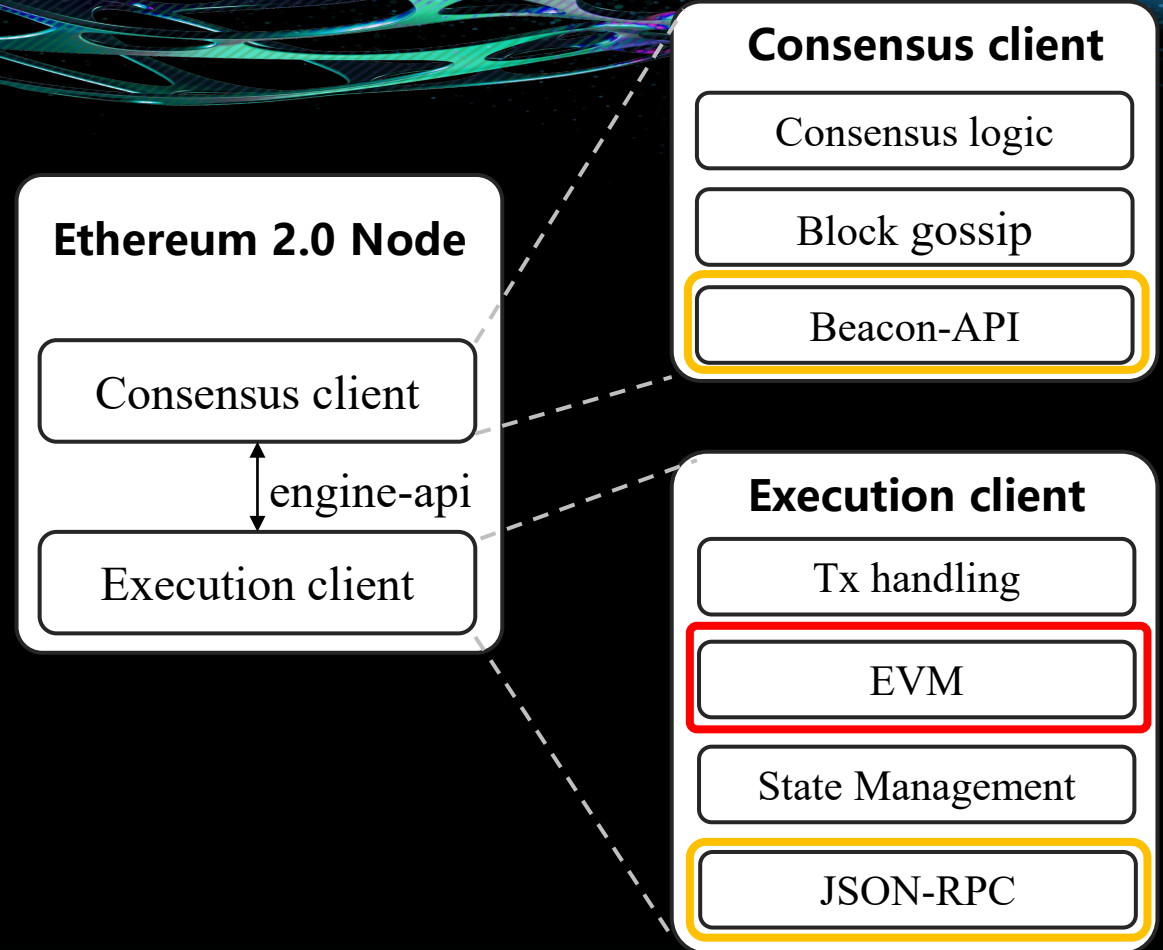
Differential Testing



- How to perform testing/analysis on such diversified targets?
 - Fuzzing? Symbolic execution?
- Differential testing
 - One input, multiple nodes (30 nodes = 6 CL clients x 5 EL clients)
 - Semantically-identical, implemented in different ways
 - Compare outputs/states
 - An inconsistency indicates a potential bug

Testing Targets

- Research focus
 - **OpDiffer** -> EVM
 - EL client
 - Consistency of smart contract handling
 - **APIDiffer** -> API
 - Beacon-API (CL client) & JSON-RPC (EL client)
 - Consistency of response to requests



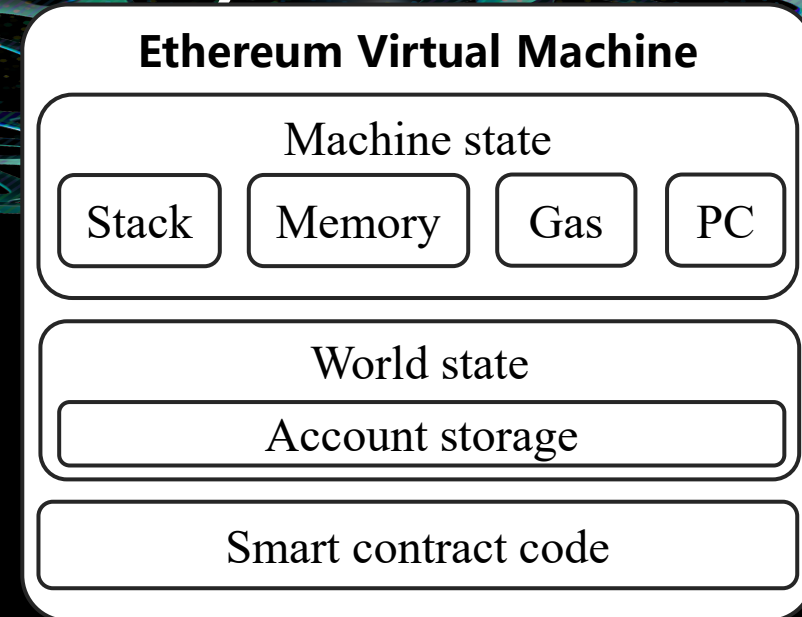


Part II: OpDiffer

**LLM-Assisted Opcode-Level Differential Testing
of Ethereum Virtual Machine
*ISSTA'25***

Ethereum Virtual Machine (EVM)

- Smart contract –compiled→ bytecode
- EVM in EL clients loads and executes smart contract bytecode



```
contract foo {  
    int256 count = 0x10;  
    function add() public {  
        count += 0x20;  
    }  
}
```

Source code in Solidity



```
0x...6010602001... // EVM bytecode  
  
PUSH1 10           // opcode PUSH1, operand 0x10  
PUSH1 20           // opcode PUSH1, operand 0x20  
ADD                // opcode ADD, pop 0x20 0x10, push result
```

EVM bytecode

Challenge

```
def exp(evm: Evm) -> None:
    base = Uint(pop(evm.stack))
    exponent = Uint(pop(evm.stack))
    exponent_bits = exponent.bit_length()
    exponent_bytes = (exponent_bits + Uint(7)) // Uint(8)
    charge_gas(
        evm, GAS_EXPONENTIATION + GAS_EXPONENTIATION_PER_BYTE * exponent_bytes
    )
    result = U256(pow(base, exponent, Uint(U256.MAX_VALUE) + Uint(1)))
    push(evm.stack, result)
    evm.pc += Uint(1)
```

The specification of the EXP opcode

- In EVM, handling an opcode, you should consider:
 - Stack management
 - Gas management
 - Semantic behavior
 - State update

Challenge #1 & Solution

- Source-code-level seed generation?

- Limited open-source contract (~0.87%)
- Contract type is singular, like lots of ERC20 contracts
- Hard to trigger corner cases

Diversity

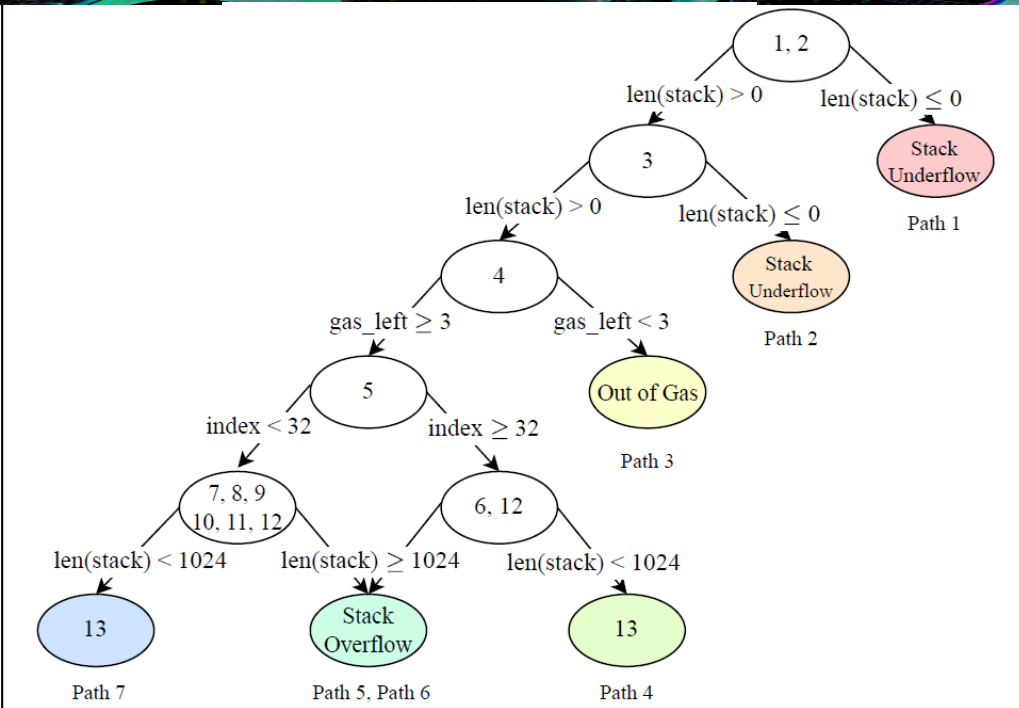
- Bytecode-level seed mutation?

- Syntax correct but semantic incorrect
 - e.g., JUMP without JUMPDEST

Semantic
Validity

Challenge #1 & Solution

```
1 def get_byte(evm: Evm) → None:
2     # STACK
3     byte_index = pop(evm.stack)
4     word = pop(evm.stack)
5     # GAS
6     charge_gas(evm, GAS_VERY_LOW)
7     # OPERATION
8     if byte_index ≥ U256(32):
9         result = U256(0)
10    else:
11        extra_bytes_to_right = U256(31) - byte_index
12        word = word >> (extra_bytes_to_right * U256(8))
13        word = word & U256(0xFF)
14        result = word
15    push(evm.stack, result)
16    # PROGRAM COUNTER
17    evm.pc += Uint(1)
```

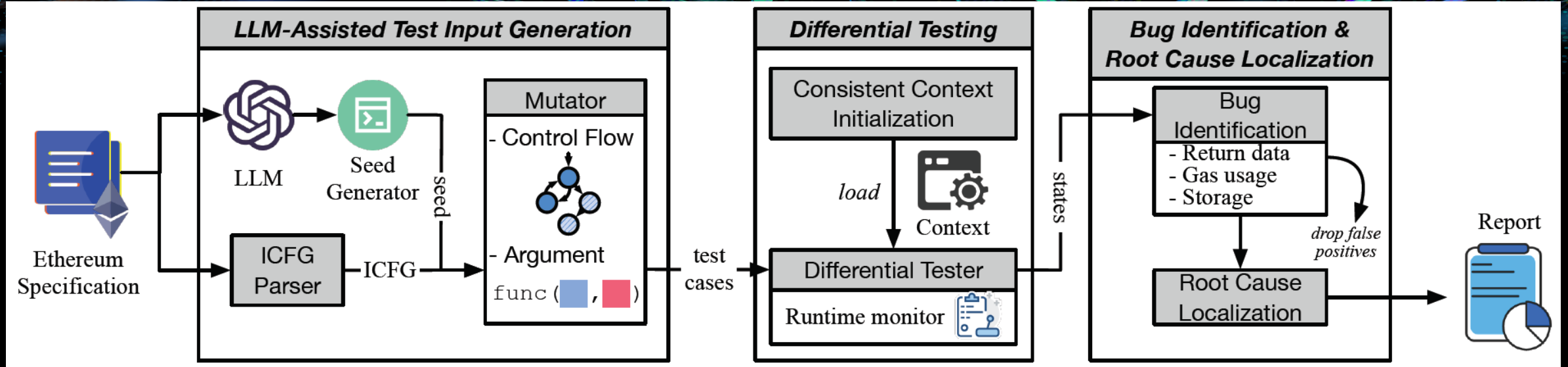


- Use LLMs to generate a minimal and semantically valid bytecode snippet that correctly invokes a target opcode
- Extract **ICFG** according to the spec
- Augment the seed and provide corresponding arguments to cover other paths

Challenge #2 & Solution

- Automated bug identification and root cause localization
- Solutions:
 - Inconsistency → Bug
 - Return data
 - Gas usage
 - State, like storage
 - Bug → Opcode (EVM log)
 - Opcode → Opcode Implementation Function (LLM)
 - Bug + Opcode + Function Implementation → Root cause

Workflow of OpDiffer



- LLM-Assisted Test Input Generation
- Differential Testing
- Bug Identification & Root Cause Localization



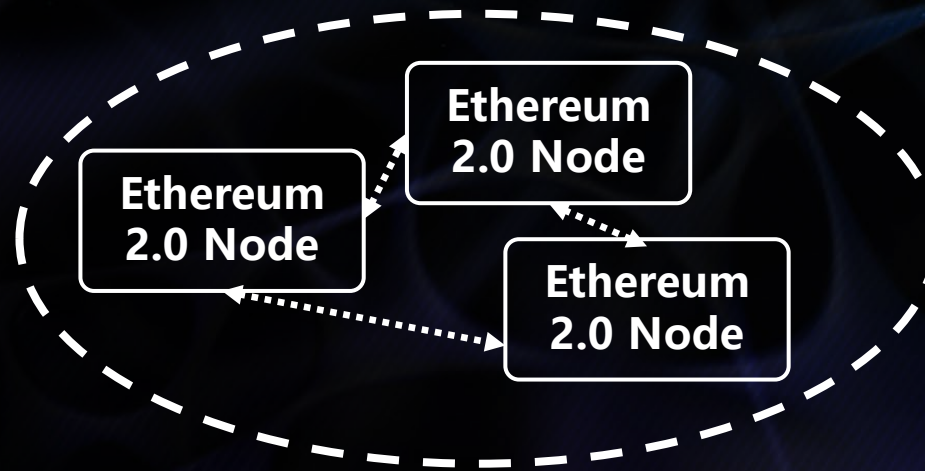
Part III: APIDiffer

**When Specifications Meet Reality: Uncovering API
Inconsistencies in Ethereum Infrastructure
*OOPSLA'26***

Ethereum Client API

Users
Web3 Applications
Wallets
...

request →
← *respond*



A screenshot of a mobile application interface for configuring an Ethereum network. The screen has a white background with a dark header bar containing the text 'Ethereum' and a fox icon. Below the header, there is a 'Network name' field with the value 'Ethereum'. A red rectangular box highlights the 'Default RPC URL' section, which contains two entries for 'Infura' with the URL 'mainnet.infura.io'. The second entry is highlighted in blue and has a red text label 'I'm here!' next to it. Below the RPC URL section is a 'Currency symbol' field and a 'Save' button at the bottom.

- **Behind the Scenes: What Powers Your Wallet?**
 - Ethereum Client API is the most heavily trafficked surface
 - e.g., JSON-RPC methods such as `eth_getBalance`

What if the API went wrong?

Fetches with API:
`debug_transaction`

Fetches with API:
`trace_transaction`

Address	Before	After	State Difference
▼ 0x897a96De...2A019f38e	0.1 Eth	0 Eth	▼ 0.1

(a) Query the transfer value of a specific transaction on Etherscan; the value shown in the *State Difference* section is 0.1 ETH.

☰ The contract call From 0xc13CeD13...16a2f9B8B To 0x897a96De...2A019f38e produced 1 Internal Transaction ⓘ ADVANCED MODE: ☐						
Type	Trace	Address	From	To	Value	Gas Limit
✓ call_0_1	📄 0x897a96De...2A019f38e 📄	→ 0xc13CeD13...16a2f9... 📄	0.01 ETH	2,300		

(b) Query the same transaction on Etherscan; the value shown in the *Internal Transactions* section and the real transfer value on the chain is 0.01 ETH.

- **Same tx, different values:** Real-world client API bug [6] on the famous Ethereum explorer Etherscan, found by Geth team
 - 0.1 ETH in a webpage
 - 0.01 ETH in another webpage

What if the API went wrong?



Address	Before	After	State Difference
0x897a96De...2A019f38e	0.1 Eth	0 Eth	▼ 0.1

(a) Query the transfer value of a specific transaction on Etherscan; the value shown in the *State Difference* section is 0.1 ETH.

The contract call From [0xc13CeD13...16a2f9B8B](#) To [0x897a96De...2A019f38e](#) produced 1 Internal Transaction

Type	Trace Address	From	To	Value	Gas Limit
call_0_1		0x897a96De...2A019f38e	0xc13CeD13...16a2f9...	0.01 ETH	2,300

(b) Query the same transaction on Etherscan; the value shown in the *Internal Transactions* section and the real transfer value on the chain is 0.01 ETH.

- **Root cause:** Etherscan relies on a faulty Ethereum client API provided by Erigon
 - debug_transaction & trace_transaction should behave alike
- Reth also has a similar bug

How to find these bugs?

Challenge

- **Generate test requests for the APIs**
 - Fortunately, we have *comprehensive* specs
 - Unfortunately, they are overly *complicated*
- **Compare test results and find bugs**
 - Non-Deterministic environment noise
 - Semantically equivalent but statically different responses

- `eth_accounts`
- `eth_blobBaseFee`
- `eth_blockNumber`
- `eth_call`
- `eth_chainId`
- `eth_coinbase`
- `eth_createAccessList`
- `eth_estimateGas`
- `eth_feeHistory`
- `eth_gasPrice`
- `eth_getBalance`
- `eth_getBlockAccessList`
- `eth_getBlockByHash`
- `eth_getBlockByNumber`
- `eth_getBlockReceipts`
- `eth_getBlockTransactionCountByHash`
- `eth_getBlockTransactionCountByNumber`

[7]

Challenge #1 & Solution

- Automatically generating test inputs for various Ethereum client API

```
1 {
2   "name": "eth_getBalance",
3   "summary": "Returns the balance of the account of given address.",
4   "params": [ {
5     "name": "Address", "required": true,
6     "schema": {
7       "title": "hex encoded address", "type": "string",
8       "pattern": "^0x[0-9a-fA-F]{40}$" } }, {
9     "name": "Block", "required": true,
10    "schema": {
11      "title": "Block number, tag, or block hash",
12      "anyOf": [ {
13        "title": "Block number", "type": "string",
14        "pattern": "^0x(0|[1-9a-f][0-9a-f]*)$" }, {
15        "title": "Block tag", "type": "string",
16        "enum": ["earliest", "finalized", "safe", "latest", "pending"],
17        "description": "...", }, {
18        "title": "Block hash", "type": "string",
19        "pattern": "^0x[0-9a-f]{64}$" } ] } ],
20    "result": {
21      "name": "Balance",
22      "schema": {
23        "title": "hex encoded unsigned integer", "type": "string",
24        "pattern": "^0x(0|[1-9a-f][0-9a-f]*)$"
25      }
26      // This field is added by the LLM to filter out false positives
27      "x-consistency-policy", "must-identical" } }
28 }
```

- Test input: request to the API
 - API name and parameters
- Exiting solutions
 - EtherDiffer [8]: DSL-based test input generation, requires experts with domain knowledge to design the DSL
 - rpctestgen [9]: Hand-written tests used by developers to conduct conformance tests

Challenge #1 & Solution

- **Syntax-Oriented Generation**
 - Use the spec schema to generate syntax valid and invalid tests
- **Semantics-Aware Generation**
 - Leverage on-chain data to enrich request semantics
 - Mutation to improve diversity

```
{  
  "id": 1,  
  "jsonrpc": "2.0",  
  "method": "eth_getBalance",  
  "params": [  
    "\ud982\udd65": [ [], 10000000.0, true, "0xfe97" ],  
    "0xffffffff...ffffeeefffff" ]  
}
```

Syntactic-invalid case

```
{  
  "id": 1,  
  "jsonrpc": "2.0",  
  "method": "eth_getBalance",  
  "params": [  
    "0xffffffff...ffffeeefffff",  
    "latest"  
  ]  
}
```

Syntactic-valid case

```
{  
  "id": 1,  
  "jsonrpc": "2.0",  
  "method": "eth_getBalance",  
  "params": [  
    "0x9411d2eb...4EBb1c8ADA7B",  
    "latest"  
  ]  
}
```

Syntactic-valid and semantic-valid case

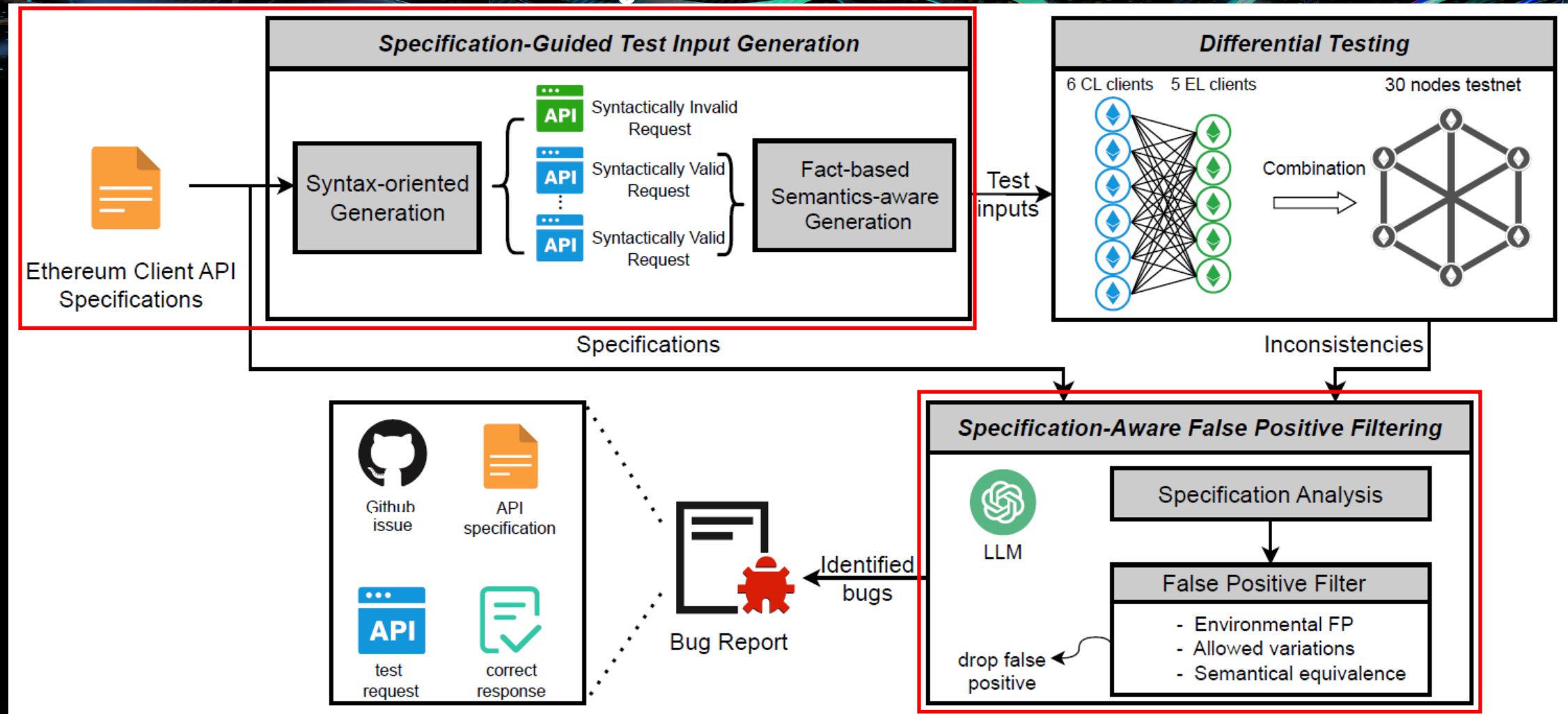
Challenge #2 & Solution

- **Distinguishing real bugs from false positives**
 - Differences in testing environments and node states
 - Allowed inconsistencies in response
 - `peer_id`, corresponding to node public key
 - Semantically equivalent, syntactically different responses
 - e.g., “Unable to decode data” vs “could not decode request body into...”
- **Solution**
 - Heuristic-based filtering
 - LLM as a judge
 - must-identical, may-divergent, must-divergent



Workflow of APIDiffer

For Challenge 1



For Challenge 2



Part IV: Bug Dissection

Discovered Bugs Overview

EVM bugs

API bugs

No.	Project	State	Root Cause	Opcode	Qty	Issue	CNVD	No.	Type	Project	Unk.	Sta.	Root cause	No.	Type	Project	Unk.	Sta.	Root cause
1	Geth	Fixed	Network fork	PUSH0	6	#29782	○	1	CL Client	Lighthouse	✓	●	Missing field in the API response	37	CL Client	Lodestar	✗	●	Incorrect encoding in API responses
2		Fixed	Instruction set	PREVRANDAO	1	#29722	●	2			✓	●	Missing field in the API response	38			✗	●	Error handling for invalid parameter
3		Fixed	Address setting	BALANCE	1	#30254	○	3			✓	●	Missing API implementation in the spec	39			✓	●	API fails to parse input parameter
4		Fixed	Fee setting	BASEFEE	1	#30279	○	4			✗	●	Missing API implementation in the spec	40			✓	●	API use wrong data type from old fork
5	Besu	Fixed	Gaslimit setting	GASLIMIT	1	#7432	○	5		✓	●	API fails to parse input parameter	41	Grandine		✓	●	Missing API implementation in the spec	
6		Fixed	Fee setting	BASEFEE	1	#7433	○	6		✓	●	Missing API implementation in the spec	42			✓	●	Missing API implementation in the spec	
7		Fixed	Tracing	REVERT	1	#7608	○	7		✓	●	Error handling for empty slashing	43			✓	●	Missing API implementation in the spec	
8		Fixed	Entry point	N/A	N/A	#7430	○	8		✓	●	Error handling for invalid parameter	44			✓	⊙	Error handling for invalid parameter	
9	Nethermind	Confirmed	Instruction set	PUSH0	6	#7629	○	9		✓	●	Missing field in the API response	45	EL Client	Geth	✓	●	Extra field in the API response	
10	EthereumJS	Confirmed	Address setting	N/A	N/A	#3236	●	10		✓	●	Incorrect field in the API response	46			✓	○	Handling of edge cases in API response values	
11		Will fix	Memory tracing	N/A	N/A	#3561	○	11		✓	●	Missing signature validation in API	47			✓	○	Inconsistent support for deprecated API	
12	Py-EVM	Fixed	Missing field	SELFDESTRUCT	1	#2176	●	12		✗	●	Missing API implementation in the spec	48		Nethermind	✓	●	Handling of edge cases in API parameter	
13		Reported	Fee setting	N/A	N/A	#2162	●	13		✓	●	Error handling for invalid parameter	49			✓	●	Stack overflow during evm execution	
14	revm	Confirmed	Fee setting	N/A	N/A	#1702	○	14		✓	●	Incorrect initialization of the chain	50			✓	●	Error handling for non-existent blocks	
15		Reported	Address setting	N/A	N/A	#1704	○	15		✓	●	Valid parameters are not supported	51			✗	⊙	Deprecated field in the API response	
16	EELS	Reported	Memory	N/A	N/A	#981	○	16		Teku	✓	●	Error handling for invalid parameter		52	Besu	✓	●	Missing check for Transaction
17		Reported	Description	N/A	N/A	#992	○	17			✓	●	Missing API implementation in the spec		53		✓	●	Type conversion error of filter_id
18	Will fix	Fee setting	N/A	N/A	N/A	#18	○	18			✗	○	Missing API implementation in the spec		54		✓	●	Handling of edge cases in API response values
19		Address setting	N/A	N/A	N/A	N/A	#19	○			19	✓	○	Wrong type in the API response	55		Erigon	✗	○
20	EELS	Reported	Memory	N/A	N/A	#981	○	20			✓	●	Fail to parse valid parameter	56	✓	○		Error handling for invalid parameter	
21		Reported	Description	N/A	N/A	#992	○	21			✓	●	Schema definition error	57	✓	●		Error handling for invalid parameter	
22	EELS	Reported	Memory	N/A	N/A	#981	○	22			Erigon	✓	●	Missing field in the API response	58	Erigon		✓	○
23		Reported	Description	N/A	N/A	#992	○	23				✓	●	Missing field in the API response	59		✓	●	Method handler crashed for nil
24	Will fix	Fee setting	N/A	N/A	N/A	#18	○	24				✓	●	Missing field in the API response	60		✓	●	Handling of edge cases in API response values
25		Address setting	N/A	N/A	N/A	N/A	#19	○				25	✓	●	Missing API implementation in the spec		61	✓	●

- 98 bugs within the EVM and API components of Ethereum clients
 - 26 EVM bugs, 72 API bugs, 86.73% (85/98) bugs confirmed or fixed
 - All 11 EL and CL clients in the Ethereum mainnet are affected

Case Study 1: EVM bug

- **Vulnerability:** SIGSEGV panic of opcode PREVRANDAO [10]
- **Root cause:** The pointer was not initialized and missing nil check for Random in EVM

```
func opRandom(pc *uint64, evm *EVM, scope *ScopeContext) ([]byte,
error) {
    v := new(uint256.Int).SetBytes(evm.Context.Random.Bytes())
    scope.Stack.push(v)
    return nil, nil
}
```

Case Study 1: EVM bug

Empty Random field since Shanghai fork #290

Closed

#289

opened on Aug 5, 2024 · edited by txhsl

Edits

Contributor

...

System information

Geth version: v0.2.1

OS & Version: Windows/Linux/OSX

Expected behaviour

Safe opcode handling in EVM.

Actual behaviour

After `The Merge`, there is an extra payload from beacon chain which contains the following attributes.

[go-ethereum/beacon/engine/types.go](https://github.com/go-ethereum/beacon/engine/types.go)

Lines 41 to 49 in a8a8758

```
41 // PayloadAttributes describes the environment context in which a block should
42 // be built.
43 type PayloadAttributes struct {
44     Timestamp      uint64      `json:"timestamp" gencodec:"required"`
45     Random          common.Hash `json:"prevRandao" gencodec:"required"`
46     SuggestedFeeRecipient common.Address `json:"suggestedFeeRecipient" gencodec:"required"`
47     Withdrawals     []*types.Withdrawal `json:"withdrawals"`
48     BeaconRoot      *common.Hash `json:"parentBeaconBlockRoot"`
49 }
```

The `Withdrawals` data is now handled by dBFT in [#122](#), but the field `Random` is left empty in EVM context.

Since Solidity 0.8.18, the `block.difficulty` is replaced by `block.prevrandao`, and the op `opDifficulty` is replaced by `opRandom`. However, the context `Random` is `nil`, so the EVM panics and closes the node.

Impact

- Denial-of-Service, Geth/evm will crash when executing PREVRANDAO
- Bug Propagation to Downstream Forks: Neo X node shutdown [11]

the context Random is nil, so the EVM panics and closes the node.

Case Study 2: API bug

- **Bug:** Besu returns payload status (INVALID) instead of JSON-RPC Error (-32602) for malformed execution requests [12]

besu response

```
{"id": 1, "jsonrpc": "2.0", "result": {"latestValidHash": null, "status": "INVALID", "validationError": "..."} }
```

geth response

```
{"error": {"code": -32602, "data": {"..."}}, "message": "Invalid parameters", "id": 1, "jsonrpc": "2.0"} }
```

neth response

```
{"error": {"code": -32602, "message": "Execution requests must not..."}, "id": 1, "jsonrpc": "2.0"} }
```

reth response

```
{"error": {"code": -32602, "data": {"..."}}, "message": "Invalid params", "id": 1, "jsonrpc": "2.0"} }
```

erigon response

```
{"error": {"code": -32602, "message": "Invalid Request at index 2"}, "id": 1, "jsonrpc": "2.0"} }
```

Case Study 2: API bug

- **Root cause:** Misrouted exception handling

SPEC

*If any element is out of order, has a length of 1-byte or shorter, or more than one element has the same type byte, or the param is null, client software **MUST** return -32602: Invalid params error.*

```
final Optional<List<Request>> maybeRequests;  
try {  
    maybeRequests = extractRequests(maybeRequestsParam);  
} catch (RequestType.InvalidRequestTypeException ex) {  
    // BUG: returns payload status INVALID instead of JSON-RPC error  
    return respondWithInvalid(  
        reqId, blockParam, ..., INVALID, "Invalid execution requests");  
} catch (Exception ex) {  
    return new JsonRpcErrorResponse(reqId, RpcErrorType.INVALID_EXECUTION_REQUESTS_PARAMS);  
}
```

Case Study 2: API bug

- **Impact**

- Consensus Confusion: Misleads the CL
- Diversity Risk: Besu is the sole outlier; all other major EL clients return -32602
- The bug was brought to Ethereum PM RPC Standard Meeting for discussion, driving further cross-client specification refinements [13]

✓ Closed

RPC Standards #7 | June 23rd, 2025 #1585

- Proposed action: Add json schemas to cover requests
 - Discussion: can these be derived from the yaml schemas?
 - Complication: where do they get executed?
 - build time validator of .io files against a json schema?
 - Do we need to add a schema validation layer to rpc-compat in hive?

Styleguide

- Empty search results: 200/null , 200/[], or -3200: Not Found?
 - best way to find these inconsistencies across the spec and start to align them?
- Ordering of error reporting - specifying error reporting often [impacts](#) [logic flow](#) from clients.

Ordering of error reporting - specifying error reporting often impacts logic flow from clients

Case Study 3: Spec bug

```
1 {
2   "body": {
3     "randao_reveal": "...",
4     "...": "...",
5     "deposits": {
6       "proof": {
7         "...": "...",
8         // Bug here: Both values should be 33
9         "minItems": 32,
10        "maxItems": 32
11      },
12      "data": "..."
13    },
14    "execution_payload": {
15      "parent_hash": "...",
16      "...": "...",
17      // Bug here: Missing definition
18      // "blob_gas_used": "...",
19      "block_hash": "..."
20    }
21  }
22 }
```

- **Bug:** Wrong definitions in CL Beacon-API specification [14]
- **Root cause**
 - Missing fields: blob_gas_used
 - introduced in 03/2024
 - remain absent in the spec as of 03/2025
 - Wrong value: proof size
 - defined as 33 in consensus-spec ✓
 - defined as 32 in the beacon-api spec ✗

Case Study 3: Spec bug



- **Impact**

- **Systemic Propagation**

- Spec-level bugs could propagate to any downstream clients or tools that trust the spec
 - Valid requests will be rejected
 - Tooling incapable of generating valid requests

Case Study 3: Spec bug

[Beacon-api] Fix discrepancies between schema definition and spec #9253

✓ Closed

opened on Mar 20, 2025

Contributor ...

[ethereum/beacon-APIs#515](#)

It's possible we're fine on both of these because of how we implement it now, but we should double check



we should double check



self-assigned this on Mar 20, 2025

- ## Mitigation

- Spec maintainer confirmed within 15 mins and pushed a fix PR in 1 Hour
- Developer from another CL client team review their implementation [15]



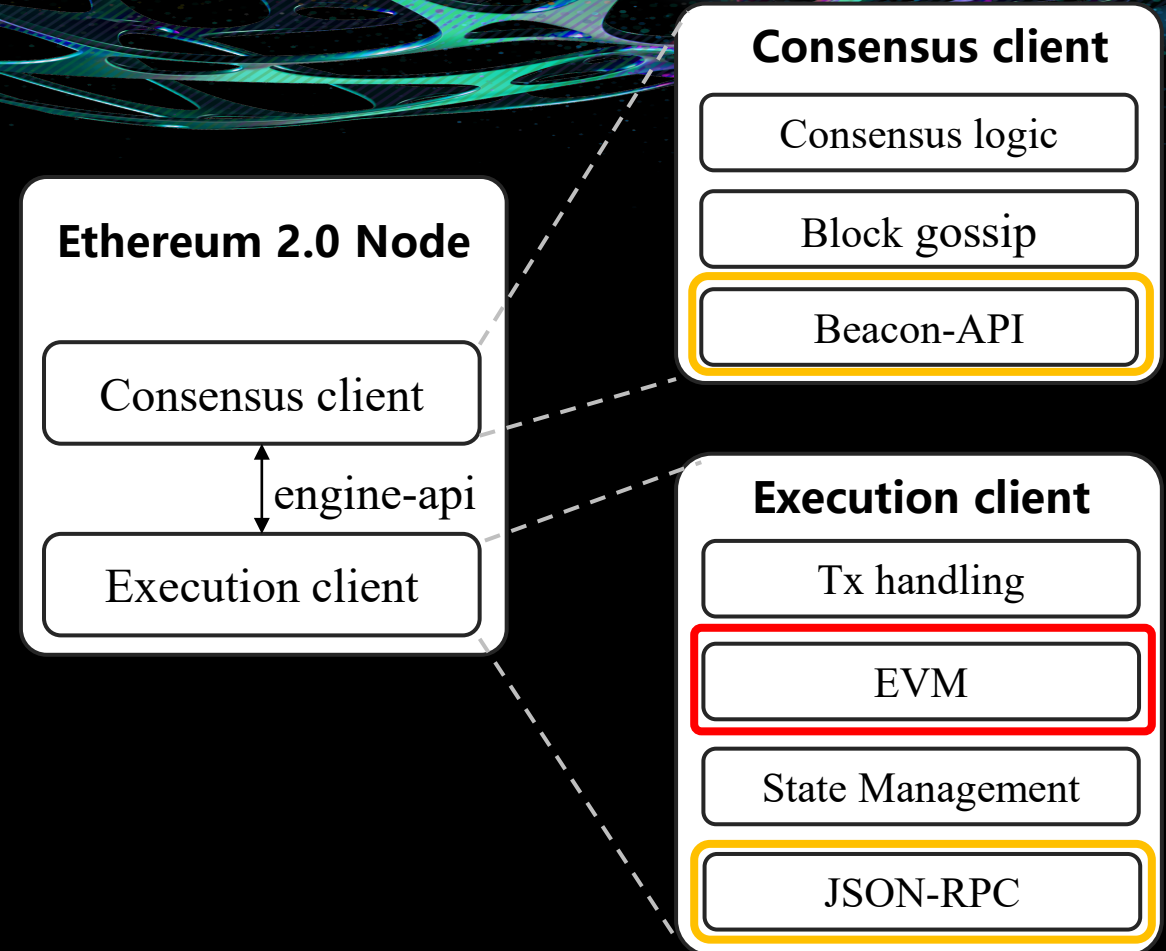
Part V: Implication & Conclusion

Implication

- **For Developers**
 - Identified real-world bugs with actionable reports
 - Detailed repro steps, spec mappings, and cross-client comparisons
- **For End Users**
 - Securing node/wallet interactions and enhancing user experience
- **For Researchers**
 - LLM-empowered differential testing approach for blockchains
- **For the Ethereum Community**
 - Safeguarding client diversity and interoperability by testing all clients

Implication

- **Other Attack Vectors**
 - Consensus Mechanism
 - State Storage & management
 - Transaction Pool
 - P2P Network
 - Gas Mechanism
 - Software Supply chain
 - ...



Conclusion

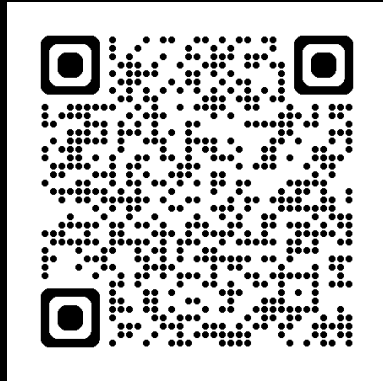
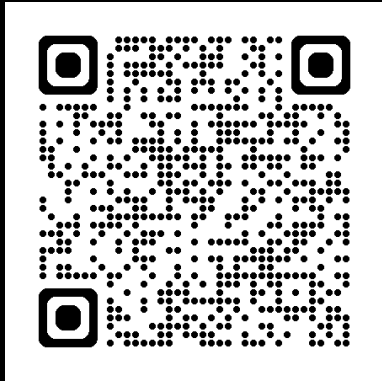
- **Prioritize Ethereum Client Security**
 - Client security is as critical as smart contract security
 - Client vulnerabilities can have far-reaching impacts
- **LLM-Empowered Differential Testing**
 - How to combine differential testing with LLMs to identify inconsistencies and bugs in Ethereum clients
- **Real-world Ethereum Client Bugs**
 - Uncovering 98 bugs across 11 mainstream Ethereum clients implementations



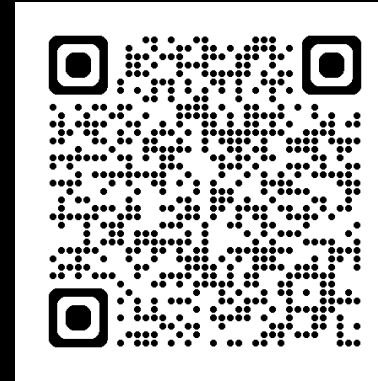
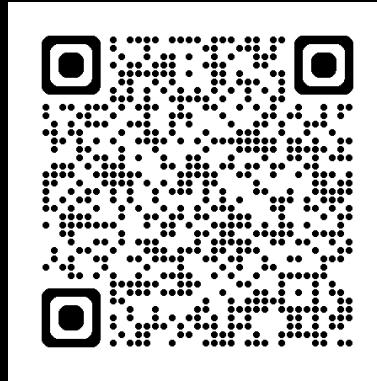
Takeaways

- **Prioritize Ethereum Client Security**
- **LLM-Empowered Differential Testing for Ethereum**
- **Real-world Ethereum Client Bugs**

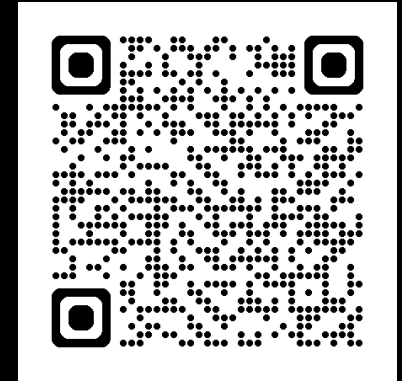
ISSTA'25 Paper & Artifact



OOPSLA'26 Paper & Artifact



Github





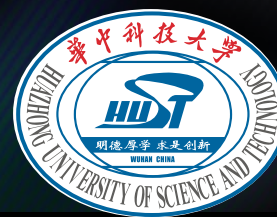
Fortifying the Foundation: LLM-Empowered Differential Testing for the Ethereum Infrastructure

Speakers: Jie Ma, Ningyu He

majie2023@buaa.edu.cn, ningyu.he@polyu.edu.hk



AMBER



ethereum
foundation

References

- [1] <https://www.certik.com/blog/hack3d-the-web3-security-report-2025>
- [2] <https://www.chainalysis.com/blog/crypto-hacking-stolen-funds-2026/>
- [3] <https://decrypt.co/47891/how-a-dormant-bug-briefly-split-the-ethereum-blockchain>
- [4] <https://coinalertnews.com/news/2025/12/14/41cc0eca>
- [5] <https://clientdiversity.org/>
- [6] <https://github.com/ethereum/go-ethereum/issues/30593>
- [7] <https://ethereum.github.io/execution-apis/>
- [8] [EtherDiffer: Differential Testing on RPC services of Ethereum Nodes\[C\]//Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering \(ESEC/FSE 2023\). 2023: 1333-1344.](#)
- [9] <https://github.com/lightclient/rpctestgen>
- [10] <https://github.com/ethereum/go-ethereum/pull/29799>
- [11] <https://github.com/bane-labs/go-ethereum/issues/290>
- [12] <https://github.com/besu-eth/besu/issues/8640>
- [13] <https://github.com/ethereum/pm/issues/1585>
- [14] <https://github.com/ethereum/beacon-APIs/issues/514>
- [15] <https://github.com/Consensys/teku/issues/9253>