Zero-Knowledge Bug Bounty Program

A Proof of Concept

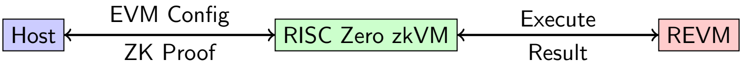
Antonio Viggiano

September 6, 2025

Problem

- **Trust Issues in Bug Bounty Programs**
  - Whitehats must reveal exploit details before payment
  - Protocols may not pay after learning the vulnerability
  - High risk for security researchers
- **Need for Privacy-Preserving Vulnerability Reporting**
  - Prove knowledge of an exploit without revealing it
  - Cryptographic guarantees for fair exchange
  - Minimize trust requirements between parties

Solution Architecture



- **Host**: Prepares bytecode and calldata, manages verification
- **zkVM**: Provides zero-knowledge execution environment
- **REVM**: Executes Ethereum Virtual Machine bytecode
- **Privacy**: Only hashes of inputs are revealed, not the exploit itself

Smart Contract Example

```solidity
// Contract.sol
pragma solidity ^0.8.26;

contract Contract {
    function isSolved(int256 x) external pure returns (bool)
     {
        // p(x) = x^4 - 84x^3 + 1765x^2 - 84x + 1764
        return ((x**4)
            - (84 * (x**3))
            + (1765 * (x**2))
            - (84 * x)
            + 1764) == 0;
    }
}
```

- Polynomial equation with specific roots
- Whitehat must find correct input value ($x = 42$)
- Function returns true for valid solutions

Host Implementation

```rust
fn main() {
    let config = EvmConfig::default();

    // Compute and display hashes for privacy
    let bytecode_hash = Keccak256::digest(config.get_bytecode());
    let calldata_hash = Keccak256::digest(config.get_calldata());

    println!("Executing bytecode hash: 0x{}",
        hex::encode(bytecode_hash));

    // Execute inside zkVM
    let (receipt, return_value) = execute_evm_bytecode(config);

    // Verify the proof
    receipt.verify(REVM_GUEST_ID).expect("Verification failed");

    println!("Receipt verified! Result: {:02x?}", return_value);
}
```

Guest (zkVM) Implementation

```rust
fn main() {
    let config: EvmConfig = env::read();

    // Create EVM with in-memory database
    let ctx = Context::mainnet().with_db(CacheDB::<EmptyDB>::default());
    let mut evm = ctx.build_mainnet();

    // Deploy contract
    let deploy_result = evm.transact_commit(
        TxEnv::builder()
            .kind(TxKind::Create)
            .data(Bytes::from(config.bytecode.clone()))
            .build().unwrap()
    ).unwrap();

    // Call function with private calldata
    let call_result = evm.transact_commit(/* ... */);

    // Commit only hashes, not raw data
    env::commit(&(bytecode_hash, calldata_hash, is_solved));
}
```

Privacy Features

- **Input Privacy**
  - Bytecode and calldata are hashed using Keccak256
  - Only hashes are committed to the proof journal
  - Original exploit details remain secret
- **Verifiable Execution**
  - RISC Zero provides cryptographic proof of correct execution
  - Anyone can verify the proof without trusting the prover
  - Result integrity is mathematically guaranteed
- **Zero-Knowledge Properties**
  - Proves knowledge of a valid exploit
  - Reveals nothing about the exploit method
  - Enables trustless bug bounty programs

Current Implementation (Version 0)

- **Proof of Concept**
  - REVM successfully runs inside RISC Zero zkVM
  - Public bytecode with private calldata
  - Commits code/calldata hashes and boolean outcome
- **Key Components**
  - Solidity compiler integration (build.rs)
  - ABI encoding for function calls
  - Hash-based privacy preservation
  - Automated testing framework
- **Verification Process**
  - Host generates proof of EVM execution
  - Proof can be verified by any third party
  - Mathematical guarantee of correctness

Next Steps (Roadmap)

- **Version 1: Real Chain State**
  - Anchor execution to real blockchain state
  - Witnessed snapshot at specific block height
  - Merkle Patricia Trie verification
  - Fully explicit, deterministic TxEnv
- **Version 2: On-Chain Integration**
  - On-chain verification via RISC Zero verifier
  - Predicate registry for vulnerability patterns
  - Escrow vault for bounty payments
  - Pay-to-reveal hash timelocked contracts

Questions?

Thank you for your attention

- Built as part of EF Core Program Brazil 2025
- Demonstrates ZK-EVM capabilities in RISC Zero
- Foundation for trustless bug bounty programs