The Polyverse Ledger - Comprehensive Test & Vulnerability Report

Project Overview

Project Name: The-Polyverse-Ledger

Repository: https://github.com/PJDEEPESH/The-Polyverse-Ledger

Summary:

This report outlines the entire process undertaken to understand, set up, test, and analyze smart contracts for potential vulnerabilities in The-Polyverse-Ledger project. It includes our test case implementations, detailed vulnerability findings, setup procedures, CLI commands, and a thorough summary of insights gathered.

1. Setup and Environment

Steps followed for local setup:

- Clone the repository:

```
git clone https://github.com/PJDEEPESH/The-Polyverse-Ledger.git
cd The-Polyverse-Ledger
```

- Install dependencies:

```
npm install
```

- Run Hardhat node:

```
npx hardhat node
```

- Compile contracts:

```
npx hardhat compile
```

- Deploy contracts locally:

```
npx hardhat run scripts/deploy.js --network localhost
```

- Run tests:

```
npx hardhat test
```

2. Smart Contract Tests Implemented

We wrote unit tests using Mocha/Chai in JavaScript under the test/ directory. Below are test snippets for the key contracts.

CreditScore.sol

3. Vulnerability Findings

UserRegistry.sol

Vulnerability: No checks on duplicate user registration

- Issue: Allows the same user to register multiple times without constraints.

- Impact: Could lead to bloated data and identity inconsistencies.

- Fix Suggestion:

```
require(!isRegistered[msg.sender], "User already registered");
```

General JSON-RPC parse error:

Error Encountered:

Parse error: Unexpected end of JSON input

This was caused by malformed JSON RPC requests and was resolved by ensuring well-formatted calls or using frontend libraries like Web3.js properly.

4. Screenshot:

Included error screenshot demonstrates malformed JSON RPC error on Ganache.

5. Additional Observations:

- Contract deployment was seamless on Hardhat.
- Tests passed successfully.
- Good logical structure in contracts.
- Improvement suggestion: add duplicate checks, role checks.

6. Commands Used:

```
npx hardhat compile
npx hardhat node
npx hardhat run scripts/deploy.js --network localhost
npx hardhat test
```

7. Learnings and Takeaways:

- Deepened understanding of Solidity and smart contract deployment.
- Used Hardhat effectively for compiling, testing, debugging.
- Analyzed security risks in contract logic.
- Realized importance of test coverage, validation, and input constraints.

8. Summary:

All major functionalities tested. Vulnerability in UserRegistry logged and recommendations made.

Contracts are modular, testable, and mostly secure under valid conditions.