Fuzzing with Echidna

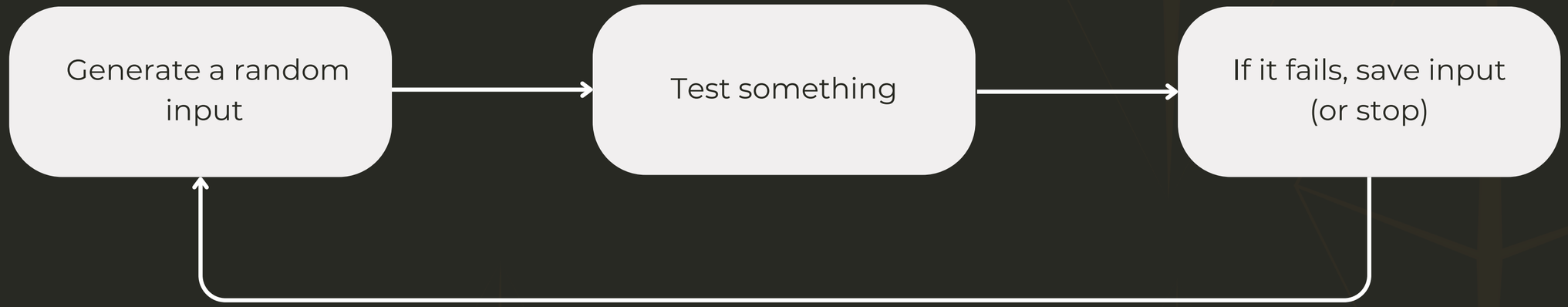
- What is fuzzing
- Defining invariants
- External vs Internal testing
- Debugging with coverage reports

WHAT IS FUZZING

4 main testing techniques

- Unit testing
- Manual analysis
- Fully automated analysis
- Semi-automated analysis

WHAT IS FUZZING



WHAT IS FUZZING

- **Fuzzing** is feeding a piece of code (function, program, etc.) data from a large corpus, possibly dynamically generated, possibly dependent on the results of execution on previous data, in order to see whether it fails.
- **Property based testing** is the construction of tests such that, when these tests are fuzzed, failures in the test reveal problems with the system under test that could not have been revealed by direct fuzzing of that system.
  - **A property-based testing library** has two parts:
    - A fuzzer.
    - A library of tools for making it easy to construct property-based tests using that fuzzer.
  -

WHAT IS FUZZING

Echidna aims to break user-defined invariants

- In smart contracts, invariants are Solidity functions that can represent any incorrect or invalid state that the contract can reach, including:
  - Incorrect access control: The attacker becomes the owner of the contract.
  - Incorrect state machine: Tokens can be transferred while the contract is paused.
  - Incorrect arithmetic: The user can underflow their balance and get unlimited free tokens.

WHAT IS FUZZING

Echidna inputs: target contracts + properties to test

OXORIO

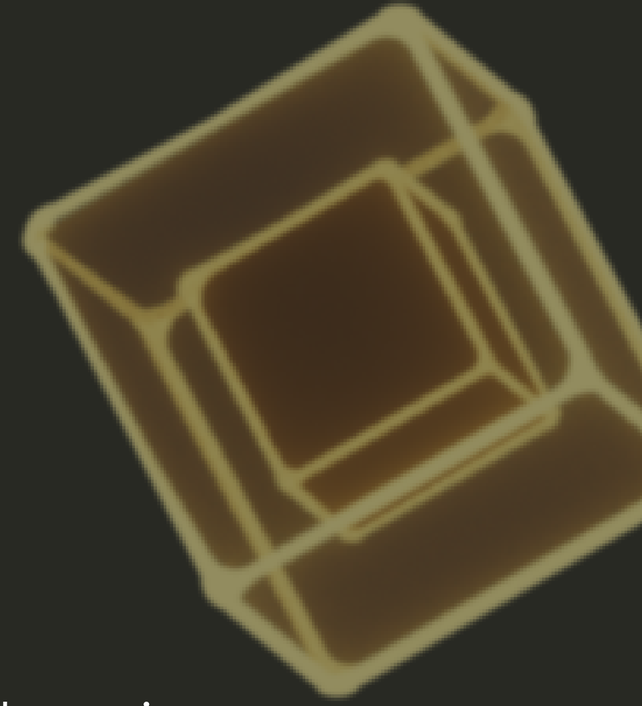
DEFINING INVARIANTS

Defining good invariants

- Start small and iterate
- Steps

1. Define invariant in English
2. Write the invariants in solidity
3. Run Echidna
    - If invariants broken: investigate
    - Once all invariants pass, go back to step 1

DEFINING INVARIANTS



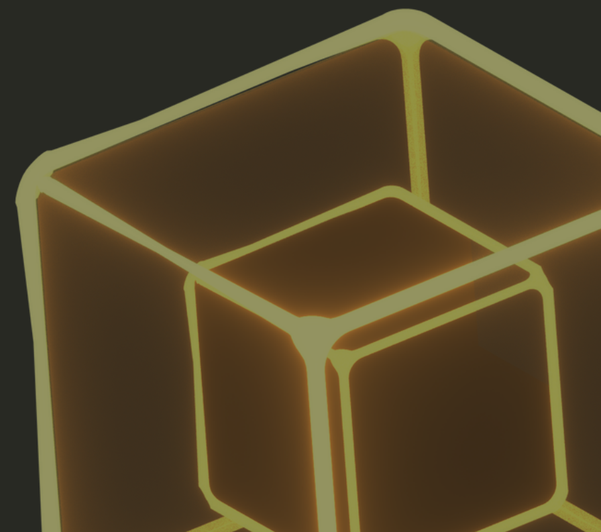
Two types of invariants

- **Function-level invariants**
  - Doesn’t **rely much on the system** OR could be **stateless**
  - Can be tested in an **isolated** fasion
  - Examples: Assotiative property of addition OR depositing tokens in a contract
- **System-level invariants**
  - Relies on the **deployment** of a **large part or the entire system**
  - Invariants are usually stateful
  - Examples: user’s balance < total supply OR yield is monotonically increasing

DEFINING INVARIANTS

Testing system level invariants require initialization

- **Simple initialization**
  - Deploy everything in the constructor
- **Complex initialization**
  - Leverage unit-test framework with Etheno



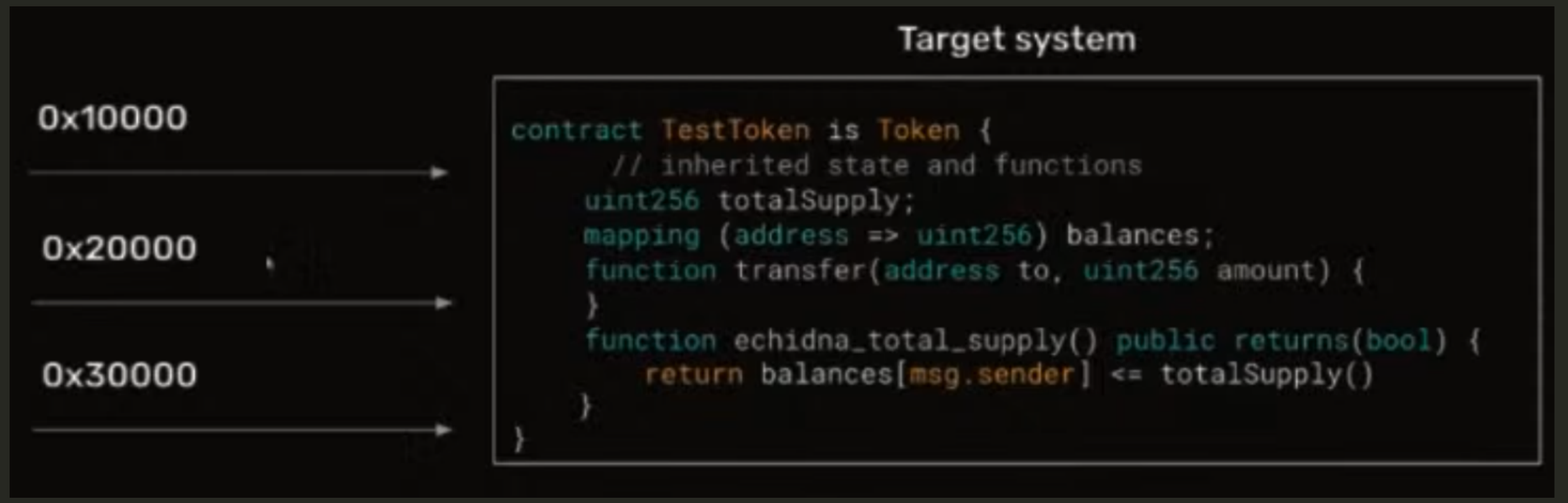
EXTERNAL VS INTERNAL TESTING

What is internal testing?

- **It uses inheritance to test the target contract**
- **Pros**
  - Easy to setup
  - Get the state all public/external functions of the inherited contract
  - msg.sender is preserver
- **Cons**
  - Not good for complex systems
  - Mostly viable for single-entrypoint systems

EXTERNAL VS INTERNAL TESTING

Visualizing internal testing



EXTERNAL VS INTERNAL TESTING

What is external testing?

- **Uses external calls to the target system**
- **Pros**
  - Good for complex systems with complex initialization
  - Good for multi-entrypoint systems
  - Mostly used in practice
- **Cons**
  - Difficult to setup
  - msg.sender is not preserved

EXTERNAL VS INTERNAL TESTING

Visualizing external testing

OXORIO

DEBUGGING WITH COVERAGE REPORTS

- **Coverage is the testing of what code was “touched” by the fuzzer**
- **How to read coverage report**
  - ***: Execution ended with STOP**
    - At some point, this line was executed with no errors
  - **r: Execution ended with REVERT**
    - At some point, this line caused transaction to revert
  - **o: out-of-gas error**
    - Common with loops
  - **e: Execution ended with any other error**
    - E.g. zero division

DEBUGGING WITH COVERAGE REPORTS

OXORIO

- **Coverage report is a crucial debugging tool that will greately improve the testing effort**
- **Provides a guarantee that the tests ran as expected**
- **You should not run Echidna without coverage enabled**
  - CLI: --corpus-dir <directory-name>
  - Config file: corpusDir: <directory-name>
-

THANK YOU

OXGRIO

CONTACTS

OXOR.IO

WEI@OXOR.IO

PUBLIC AUDITS

@0XORIO

WORK WITH US!

MORE IN OUR BLOG:

blog.oxor.io

OXORIO