ConFuzius: A Data Dependency-Aware Hybrid Fuzzer for Smart Contracts

Christof Ferreira Torres, Antonio Ken Iannillo, Arthur Gervais and Radu State
Attacks on Smart Contracts

A $50 million hack just showed that the DAO was all too human.

A hacker stole $31M of Ether — how it happened, and what it means for Ethereum.

SpankChain, a cryptocurrency project focused on the adult industry, has suffered a breach that saw almost $40,000 in ethereum (ETH) stolen.

Wallet bug freezes more than $150 million worth of Ethereum.


by Alina Bizga on April 21, 2020
Smart Contract Security Challenges

```solidity
contract TokenSale {
    uint256 start = now;
    uint256 end = now + 30 days;
    address wallet = 0xcafebabe...;
    Token token = Token(0x12345678...);

    address owner;
    bool sold;

    function Tokensale() public {
        owner = msg.sender;
    }

    function buy() public payable {
        require(now < end);
        require(msg.value == 42 ether + (now - start) / 60 / 60 / 24 * 1 ether);
        require(token.transferFrom(this, msg.sender, token.allowance(wallet, this)));
        sold = true;
    }

    function withdraw() public {
        require(msg.sender == owner);
        require(now >= end);
        require(sold);
        owner.transfer(address(this).balance);
    }
}
```

Stateful exploration
Solution: read-after-write data dependency

Input generation
Solution: Input generation via symbolic execution

Environmental dependency
Solution: Model and fuzz environmental inputs
Architecture

ConFuzzius

Evolutionary Fuzzing Engine

- Initial Population
- Selection
- Crossover
- Mutation

Data Dependencies

- Read Value
- Add Value
- Remove Value

Mutation Pools

Individual

Instrumented EVM

Execution Trace

Execution Trace Analyzer

- Code Coverage Evaluation
- Data Dependency Analysis
- Constraint Solving
- Termination Analysis
- Symbolic Taint Analysis

- Vulnerability Detection

Smart Contract

Blockchain State (Optional)

Report
Encoding Individuals

Environment

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>1533907326</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Number</td>
<td>9867543</td>
</tr>
<tr>
<td>Call Result</td>
<td>...</td>
</tr>
<tr>
<td>Return Data Size</td>
<td>...</td>
</tr>
<tr>
<td>External Code Size</td>
<td>...</td>
</tr>
</tbody>
</table>

Transaction

<table>
<thead>
<tr>
<th>From</th>
<th>0xdead_beef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>100</td>
</tr>
<tr>
<td>Gas Limit</td>
<td>300000</td>
</tr>
<tr>
<td>Data</td>
<td>...</td>
</tr>
</tbody>
</table>

Contract Address

<table>
<thead>
<tr>
<th>0x1234_5678</th>
<th>32</th>
</tr>
</thead>
</table>

Function Selector

<table>
<thead>
<tr>
<th>0x7d6cd25</th>
<th>&quot;hello&quot;</th>
<th>42</th>
</tr>
</thead>
</table>

String

<table>
<thead>
<tr>
<th>&quot;hello&quot;</th>
<th>Uint32</th>
</tr>
</thead>
</table>

External Code Size

<table>
<thead>
<tr>
<th>0x1234_5678</th>
<th>32</th>
</tr>
</thead>
</table>

Uint256

<table>
<thead>
<tr>
<th>0x1234_5678</th>
<th>32</th>
</tr>
</thead>
</table>

Individual

Population

Input
Access to state variables is retrieved dynamically via **SLOAD** and **SSTORE** instructions.

Retrieve storage variable from storage location:

- **Statically-sized:**
  - Includes: primitives, structs, and fixed arrays.
  - Pop first element from stack.

- **Dynamically-sized:**
  - Mappings: Map result of **SHA3** instruction to memory contents and only extract the last 32 bytes (concatenation).
  - Dynamic Arrays: keep track of arithmetic additions of **SHA3** hashes.

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Declaration</th>
<th>Access</th>
<th>Storage Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive</td>
<td>T v</td>
<td>v</td>
<td>s(v)</td>
</tr>
<tr>
<td>Struct</td>
<td>struct v { T a }</td>
<td>v.a</td>
<td>s(v) + s(a)</td>
</tr>
<tr>
<td>Fixed Array</td>
<td>T[10] v</td>
<td>v[n]</td>
<td>s(v) + n *</td>
</tr>
<tr>
<td>Dynamic Array</td>
<td>T[] v</td>
<td>v[n]</td>
<td>h(s(v)) + n *</td>
</tr>
<tr>
<td></td>
<td>v.length</td>
<td>s(v)</td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>mapping(T1 =&gt; T2) v</td>
<td>v[k]</td>
<td>h(k</td>
</tr>
</tbody>
</table>
Evaluation
Datasets, Baselines, and Experimental Setup

Datasets

1. **Real-World dataset**
   - 21,147 contracts with source code from Etherscan.
   - Clustered using k-means into a large cluster (3,344 contracts) and small cluster (17,803 contracts).

2. **Curated dataset**
   - Based on **SmartBugs** by Durieux et al. and extended with 5 additional types from SWC registry.
   - 128 contracts with 148 annotated vulnerabilities across 10 different types.

Baselines

<table>
<thead>
<tr>
<th>Toolname</th>
<th>Type</th>
<th>Requires Source Code</th>
<th>Requires ABI</th>
<th>AF</th>
<th>IO</th>
<th>RE</th>
<th>TD</th>
<th>BD</th>
<th>UE</th>
<th>UD</th>
<th>LE</th>
<th>LO</th>
<th>US</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTENTE [28]</td>
<td>Symbolic</td>
<td>×</td>
<td>×</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>MYTHRIL [31]</td>
<td>Symbolic</td>
<td>×</td>
<td>×</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>M-PRO [31]</td>
<td>Symbolic</td>
<td>✓</td>
<td>×</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>ILF [18]</td>
<td>Fuzzer</td>
<td>✓</td>
<td>✓</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>SFUZZ [32]</td>
<td>Fuzzer</td>
<td>✓</td>
<td>✓</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td></td>
</tr>
<tr>
<td>CONFUZZIUS</td>
<td>Hybrid</td>
<td>✓</td>
<td>✓</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
</tr>
</tbody>
</table>

Experimental Setup

10 runs with independent seeds with 10 min for small contracts and 1 hour for large contracts.
RQ1: Does ConFuziius achieve higher code coverage than current state-of-the-art symbolic execution and fuzzing tools for smart contracts?
ConFuzzius achieves highest code coverage:

- 91% small contracts and 81% large contracts
- Every tool struggles with large contracts
- 10% difference between small and large for ConFuzzius vs. 31% difference for symbolic execution tools (Mythril)
Code Coverage Over Time

- **ConFuzzius outperforms state-of-the-art fuzzers:**
  - **66% after 1 second** on small contracts vs. **12% ILF** and **15% sFuzz**
  - **46% after 1 second** on large contracts vs. **10% ILF** and **11% sFuzz**
RQ2: Does ConFuzzius discover more vulnerabilities than current state-of-the-art symbolic execution and fuzzing tools for smart contracts?
ConFuzzius detected most vulnerabilities (106/148).

All symbolic execution tools reported false positives, especially for integer overflows.

ConFuzzius does not report false positives.

ILF and sFuzz reported false positives for unsafe delegatecalls due to the imprecision of their detectors:

```solidity
function setCallee(address newCallee) {
    require(msg.sender == owner);
    callee = newCallee;
}

function forward(bytes _data) {
    require(callee.delegatecall(_data));
}
```
RQ3: How relevant are ConFuzzius’s individual components in terms of code coverage and vulnerability detection?
We randomly selected 100 contracts from each cluster and disabled each component separately.

Each component is an added value.

Constraint solving plays an essential part in coverage and vulnerability detection.

Environmental instrumentation is more relevant for detecting vulnerabilities.

Data dependency analysis allows to find 10% and 18% more vulnerabilities in small and large contracts, respectively.
Conclusion
Conclusion

- We presented ConFuzzius – the first data dependency-aware hybrid fuzzer for smart contracts that tackles the following three challenges:
  - **Input generation**: evolutionary fuzzing + constraint solving.
  - **Stateful exploration**: read-after-write access patterns across state variables.
  - **Environmental dependencies**: modelling block and contract information as fuzzable inputs.

- We compared ConFuzzius against 2 fuzzers and 3 symbolic execution tools using a curated dataset of 128 contracts and a dataset of 21K real-world contracts.

- ConFuzzius detects more bugs (up to 23%) and achieves higher code coverage (up to 69%).

- Our data dependency analysis can boost the detection of bugs (up to 18%).
All code & data is available on GitHub:

https://github.com/christoftorres/ConFuzzius

Contact information:

christof.torres@uni.lu

Supported by: