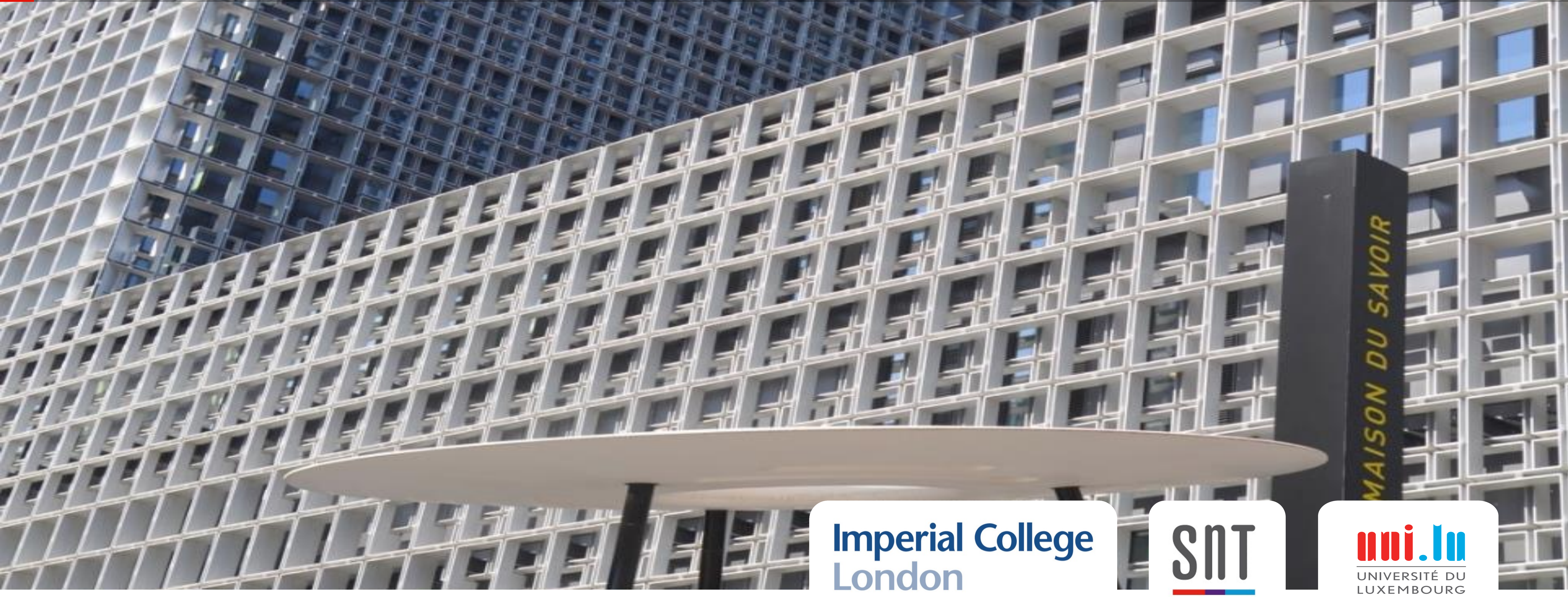
University of Luxembourg

Interdisciplinary Centre for Security, Reliability and Trust

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

Christof Ferreira Torres, Antonio Ken Iannillo, Arthur Gervais and Radu State



Imperial College London

SnT

uni.lu UNIVERSITÉ DU LUXEMBOURG

Attacks on Smart Contracts

KLINT FINLEY BUSINESS 06.18.16 04:30 AM

A $50 MILLION HACK JUST SHOWED THAT THE DAO WAS ALL TOO HUMAN

20 JULY 2017 / #ETHEREUM #BLOCKCHAIN #SECURITY

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

A hackers' dream payday: Ledf.Me and Uniswap lose $25 million worth of cryptocurrency

by Alina Bizga on April 21, 2020

Smart Contract Security Challenges

```
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);
  }
}
```

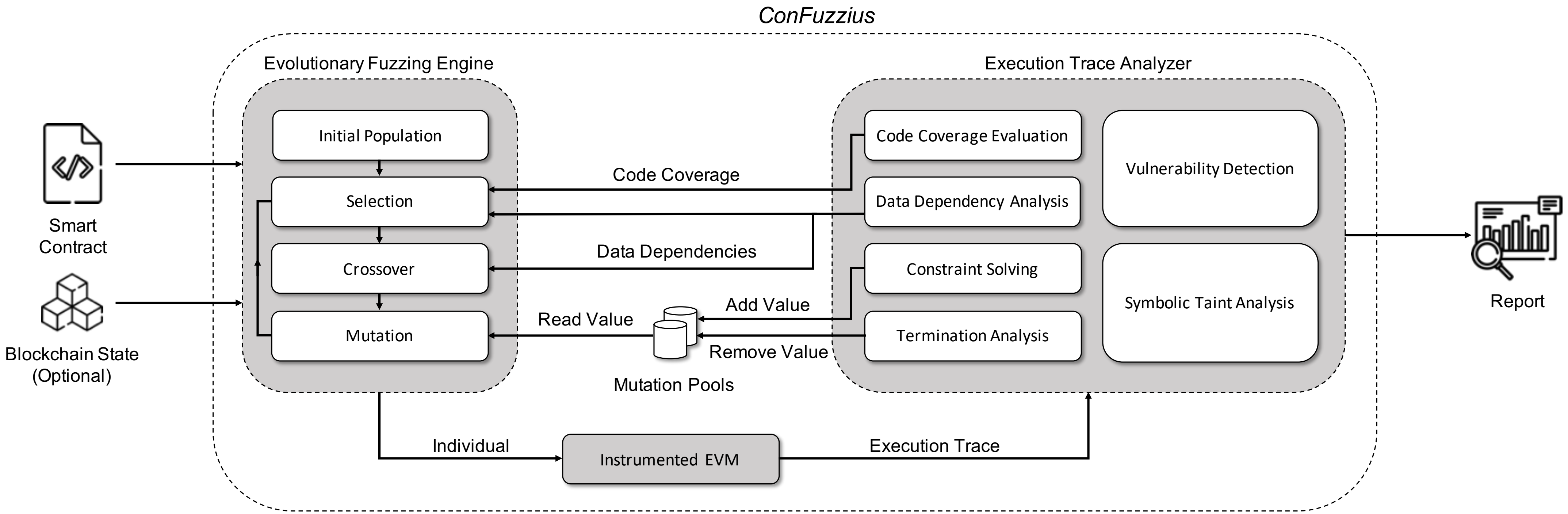
Stateful exploration
Solution: read-after-write data dependency

Input generation
Solution: Input generation via symbolic execution

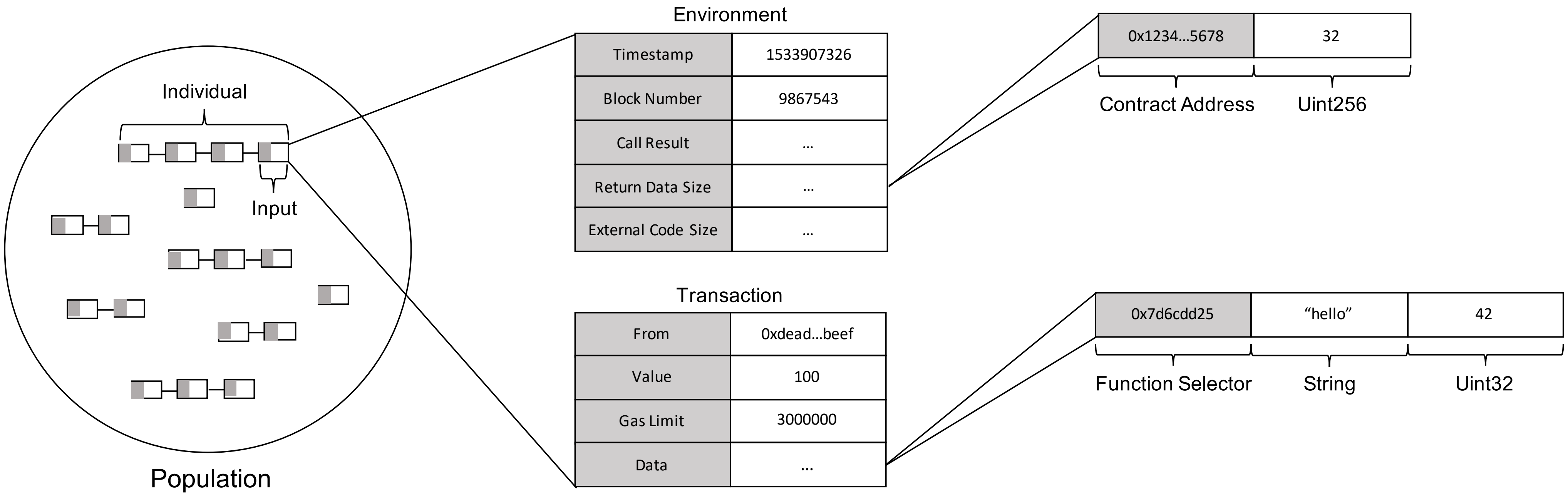
Environmental dependency
Solution: Model and fuzz environmental inputs

ConFuzzius

Architecture



Encoding Individuals



Data Dependency Analysis

- 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.

Variable Type	**Declaration**	**Access**	**Storage Location**
Primitive	`T v`	`v`	s(v)
Struct	`struct v { T a }`	`v.a`	s(v) + s(a)
Fixed Array	`T[10] v`	`v[n]`	$s(v) + n \cdot \lvert T \rvert$
Dynamic Array	`T[] v`	`v[n]`	$h(s(v)) + n \cdot \lvert T \rvert$
		`v.length`	s(v)
Mapping	`mapping(`T_1` => `T_2`) v`	`v[k]`	$h(k \,\Vert\, s(v))$

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

Toolname	Type	Requires Source Code	Requires ABI	Vulnerability Detectors AF	IO	RE	TD	BD	UE	UD	LE	LO	US
OYENTE [28]	Symbolic	✗	✗	●	●	●	●	●	○	○	○	○	●
MYTHRIL [31]	Symbolic	✗	✗	●	●	●	●	●	●	●	●	○	●
M-PRO [31]	Symbolic	✓	✗	●	●	●	●	●	●	●	●	○	●
ILF [18]	Fuzzer	✓	✓	○	○	○	○	●	●	●	●	●	●
sFUZZ [32]	Fuzzer	✓	✓	○	●	●	○	●	●	●	○	●	○
CONFUZZIUS	**Hybrid**	✗	✓	●	●	●	●	●	●	●	●	●	●

Experimental Setup

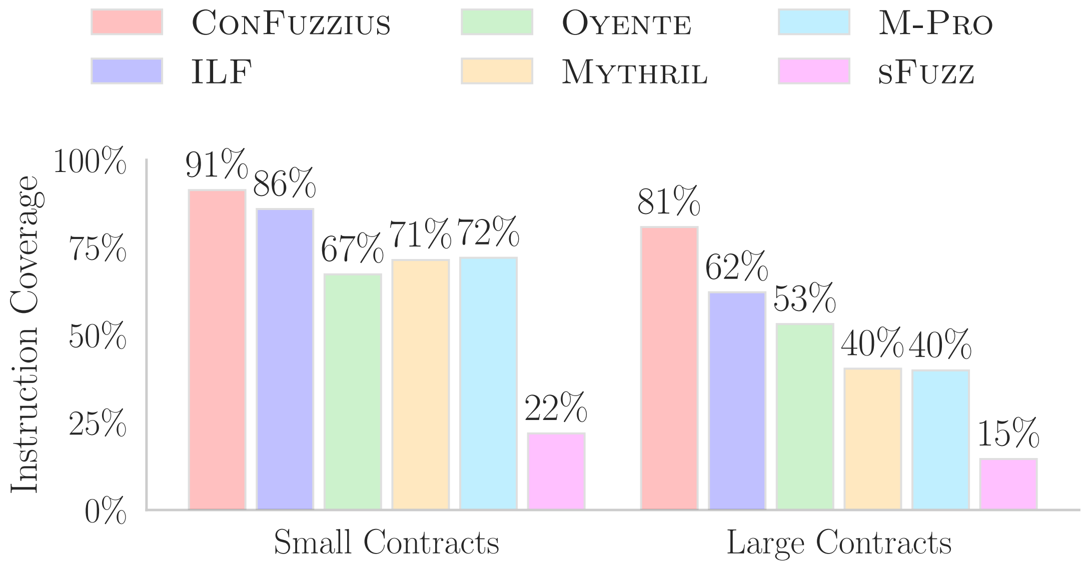
10 runs with **independent seeds** with **10 min for small** contracts and **1 hour for large** contracts.

1st Research Question

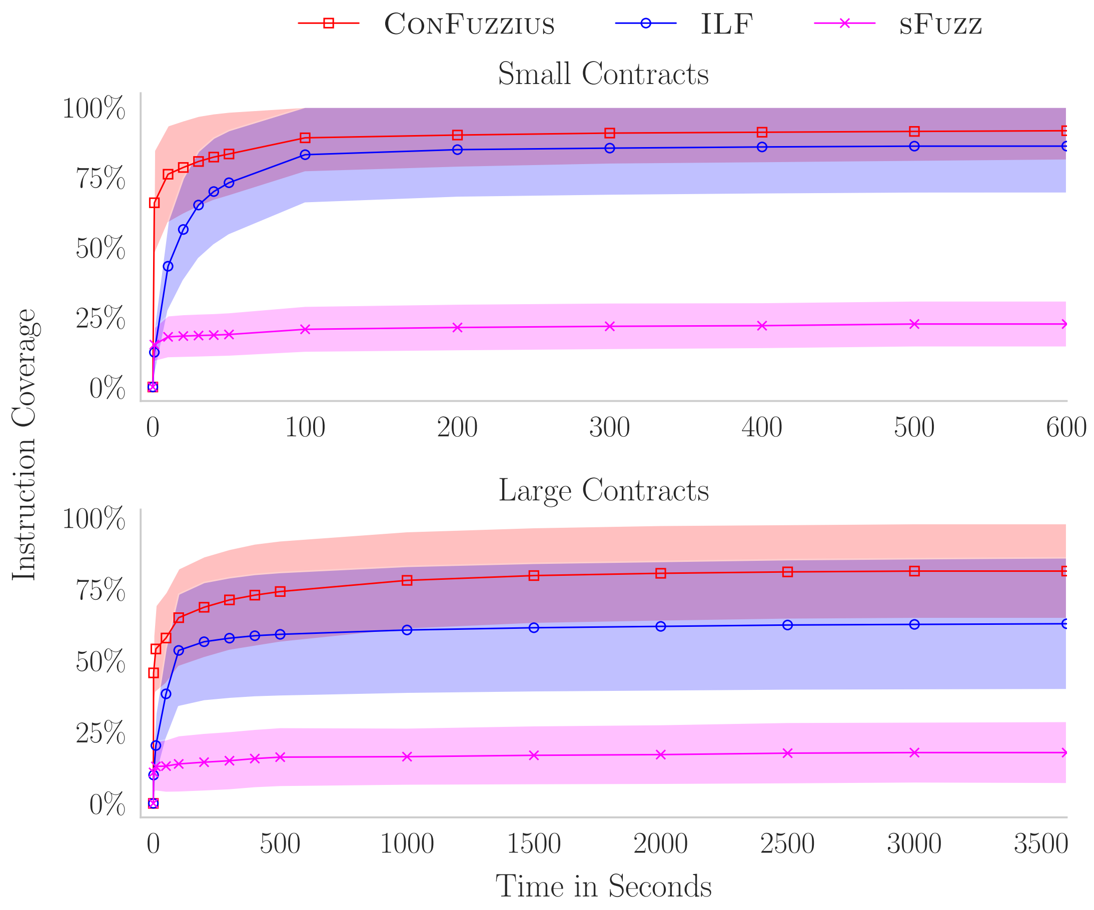
RQ1: Does ConFuzzius achieve higher code coverage than current state-of-the art symbolic execution and fuzzing tools for smart contracts?

Code Coverage

- 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

2^{nd} Research Question

RQ2: Does ConFuzzius discover more vulnerabilities than current state-of-the art symbolic execution and fuzzing tools for smart contracts?

Vulnerability Detection

- 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:

```
function setCallee(address newCallee)
  require(msg.sender == owner);
  callee = newCallee;
}
function forward(bytes _data) {
  require(callee.delegatecall(_data));
}
```

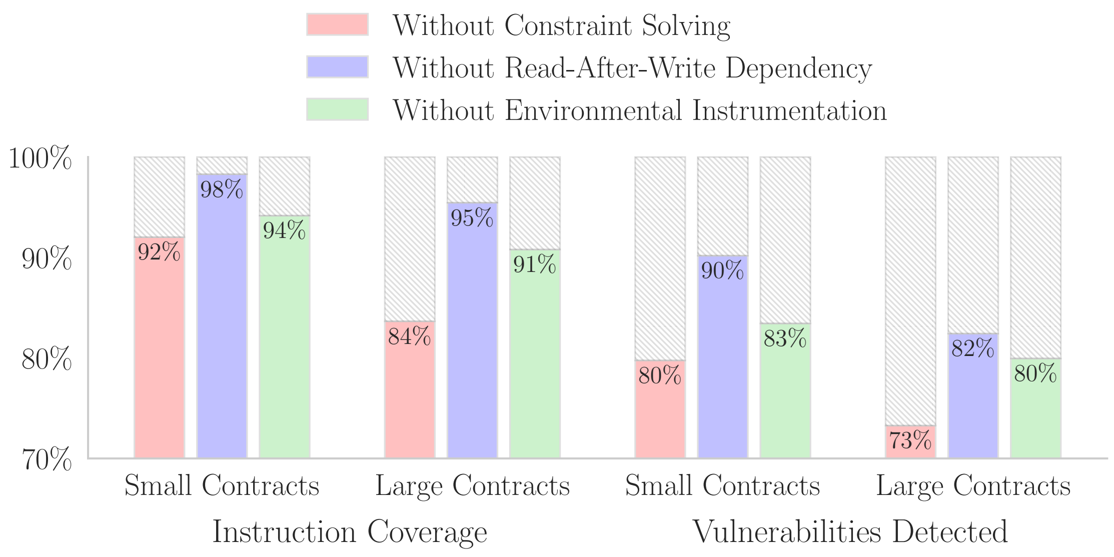
Toolname	Vulnerabilities										Total
	AF	IO	RE	TD	BD	UE	UD	LE	LO	US	
OYENTE	6/6	12/4	8/0	2/0	0/0	-	-	-	-	0/0	28
MYTHRIL	7/3	18/5	10/0	0/0	3/0	24/0	0/0	4/0	-	2/0	68
M-PRO	7/3	18/5	10/0	0/0	3/0	24/0	0/0	4/0	-	2/0	68
ILF	-	-	-	-	0/0	10/0	1/2	4/0	5/0	3/0	23
SFUZZ	-	12/0	7/0	-	1/0	21/0	1/2	-	0/0	-	42
CONFUZZIUS	**10/0**	**18/0**	**10/0**	**2/0**	**7/0**	**46/0**	**1/0**	**4/0**	**5/0**	**3/0**	**106**
Total Unique	14	19	11	4	7	75	1	9	5	3	148

3rd Research Question

RQ3: How relevant are ConFuzzius's individual components in terms of code coverage and vulnerability detection?

Component Evaluation

- 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%**).

Questions?

All **code** & **data** is available on GitHub:

https://github.com/christoftorres/ConFuzzius

Contact information:

christof.torres@uni.lu

Supported by:

Luxembourg National Research Fund

Horizon 2020 European Union Funding for Research & Innovation