Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università Ca'Foscari Venezia

Reentrancy Detection in the Age of LLMs*

Dalila Ressi, Alvise Spanò, Matteo Rizzo, Lorenzo Benetollo, Sabina Rossi
Ca' Foscari University of Venice

DLT2026 – 8th Distributed Ledger Technology Workshop
Pula, 04 - 06 June 2026

*Accepted at the 56th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN2026)

How it started

?

Can AI be used to detect vulnerabilities in Smart Contracts?

What kind of vulnerabilities?

Why

Real Loss

Assets (tokens)
Liquidity (Ether, Wei)

What

is the State of the Art?

How it started

Can AI be used to detect vulnerabilities in Smart Contracts?

What kind of vulnerabilities?

Real Loss

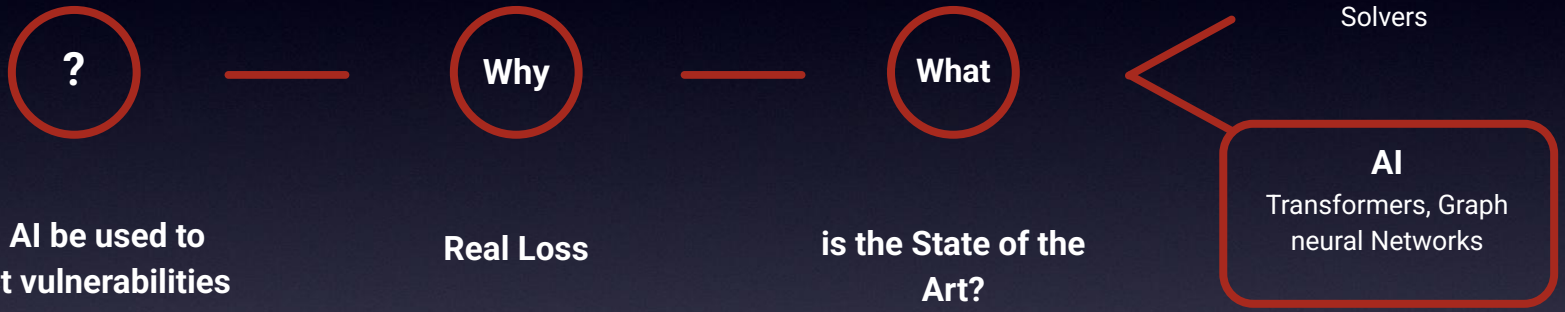
Assets (tokens)
Liquidity (Ether, Wei)

is the State of the Art?

Formal methods
Symbolic Execution + Solvers

AI
Transformers, Graph neural Networks

How it started



Can AI be used to detect vulnerabilities in Smart Contracts?

What kind of vulnerabilities?

Real Loss

Assets (tokens)
Liquidity (Ether, Wei)

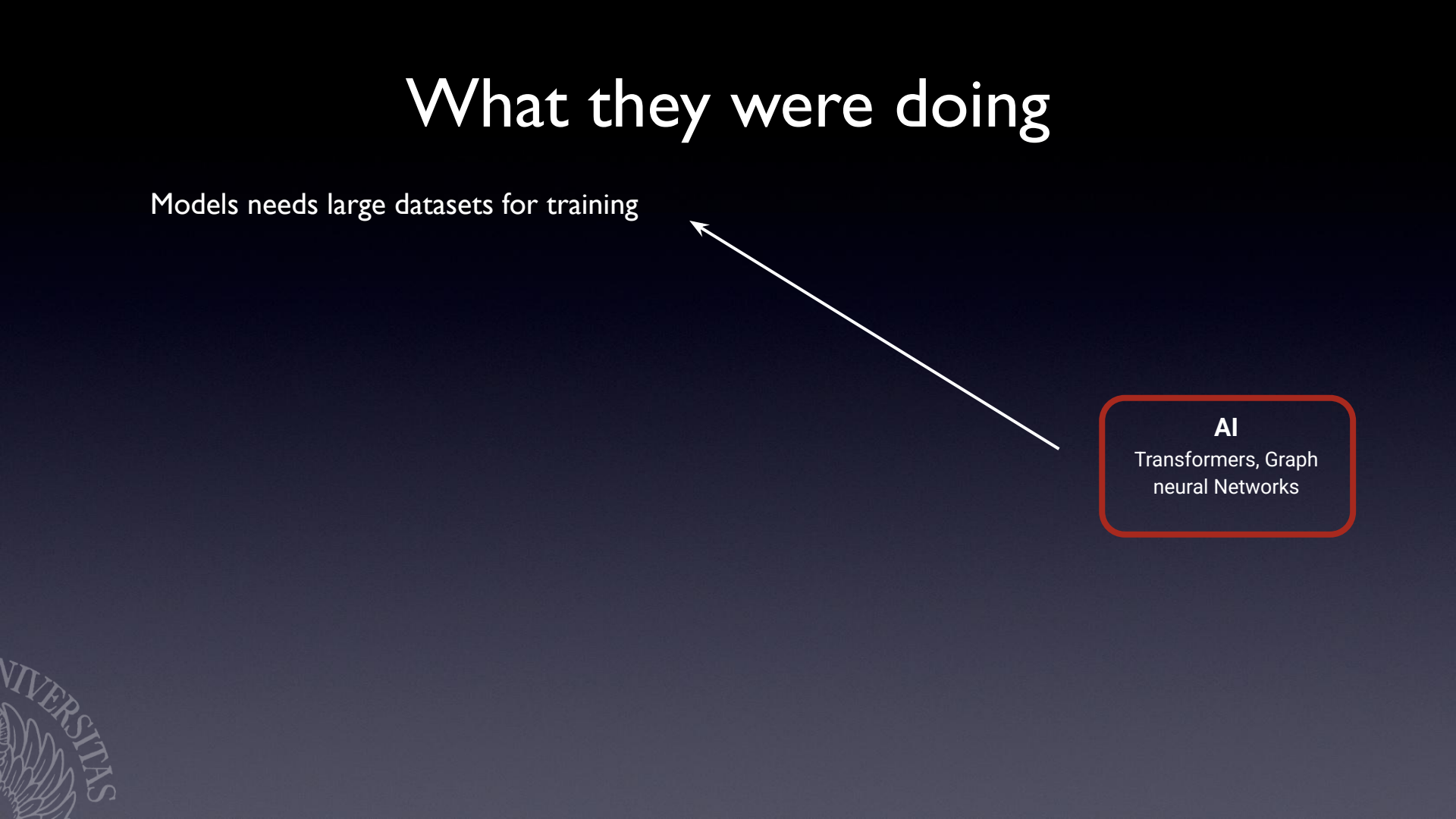
is the State of the Art?

What they were doing

AI
Transformers, Graph neural Networks

What they were doing

Models needs large datasets for training

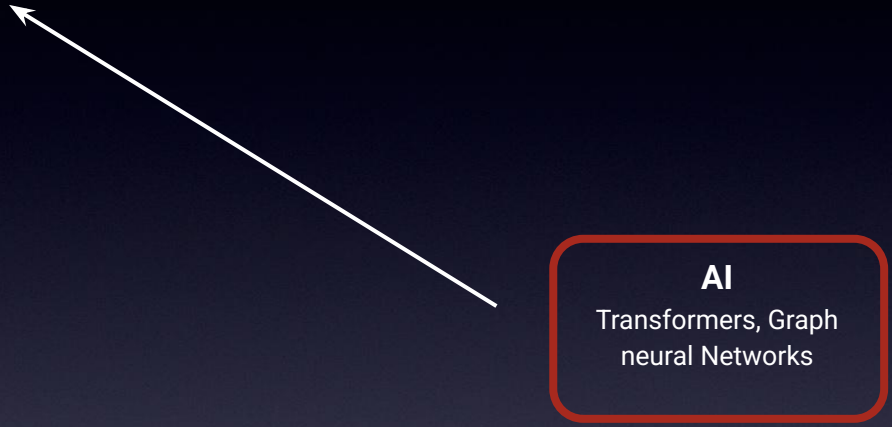


What they were doing

Models needs large datasets for training

↓

AI community ≠ Security experts



What they were doing

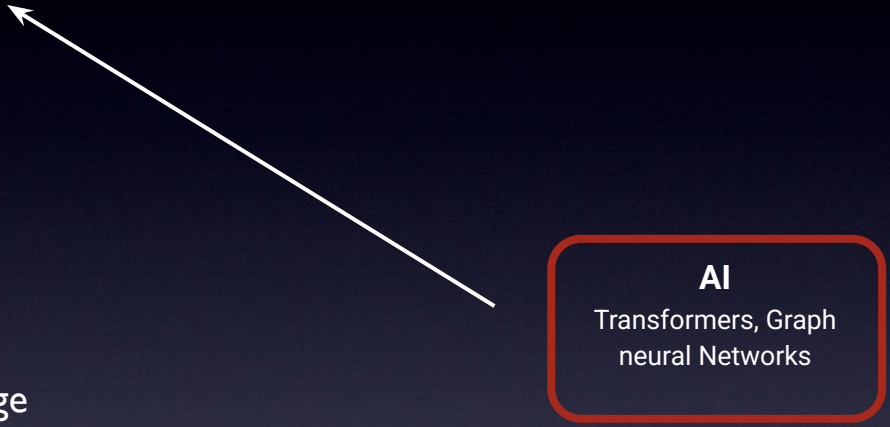
Models needs large datasets for training

↓

AI community ≠ Security experts

↓

Use formal methods based detectors to label large corpora of contracts



What they were doing

Models needs large datasets for training

↓

AI community ≠ Security experts

↓

Use formal methods based detectors to label large corpora of contracts

↓

Train models on that **dataset**

AI
Transformers, Graph neural Networks

What they were doing

Models needs large datasets for training

↓

AI community ≠ Security experts

↓

Use formal methods based detectors to label large corpora of contracts

↓

Train models on that **dataset** → ***Very old contracts, deprecated syntaxes, different definitions of vulnerabilities***

AI
Transformers, Graph neural Networks

Detection Tools

Tool	Version/Commit	Working	Input	Techniques
Aderyn	v0.6.5	✓	Source	Abstract syntax tree traversal and rule-based static pattern matching
AutoAR	N/A	N/A	Mixed	Program dependency graph construction and inter-procedural static analysis
CCC	#c531ae3	✓	Bytecode	Code property graph analysis with Cypher query patterns
ConFuzzius	#4315fb7 v0.0.1	✓	Runtime	Hybrid approach combining symbolic execution, syntax inspection, SMT solving, and fuzzing
Conkas	#4e0f256	✓	Runtime	Symbolic execution with syntactic and SMT-based constraint analysis
ContractWard	N/A	✓	Bytecode	Machine-learning-based vulnerability classification with AST feature extraction
eThor	2023	×	Bytecode	Abstract interpretation, Horn clause resolution, and SMT solving
DefectChecker	N/A	✓	Bytecode	Abstract syntax tree analysis with supervised machine learning
Manticore	v0.3.7	×	Bytecode	Symbolic execution and SMT-based path exploration
Mythril	v0.24.8	✓	Bytecode	Symbolic execution with constraint solving and pattern-based vulnerability detection
NPChecker	N/A	N/A	Bytecode	Inter-procedural static analysis with constraint solving
Oyente+	#060ca34	✓	Bytecode	Symbolic execution and SMT constraint analysis
Sailfish	v0.1	✓	Bytecode	Symbolic execution combined with static dependency and control-flow analysis
Securify	v1.0	✓	Bytecode	Datalog-based data-flow and call-dependency analysis
Securify2	N/A	×	Bytecode	Declarative intermediate-representation analysis using Datalog compliance and violation rules
Sereum	v1.0	N/A	Runtime	Dynamic taint tracking on Ethereum bytecode during runtime execution
sFuzz	#48934c0	✓	Runtime	Feedback-guided fuzzing with transaction sequence generation
Slither	v0.11.3	✓	Source	Static pattern matching with inter-procedural data-flow and control-flow analysis
Smartcheck	N/A	×	Source	Abstract syntax tree pattern matching and heuristic regular expressions
Solhint	v6.0.0	✓	Bytecode	Linting rules and static pattern matching on Solidity source code
teEther	#04adf56	×	Bytecode	Symbolic execution and SMT-based exploit generation
TotalSol	#04adf56	N/A	Mixed	Multi-layer analysis combining control-flow, data-flow and inter-procedural dependency
Vandal	#d2b0043	✓	Bytecode	Bytecode decompilation and Datalog-based static analysis

A reasonable doubt

AI detector as good as the tool used for labeling…

…but how good are these tools?

Meanwhile… LLMs start to be widely used. Can they do better?

A fair setup

1) One single vulnerability: **REENTRANCY**
2) Assessment of detection tools/LLMs according to:

Reliability (correctness) + Robustness (functionality) = Dependability

3) Need of manually labeled dataset(s) to avoid circularity
4) *Can LLMs help us in the labeling process?*

Datasets: Aggregated Benchmark

Aggregation of 432 contracts from 3 among the most used benchmarks claimed to be *manually inspected:*

- **Consolidated Groundtruth** (Smarbugs research group)
- **HuangGui dataset** (vulnerability injection)
- **"Turn the rudder"** dataset (manual inspection, ICSE23)

PROBLEM: we needed to double check if the labels were consistent, but we were not very expert at the time
BONUS PROBLEM: PhD student needed to publish about explainability

Datasets: Aggregated Benchmark

Aggregation of 432 contracts from 3 among the most used benchmarks claimed to be *manually inspected:*

- **Consolidated Groundtruth** (Smarbugs research group)
- **HuangGui dataset** (vulnerability injection)
- **"Turn the rudder"** dataset (manual inspection, ICSE23)

PROBLEM: we needed to double check if the labels were consistent, but we were not very expert at the time
BONUS PROBLEM: PhD student needed to publish about explainability
SOLUTION: Ask gpt if the contracts are vulnerable to reentrancy, and provide an **explanation**

Datasets: Aggregated Benchmark

Aggregation of 432 contracts from 3 among the most used benchmarks claimed to be *manually inspected:*

- **Consolidated Groundtruth** (Smarbugs research group)
- **HuangGui dataset** (vulnerability injection)
- **"Turn the rudder"** dataset (manual inspection, ICSE23)

PROBLEM: we needed to double check if the labels were consistent, but we were not very expert at the time
BONUS PROBLEM: PhD student needed to publish about explainability
SOLUTION: Ask gpt if the contracts are vulnerable to reentrancy, and provide an **explanation**

- ➔ Hundred of contracts
- ➔ Hundreds of explanations
- ➔ Blogs
- ➔ Forums
- ➔ Official docs

Datasets: Aggregated Benchmark

Aggregation of 432 contracts from 3 among the most used benchmarks claimed to be *manually inspected:*

- **Consolidated Groundtruth** (Smarbugs research group)
- **HuangGui dataset** (vulnerability injection)
- **"Turn the rudder"** dataset (manual inspection, ICSE23)

PROBLEM: we needed to double check if the labels were consistent, but we were not very expert at the time
BONUS PROBLEM: PhD student needed to publish about explainability
SOLUTION: Ask gpt if the contracts are vulnerable to reentrancy, and provide an **explanation**

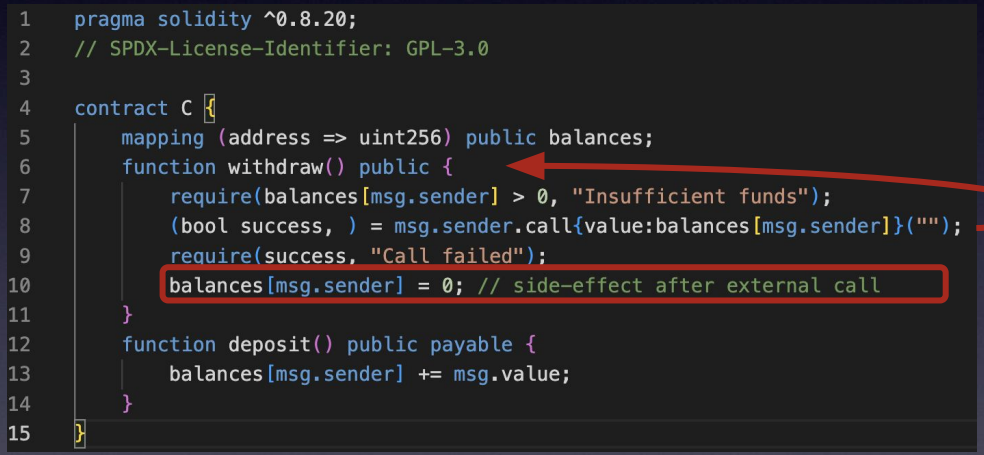
- ➔ Hundred of contracts
- ➔ Hundreds of explanations
- ➔ Blogs
- ➔ Forums
- ➔ Official docs

At a certain point, we had seen so many contracts that we decided to synthetize our own dataset and a operational definition of reentrancy

Reentrancy

A contract is vulnerable iff:

1) it performs an external call to attacker-controlled code;
2) execution later performs a state update affected by that interaction;
3) reentrant execution reaches a state that cannot be reproduced without reentrancy.



```
1   pragma solidity ^0.8.20;
2   // SPDX-License-Identifier: GPL-3.0
3
4   contract C {
5       mapping (address => uint256) public balances;
6       function withdraw() public {
7           require(balances[msg.sender] > 0, "Insufficient funds");
8           (bool success, ) = msg.sender.call{value:balances[msg.sender]}("");
9           require(success, "Call failed");
10          balances[msg.sender] = 0; // side-effect after external call
11      }
12      function deposit() public payable {
13          balances[msg.sender] += msg.value;
14      }
15  }
```

Datasets: Reentrancy Scenario Dataset

- 143 Minimal Working Examples (**MWEs**)
- Very high coverage of possible **external calls** (all primitives, high level calls)
- Simple AND **complex** reentrancy scenarios (cross contract reentrancy, read only, attacks from constructor)
- Some of the most common contracts (ERC20, proxies)
- Modern Solidity (v 0.8.20)

Datasets: Reentrancy Scenario Dataset

Code	Name Prefix	#Safe	#Ree	Reentrancy Type	Description	Variants
00	Basic	8	7	single function	Basic DAO in multiple variants and flavours	Emit, Error, Fold, Staticcall, Unchecked, Inline
00	BasicConst	1	1	single function	Basic DAO with constant address as target	
00	BasicCross	0	1	cross function	Basic DAO in a cross-function setting	
00	BasicNoChecks	1	1	single function	Basic DAO implementing CEI and omitting the checks part	
00	BasicNoCall	1	0	single function	Dummy contract without external calls	
01	SingleMutex	4	6	single function	DAO-like with flag-based mutex protection in a single function settings	Fold, Underflow
02	CrossMutex	4	4	cross function	DAO-like with flag-based mutex protection in a cross-function setting	Underflow, Unchecked
03	SingleMod	5	8	single function	DAO-like with a modifier guard in a single function settings	Fold, Underflow
04	CrossMod	5	8	cross function	DAO-like with a modifier guard in a cross-function setting	Fold, Underflow
05	Send	6	0	single function	DAO-like using send primitive (always safe)	Emit, Unchecked
06	Transfer	4	0	single function	DAO-like using transfer primitive (always safe)	Unchecked
07	MixedSend	3	3	single function	Send primitive followed by a low-level call	Fold, Emit
08	MixedTransfer	2	2	single function	Transfer primitive followed by a low-level call	Emit
09	ERC20	7	10	single function	Donations via ERC20 tokens mixing various high-level calls	Mod, Inheritance, Staticcall
09	ERC20OnlyOnce	2	1	single function	Donations via ERC20 tokens that can occur only once per user	
09	ERC20Staking	3	3	single function	Staking application with ERC20 tokens	
09	ERC20StakingPull	3	3	single function	Staking application with ERC20 tokens including withdrawal	Mod
10	OnlyOnce	2	0	single function	DAO-like where withdrawal can occur only once per user (always safe)	
11	Proxy	3	0	none	Proxy contract forwarding calls to a given target	Staticcall
12	OnlyOwner	1	1	cross function	Special guard checking if the sender is the original owner of the contract	
13	Loop	1	1	single function	Loop containing CEI-breaking code	
13	LoopCrossMod	3	4	cross function	Loop containing CEI-breaking code in a cross-function setting	
13	LoopCrossMutex	3	4	cross function	Loop guarded by mutexes	
14	DelegateCall	0	4	delegatecall	Delegate calls are always vulnerable under our definition	
15	ReadOnly	3	3	cross contract	Modifications in the state of one contract make another vulnerable	Staticcall

Two datasets, different purposes

Aggregated Benchmark

Assess performance on real world contracts
Include old versions of contracts (Solidity 0.4/0.5)

*Address **RELIABILITY***

Grant FAIRNESS of the evaluation

RSD

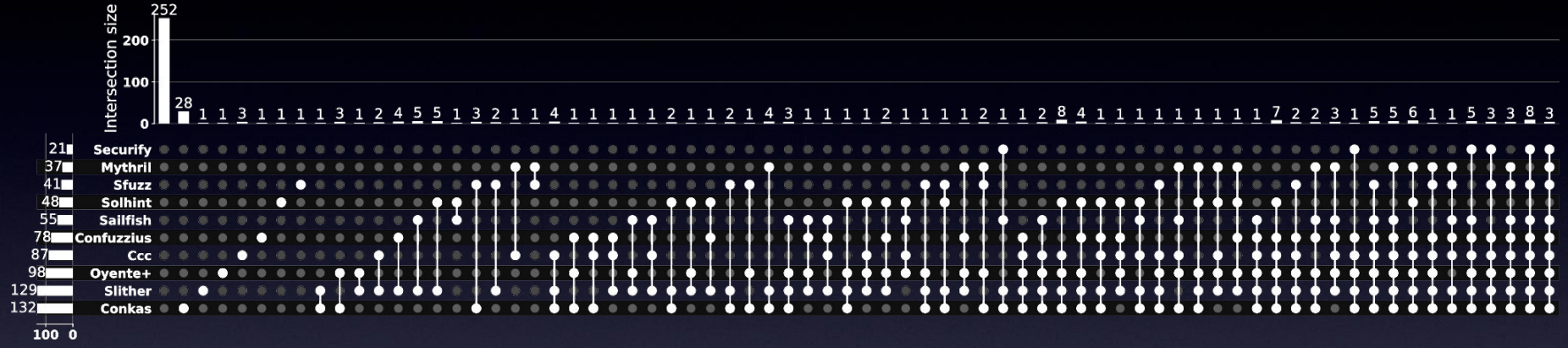
Assess the robustness of the tools

*Address **ROBUSTNESS** (and reliability)*

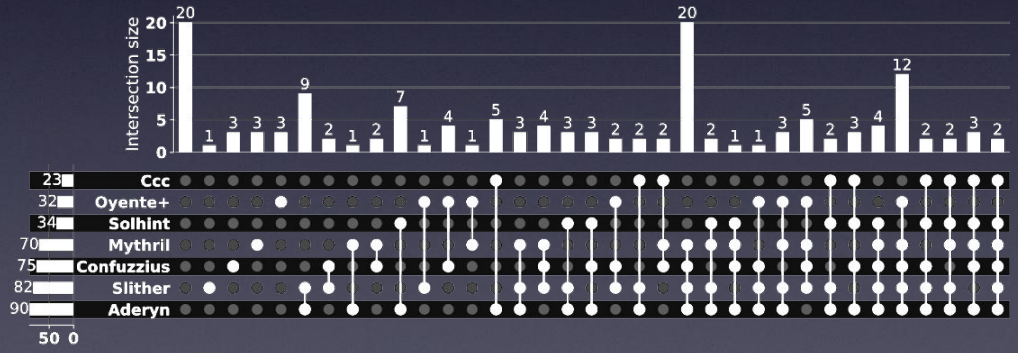
Grant evaluation to be HIGH COVERAGE

About Reliability

Agreement Across Tools - Aggregated Benchmark



Agreement Across Tools - RSD



Evaluation Results

Low Precision:
many false positives

		Aggregated Benchmark					*Reentrancy Scenarios Dataset (RSD)*				
	Tool/Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error (%)**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error (%)**
Analyzers	Aderyn	-	-	-	-	100.00	0.66	0.62	0.79	0.70	0.00
	CCC	0.89	0.91	0.66	0.76	25.46	0.58	0.74	0.24	0.36	0.00
	Confuzzius	0.88	0.95	0.62	0.75	15.97	0.58	0.57	0.61	0.59	12.59
	Conkas	0.84	0.67	0.78	0.72	18.18	-	-	-	-	100.00
	Mythril	0.86	0.84	0.48	0.61	7.22	0.67	0.67	0.66	0.67	0.00
	Oyente+	0.91	0.92	0.75	0.83	0.00	0.60	0.72	0.32	0.45	0.00
	Sailfish	0.83	0.93	0.42	0.58	0.00	-	-	-	-	100.00
	Securify	0.88	**1.00**	0.48	0.65	25.54	-	-	-	-	100.00
	Sfuzz	0.76	0.71	0.24	0.36	45.14	-	-	-	-	100.00
	Slither	0.95	0.88	0.95	0.92	0.00	0.74	0.71	0.82	0.76	0.00
	Solhint	0.81	0.90	0.36	0.51	0.69	0.55	0.59	0.28	0.38	0.00
	Vandal	0.74	0.66	0.52	0.58	68.72	0.52	0.57	0.25	0.34	78.24
Machine Learning	gradient_boosting	0.90 ± 0.04	0.87 ± 0.04	0.76 ± 0.11	0.81 ± 0.05	-	0.62 ± 0.02	0.63 ± 0.03	0.62 ± 0.02	0.61 ± 0.02	-
	gnb	0.82 ± 0.05	0.70 ± 0.06	0.65 ± 0.12	0.67 ± 0.07	-	0.58 ± 0.03	0.71 ± 0.05	0.58 ± 0.03	0.51 ± 0.07	-
	knn	0.88 ± 0.01	0.85 ± 0.13	0.74 ± 0.13	0.78 ± 0.01	-	0.62 ± 0.01	0.62 ± 0.01	0.62 ± 0.01	0.62 ± 0.01	-
	logistic_regression	0.80 ± 0.06	0.93 ± 0.12	0.33 ± 0.12	0.47 ± 0.12	-	0.49 ± 0.01	0.33 ± 0.12	0.49 ± 0.01	0.35 ± 0.03	-
	random_forest	0.90 ± 0.04	0.90 ± 0.07	0.73 ± 0.13	0.80 ± 0.07	-	0.63 ± 0.03	0.64 ± 0.03	0.63 ± 0.03	0.62 ± 0.04	-
	svm	0.84 ± 0.04	0.93 ± 0.12	0.51 ± 0.16	0.64 ± 0.09	-	0.50 ± 0.00	0.33 ± 0.01	0.50 ± 0.00	0.34 ± 0.02	-
	xgboost	0.91 ± 0.02	0.88 ± 0.08	0.79 ± 0.10	0.83 ± 0.02	-	0.72 ± 0.03	0.72 ± 0.03	0.72 ± 0.03	0.72 ± 0.03	-
	codebert	0.90 ± 0.13	0.82 ± 0.24	$\mathbf{0.96} \pm 0.03$	0.87 ± 0.14	-	0.65 ± 0.06	0.67 ± 0.07	0.65 ± 0.06	0.64 ± 0.06	-
	lstm	0.86 ± 0.12	0.76 ± 0.17	$\mathbf{0.96} \pm 0.03$	0.83 ± 0.11	-	0.60 ± 0.05	0.62 ± 0.06	0.60 ± 0.05	0.59 ± 0.05	-
	ffnn	$\mathbf{0.96} \pm 0.01$	0.85 ± 0.03	0.89 ± 0.03	0.86 ± 0.02	-	0.61 ± 0.03	0.62 ± 0.03	0.61 ± 0.03	0.61 ± 0.03	-
LLMs	gemini-2.0-flash	0.87	0.87	0.84	0.85	-	0.51	0.52	0.52	0.49	-
	gemini-2.5-flash	0.83	0.85	0.82	0.83	-	0.58	0.66	0.58	0.53	-
	gpt-4o	0.85	0.89	0.85	0.86	-	0.53	0.53	0.53	0.51	-
	gpt-4.1	0.87	0.90	0.87	0.88	-	0.63	0.63	0.63	0.63	-
	o3-mini	**0.96**	0.96	**0.96**	**0.96**	-	0.79	0.80	0.79	0.79	-
	o4-mini	0.94	0.94	0.94	0.93	-	0.82	0.83	**0.82**	**0.82**	-
	gpt-oss	0.94	0.96	0.82	0.88	-	**0.87**	**0.98**	0.75	0.85	-
	qwen3	0.91	0.78	0.93	0.85	-	0.68	0.66	0.73	0.69	-
	phi4	0.80	0.58	**0.96**	0.72	-	0.56	0.55	0.65	0.60	-

Evaluation Results

Low Precision:
many false positives

Low Recall:
many false negatives

		Aggregated Benchmark					*Reentrancy Scenarios Dataset (RSD)*				
	Tool/Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error (%)**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error (%)**
Analyzers	Aderyn	-	-	-	-	100.00	0.66	0.62	0.79	0.70	0.00
	CCC	0.89	0.91	0.66	0.76	25.46	0.58	0.74	0.24	0.36	0.00
	Confuzzius	0.88	0.95	0.62	0.75	15.97	0.58	0.57	0.61	0.59	12.59
	Conkas	0.84	0.67	0.78	0.72	18.18	-	-	-	-	100.00
	Mythril	0.86	0.84	0.48	0.61	7.22	0.67	0.67	0.66	0.67	0.00
	Oyente+	0.91	0.92	0.75	0.83	0.00	0.60	0.72	0.32	0.45	0.00
	Sailfish	0.83	0.93	0.42	0.58	0.00	-	-	-	-	100.00
	Securify	0.88	**1.00**	0.48	0.65	25.54	-	-	-	-	100.00
	Sfuzz	0.76	0.71	0.24	0.36	45.14	-	-	-	-	100.00
	Slither	0.95	0.88	0.95	0.92	0.00	0.74	0.71	0.82	0.76	0.00
	Solhint	0.81	0.90	0.36	0.51	0.69	0.55	0.59	0.28	0.38	0.00
	Vandal	0.74	0.66	0.52	0.58	68.72	0.52	0.57	0.25	0.34	78.24
Machine Learning	gradient_boosting	0.90 ± 0.04	0.87 ± 0.04	0.76 ± 0.11	0.81 ± 0.05	-	0.62 ± 0.02	0.63 ± 0.03	0.62 ± 0.02	0.61 ± 0.02	-
	gnb	0.82 ± 0.05	0.70 ± 0.06	0.65 ± 0.12	0.67 ± 0.07	-	0.58 ± 0.03	0.71 ± 0.05	0.58 ± 0.03	0.51 ± 0.07	-
	knn	0.88 ± 0.01	0.85 ± 0.13	0.74 ± 0.13	0.78 ± 0.01	-	0.62 ± 0.01	0.62 ± 0.01	0.62 ± 0.01	0.62 ± 0.01	-
	logistic_regression	0.80 ± 0.06	0.93 ± 0.12	0.33 ± 0.12	0.47 ± 0.12	-	0.49 ± 0.01	0.33 ± 0.12	0.49 ± 0.01	0.35 ± 0.03	-
	random_forest	0.90 ± 0.04	0.90 ± 0.07	0.73 ± 0.13	0.80 ± 0.07	-	0.63 ± 0.03	0.64 ± 0.03	0.63 ± 0.03	0.62 ± 0.04	-
	svm	0.84 ± 0.04	0.93 ± 0.12	0.51 ± 0.16	0.64 ± 0.09	-	0.50 ± 0.00	0.33 ± 0.01	0.50 ± 0.00	0.34 ± 0.02	-
	xgboost	0.91 ± 0.02	0.88 ± 0.08	0.79 ± 0.10	0.83 ± 0.02	-	0.72 ± 0.03	0.72 ± 0.03	0.72 ± 0.03	0.72 ± 0.03	-
	codebert	0.90 ± 0.13	0.82 ± 0.24	$\mathbf{0.96} \pm 0.03$	0.87 ± 0.14	-	0.65 ± 0.06	0.67 ± 0.07	0.65 ± 0.06	0.64 ± 0.06	-
	lstm	0.86 ± 0.12	0.76 ± 0.17	$\mathbf{0.96} \pm 0.03$	0.83 ± 0.11	-	0.60 ± 0.05	0.62 ± 0.06	0.60 ± 0.05	0.59 ± 0.05	-
	ffnn	$\mathbf{0.96} \pm 0.01$	0.85 ± 0.03	0.89 ± 0.03	0.86 ± 0.02	-	0.61 ± 0.03	0.62 ± 0.03	0.61 ± 0.03	0.61 ± 0.03	-
LLMs	gemini-2.0-flash	0.87	0.87	0.84	0.85	-	0.51	0.52	0.52	0.49	-
	gemini-2.5-flash	0.83	0.85	0.82	0.83	-	0.58	0.66	0.58	0.53	-
	gpt-4o	0.85	0.89	0.85	0.86	-	0.53	0.53	0.53	0.51	-
	gpt-4.1	0.87	0.90	0.87	0.88	-	0.63	0.63	0.63	0.63	-
	o3-mini	**0.96**	0.96	**0.96**	**0.96**	-	0.79	0.80	0.79	0.79	-
	o4-mini	0.94	0.94	0.94	0.93	-	0.82	0.83	**0.82**	**0.82**	-
	gpt-oss	0.94	0.96	0.82	0.88	-	**0.87**	**0.98**	0.75	0.85	-
	qwen3	0.91	0.78	0.93	0.85	-	0.68	0.66	0.73	0.69	-
	phi4	0.80	0.58	**0.96**	0.72	-	0.56	0.55	0.65	0.60	-

Evaluation Results

Low Precision: many false positives

Low Recall: many false negatives

Low Error: high robustness

		Aggregated Benchmark					*Reentrancy Scenarios Dataset (RSD)*				
	Tool/Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error (%)**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error (%)**
Analyzers	Aderyn	-	-	-	-	100.00	0.66	0.62	0.79	0.70	0.00
	CCC	0.89	0.91	0.66	0.76	25.46	0.58	0.74	0.24	0.36	0.00
	Confuzzius	0.88	0.95	0.62	0.75	15.97	0.58	0.57	0.61	0.59	12.59
	Conkas	0.84	0.67	0.78	0.72	18.18	-	-	-	-	100.00
	Mythril	0.86	0.84	0.48	0.61	7.22	0.67	0.67	0.66	0.67	0.00
	Oyente+	0.91	0.92	0.75	0.83	0.00	0.60	0.72	0.32	0.45	0.00
	Sailfish	0.83	0.93	0.42	0.58	0.00	-	-	-	-	100.00
	Securify	0.88	**1.00**	0.48	0.65	25.54	-	-	-	-	100.00
	Sfuzz	0.76	0.71	0.24	0.36	45.14	-	-	-	-	100.00
	Slither	0.95	0.88	0.95	0.92	0.00	0.74	0.71	0.82	0.76	0.00
	Solhint	0.81	0.90	0.36	0.51	0.69	0.55	0.59	0.28	0.38	0.00
	Vandal	0.74	0.66	0.52	0.58	68.72	0.52	0.57	0.25	0.34	78.24
Machine Learning	gradient_boosting	0.90 ± 0.04	0.87 ± 0.04	0.76 ± 0.11	0.81 ± 0.05	-	0.62 ± 0.02	0.63 ± 0.03	0.62 ± 0.02	0.61 ± 0.02	-
	gnb	0.82 ± 0.05	0.70 ± 0.06	0.65 ± 0.12	0.67 ± 0.07	-	0.58 ± 0.03	0.71 ± 0.05	0.58 ± 0.03	0.51 ± 0.07	-
	knn	0.88 ± 0.01	0.85 ± 0.13	0.74 ± 0.13	0.78 ± 0.01	-	0.62 ± 0.01	0.62 ± 0.01	0.62 ± 0.01	0.62 ± 0.01	-
	logistic_regression	0.80 ± 0.06	0.93 ± 0.12	0.33 ± 0.12	0.47 ± 0.12	-	0.49 ± 0.01	0.33 ± 0.12	0.49 ± 0.01	0.35 ± 0.03	-
	random_forest	0.90 ± 0.04	0.90 ± 0.07	0.73 ± 0.13	0.80 ± 0.07	-	0.63 ± 0.03	0.64 ± 0.03	0.63 ± 0.03	0.62 ± 0.04	-
	svm	0.84 ± 0.04	0.93 ± 0.12	0.51 ± 0.16	0.64 ± 0.09	-	0.50 ± 0.00	0.33 ± 0.01	0.50 ± 0.00	0.34 ± 0.02	-
	xgboost	0.91 ± 0.02	0.88 ± 0.08	0.79 ± 0.10	0.83 ± 0.02	-	0.72 ± 0.03	0.72 ± 0.03	0.72 ± 0.03	0.72 ± 0.03	-
	codebert	0.90 ± 0.13	0.82 ± 0.24	$\mathbf{0.96} \pm 0.03$	0.87 ± 0.14	-	0.65 ± 0.06	0.67 ± 0.07	0.65 ± 0.06	0.64 ± 0.06	-
	lstm	0.86 ± 0.12	0.76 ± 0.17	$\mathbf{0.96} \pm 0.03$	0.83 ± 0.11	-	0.60 ± 0.05	0.62 ± 0.06	0.60 ± 0.05	0.59 ± 0.05	-
	ffnn	$\mathbf{0.96} \pm 0.01$	0.85 ± 0.03	0.89 ± 0.03	0.86 ± 0.02	-	0.61 ± 0.03	0.62 ± 0.03	0.61 ± 0.03	0.61 ± 0.03	-
LLMs	gemini-2.0-flash	0.87	0.87	0.84	0.85	-	0.51	0.52	0.52	0.49	-
	gemini-2.5-flash	0.83	0.85	0.82	0.83	-	0.58	0.66	0.58	0.53	-
	gpt-4o	0.85	0.89	0.85	0.86	-	0.53	0.53	0.53	0.51	-
	gpt-4.1	0.87	0.90	0.87	0.88	-	0.63	0.63	0.63	0.63	-
	o3-mini	**0.96**	0.96	**0.96**	**0.96**	-	0.79	0.80	0.79	0.79	-
	o4-mini	0.94	0.94	0.94	0.93	-	0.82	0.83	**0.82**	**0.82**	-
	gpt-oss	0.94	0.96	0.82	0.88	-	**0.87**	**0.98**	0.75	0.85	-
	qwen3	0.91	0.78	0.93	0.85	-	0.68	0.66	0.73	0.69	-
	phi4	0.80	0.58	**0.96**	0.72	-	0.56	0.55	0.65	0.60	-

Takeaways

Finding 1
There is no consensus definition of reentrancy

Finding 2
Tool dependability has degraded over time

Finding 3
Modern LLMs are already competitive baselines (and extremely useful for labeling)

Thank you!