

Machine Learning for Vulnerability Detection in Ethereum Smart Contracts

Dalila Ressi, Alvisè Spanò, Lorenzo Benetollo, Carla Piazza,
Michele Bugliesi, Sabina Rossi



Università
Ca' Foscari
Venezia



**UNIVERSITÀ
DEGLI STUDI
DI UDINE**
hic sunt futura



UNIVERSITÀ
di CAMERINO

Vulnerabilities in Ethereum Smart Contracts: Why?

2016 DAO attack 70 million dollars
worth in ETH

Hard fork (ETH and ETC)

Reentrancy

```
1 // UNSAFE EXTERNAL CALL: Reentrancy
2
3 // CONTRACT 1
4 contract EtherStore {
5     mapping(address => uint) public balances;
6
7     function deposit() public payable {
8         balances[msg.sender] += msg.value;
9     }
10
11     // VULNERABLE FUNCTION
12     function withdraw() external {
13         uint256 amount = balances[msg.sender];
14         (bool success,) = msg.sender.call{value:
            balances[msg.sender]}("");
15         require(success);
16         balances[msg.sender] = 0;
17     }
18 }
19
20 // CONTRACT 2
21 contract Attacker {
22     fallback() external payable {
23         if (address(msg.sender).balance >= 1 ether) {
24             EtherStore(msg.sender).withdraw();
25         }
26     }
27 }
```

Vulnerabilities in Ethereum Smart Contracts: What?

Some vulnerabilities are very well known, others can be found under different nomenclatures

DASP TOP 10

1. Reentrancy
2. Access Control
3. Arithmetic
4. Unchecked Low Level Calls
5. Denial of Services
6. Bad Randomness
7. Front Running
8. Time Manipulation
9. Short Addresses
10. Unknown Unknowns

Vulnerabilities in Ethereum Smart Contracts: What?

Other taxonomies categorize the vulnerabilities according to where they originate from

Denial of Service

“tx.origin” usage

Integer Overflow/Underflow

Re-entrancy

Call to the unknown

Gasless “send”

...



Solidity Programming Language

Immutable bugs/mistakes

Ether lost in transfer

...



Ethereum Virtual Machine

Timestamp dependency

Transaction Ordering Dependency

...



Ethereum Blockchain Design

Vulnerabilities in Ethereum Smart Contracts: What?

“OpenSCV: An Open Hierarchical Taxonomy for Smart Contract Vulnerabilities” Vidal et al. (2023)

1. Unsafe External Calls

- 1.1 Reentrancy
- 1.2 Malicious Fallback Function
- 1.3 Improper Check of External Call Result
- 1.4 Improper locking during external calls
- 1.5 Interoperability issues with other contracts
- 1.6 Delegatecall to Untrusted Callee

2. Mishandled Events

- 2.1 Improper Exceptional Events Handling
- 2.2 Improper Token Exception Handling

3. Gas Depletion

- 3.1 Improper Gas Requirements Checking
- 3.2 Call with hardcoded gas amount

4. Bad Programming Practices & Language Weakness

- 4.1 Bad Randomness
- 4.2 Improper Initialization
- 4.3 Improper Credit Transfer
- 4.4 Error in Function Call
- 4.5 Wrong class inheritance order

...

...

- 4.6 Improper Type Usage
- 4.7 Useless Code
- 4.8 Version Issues
- 4.9 Inadequate Data Representation
- 4.10 Improper Modifier
- 4.11 Redundant Functionality
- 4.12 Redundant Functionality
- 4.13 Buffer Overflow
- 4.14 Use of Malicious Libraries
- 4.15 Typographical Error

5. Incorrect Control Flow

- 5.1 Incorrect Sequencing of Behavior
- 5.2 Inadequate Input Validation

6. Arithmetic Issues

- 6.1 Overflow and Underflow
- 6.2 Division Bugs
- 6.3 Conversion Bugs

7. Improper Access Control

- 7.1 Incorrect Authentication or Authorization
- 7.2 Improper Protection of Sensitive Data
- 7.3 Cryptography Misuse

74 vulnerabilities

Most complete work we could find

What Kind of Detectors are Available?

Given a fixed set of vulnerabilities, are we able to find them on smart contracts?

Two types of detector:

- Frameworks/tools based on static analysis and formal verification
- Machine learning based solutions

What Kind of Detectors are Available?

Given a fixed set of vulnerabilities, are we able to find them on smart contracts?

Two types of detector:

- **Frameworks/tools based on static analysis and formal verification**
- Machine learning based solutions

Vulnerability Detectors: Formal Verification Techniques

Hundreds of different tools available

Most of them are based on static analysis (abstract interpretation, taint analysis, model checking, symbolic execution)

Examples:

- Slither
- Mythril
- Oyente
- Securify
- SmartCheck
- SmartScan
- ...

Analysing the analysers: SmartBugs Framework

Authors propose:

- Framework
- Datasets:
 - Smartbugs Wild
 - Smartbugs Curated

SmartBugs: A Framework to Analyze Solidity Smart Contracts

João F. Ferreira
INESC-ID and IST, University of Lisbon, Portugal
joao@joaoff.com

Thomas Durieux
KTH Royal Institute of Technology, Sweden
thomas@durieux.me

Pedro Cruz
INESC-ID and IST, University of Lisbon, Portugal
pedrocrvz@gmail.com

Rui Abreu
INESC-ID and IST, University of Lisbon, Portugal
rui@computer.org

Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts

Thomas Durieux
INESC-ID and IST, University of Lisbon, Portugal
thomas@durieux.me

Rui Abreu
INESC-ID and IST, University of Lisbon, Portugal
rui@computer.org

João F. Ferreira
INESC-ID and IST, University of Lisbon, Portugal
joao@joaoff.com

Pedro Cruz
INESC-ID and IST, University of Lisbon, Portugal
pedrocrvz@gmail.com

Analysing the analysers: SmartBugs Framework

Authors propose:

- **Framework**
- **Datasets:**
 - Smartbugs Wild
 - Smartbugs Curated

Tool	Version	New	Contract format		
			Solidity	Creation	Runtime
ConFuzzius	#4315fb7	✓	✓		
Conkas	#4e0f256		✓		✓
Ethainter		✓			✓
eThor	2021 (CCS'20)	✓			✓
HoneyBadger	#ff30c9a		✓		✓
MadMax	#6e9a6e9	✓			✓
Maian	#4bab09a		✓	✓	✓
Manticore	0.3.7		✓		
Mythril	0.23.15		✓	✓	✓
Osiris	#d1ecc37		✓		✓
Oyente	#480e725		✓		✓
Pakala	#c84ef38	✓			✓
Securify			✓		✓
sFuzz	#48934c0	✓	✓		
Slither			✓		
Smartcheck			✓		
Solhint	3.3.8		✓		
teEther	#04adf56	✓			✓
Vandal	#d2b0043	✓			✓
19 tools		8	13	2	13

Analysing the analysers: SmartBugs Framework

Authors propose:

- Framework
- Datasets:
 - Smartbugs Wild
 - Smartbugs Curated

Table 5: Vulnerabilities identified per category by each tool. The number of vulnerabilities identified by a single tool is shown in brackets.

Category	HoneyBadger	Maian	Manticore	Mythril	Osiris	Oyente	Securify	Slither	Smartcheck	Total
Access Control	0/19 0%	0/19 0%	4/19 21%	4/19 21%	0/19 0%	0/19 0%	0/19 0%	4/19 21% (1)	2/19 11%	5/19 26%
Arithmetic	0/22 0%	0/22 0%	4/22 18%	15/22 68%	11/22 50% (2)	12/22 55% (2)	0/22 0%	0/22 0%	1/22 5%	19/22 86%
Denial Service	0/7 0%	0/7 0%	0/7 0%	0/7 0%	0/7 0%	0/7 0%	0/7 0%	0/7 0%	0/7 0%	0/7 0%
Front Running	0/7 0%	0/7 0%	0/7 0%	2/7 29%	0/7 0%	0/7 0%	2/7 29%	0/7 0%	0/7 0%	2/7 29%
Reentrancy	0/8 0%	0/8 0%	2/8 25%	5/8 62%	5/8 62%	5/8 62%	5/8 62%	7/8 88% (2)	5/8 62%	7/8 88%
Time Manipulation	0/5 0%	0/5 0%	1/5 20%	0/5 0%	0/5 0%	0/5 0%	0/5 0%	2/5 40% (1)	1/5 20% (1)	3/5 60%
Unchecked Low Calls	0/12 0%	0/12 0%	2/12 17%	5/12 42% (1)	0/12 0%	0/12 0%	3/12 25%	4/12 33% (3)	4/12 33% (1)	9/12 75%
Other	2/3 67%	0/3 0%	0/3 0%	0/3 0%	0/3 0%	0/3 0%	0/3 0%	3/3 100% (1)	0/3 0%	3/3 100%
Total	2/115 2%	0/115 0%	13/115 11%	31/115 27%	16/115 14%	17/115 15%	10/115 9%	20/115 17%	13/115 11%	48/115 42%

Table 6: Total number of detected vulnerabilities by each tool, including vulnerabilities not tagged in the dataset.

Category	HoneyBadger	Maian	Manticore	Mythril	Osiris	Oyente	Securify	Slither	Smartcheck	Total
Access Control	0	10	28	24	0	0	6	20	3	91
Arithmetic	0	0	11	92	62	69	0	0	23	257
Denial of Service	0	0	0	0	27	11	0	2	19	59
Front Running	0	0	0	21	0	0	55	0	0	76
Reentrancy	0	0	4	16	5	5	32	15	7	84
Time Manipulation	0	0	4	0	4	5	0	5	2	20
Unchecked Low Level Calls	0	0	4	30	0	0	21	13	14	82
Unknown Unknowns	5	2	25	32	0	0	0	28	8	100
Total	5	12	76	215	98	90	114	83	76	769

Analysing the analysers: SmartBugs Framework

Authors propose:

- Framework
- Datasets:
 - Smartbugs Wild
 - Smartbugs Curated
 - **Consolidated**










Table 3: Overlap of Mapped Assessments in the Consolidated GT Set.

Set	CodeSmells	ContractFuzzer	Doublade	eThor	EthRacer	EverEvolvingG	NPChecker	Zeus	JiuZhou	NotSoSmartC	SBcurated	SolidiFI	SWCregistry
CodeSmells	5300	6	4	26	0	0	14	145	0	1	0	0	0
ContractFuzzer	6	375	6	0	0	0	3	15	0	0	10	0	0
Doublade	4	6	279	2	0	0	2	10	0	0	7	0	0
eThor	26	0	2	702	0	0	25	691	0	0	0	0	0
EthRacer	0	0	0	0	111	0	0	5	0	0	0	0	0
EverEvolvingG.	0	0	0	0	0	65	0	0	0	0	0	0	0
NPChecker	14	3	2	25	0	0	219	128	0	0	0	0	0
Zeus	145	15	10	691	5	0	128	7323	0	0	1	0	0
JiuZhou	0	0	0	0	0	0	0	0	129	0	0	0	2
NotSoSmartC	1	0	0	0	0	0	0	0	0	32	7	0	0
SBcurated	0	10	7	0	0	0	0	1	0	7	129	0	31
SolidiFI	0	0	0	0	0	0	0	0	0	0	0	343	0
SWCregistry	0	0	0	0	0	0	0	0	2	0	31	0	116

Summary on Static Analysis-based Detectors

Not accurate, specialized only on some vulnerabilities... at least fast?

Table 8: Average execution time for each tool.

#	Tools	Execution time	
		Average	Total
1	Honeybadger	0:01:38	23 days, 13:40:00 
2	Maian	0:05:16	49 days, 10:06:15 
3	Manticore	0:24:28	184 days, 01:59:02 
4	Mythril	0:01:24	46 days, 07:46:55 
5	Osiris	0:00:34	18 days, 10:19:01 
6	Oyente	0:00:30	16 days, 04:50:11 
7	Securify	0:06:37	217 days, 22:46:26 
8	Slither	0:00:05	2 days, 15:09:36 
9	Smartcheck	0:00:10	5 days, 12:33:14 
Total		0:04:31	564 days, 3:10:39

Summary on Static Analysis-based Detectors

Not accurate, specialized only on some vulnerabilities... at least maintained?

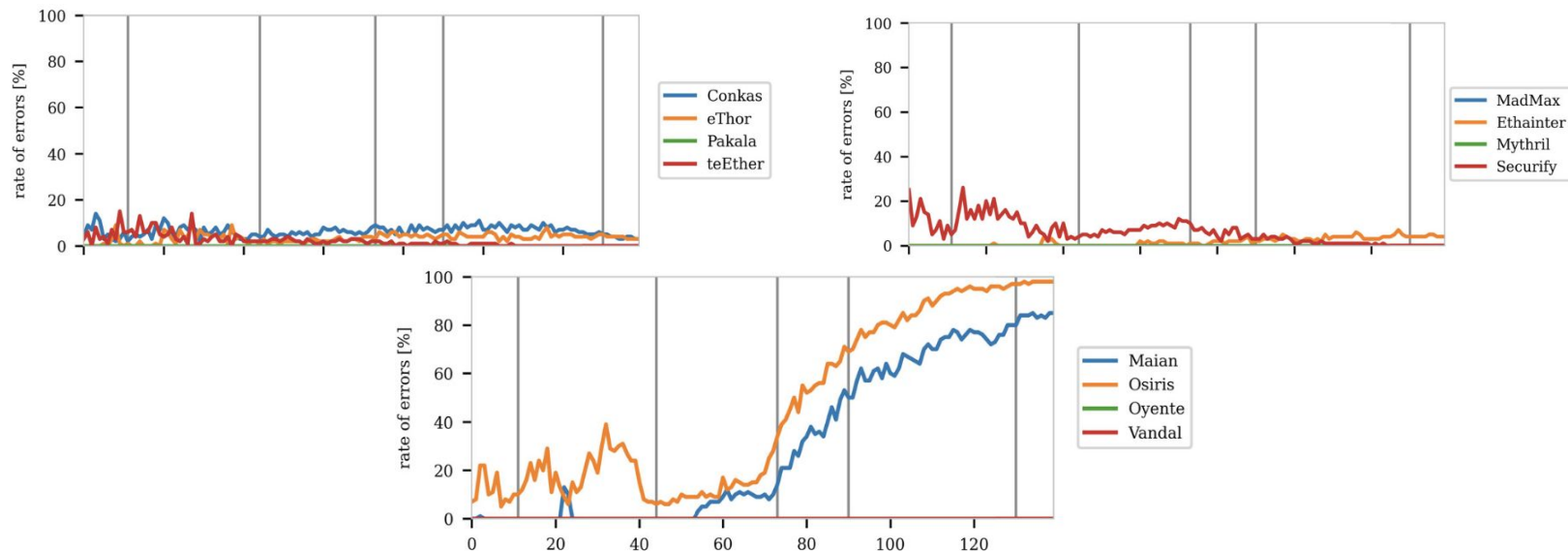


Fig. 6 Tool errors over time. Each data point shows the percentage of errors reported by the tools, in bins of 100k blocks. Mythril, Oyente and Vandal had no errors

Can we do better?



What Kind of Detectors are Available?

Given a fixed set of vulnerabilities, are we able to find them on smart contracts?

Two types of detector:

- Frameworks/tools based on static analysis and formal verification
- **Machine learning based solutions**

Vulnerability Detectors: AI Techniques

ML learning techniques take advantage of existing frameworks for dataset labeling, and then they simply perform classification with techniques such as:

- SVM
- Boosting
- Random Forest
- Decision Tree
- LSTM
- CNN
- GNN
- ...

Machine Learning techniques claim to have higher performance than static analyzers with respect to inference time and accuracy

Vulnerability Detectors: AI Techniques

ML learning techniques:

- SVM
- Boosting
- Random Forest
- Decision Tree
- ...

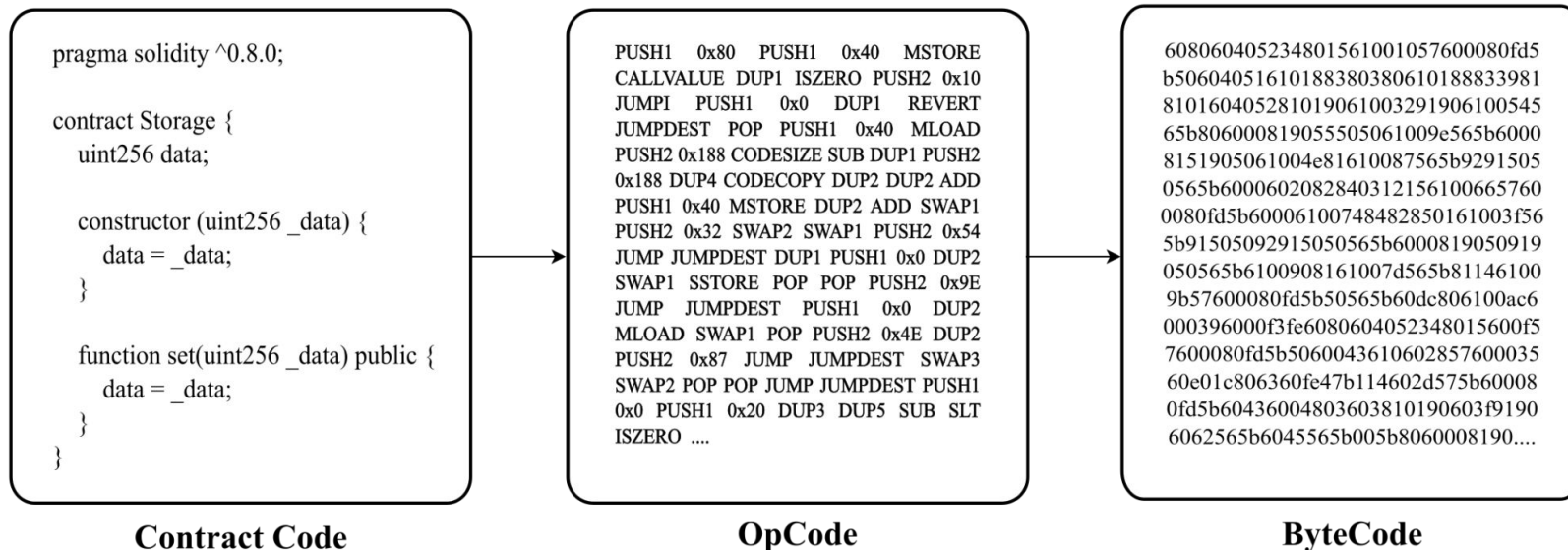
Neural Networks:

- LSTM/GRU
- LLM (ChatGPT)
- GNN (Graphs)
- ...

Smart contracts can be seen as special case of sequential data

A feature extraction step before feeding the input to the classifier greatly improves detection

Ethereum Smart Contracts



Source code is essential for training and testing the model, but a classifier can be designed to work on opcode/bytecode

Our Contributions

1. Collection and categorization of 26 ML-based vulnerability detectors;
2. A consistent mapping from all different definitions across the revised papers to a common taxonomy;
3. Comparison of the detectors according to model, dataset and vulnerabilities considered;
4. Discussion of emerged problems and possible mitigations.

Common Taxonomy

- Mapping of vulnerabilities with different names across the papers
- Inspired by OpenSCV
- Association of risk (color)

Category	Vulnerability	Acronym	Severity	Solved
Unsafe External Calls	Reentrancy	RE	High	
	Dangerous Delegate Calls	DC	High	
	Unchecked Low-level Calls	LLC	High	
	Malicious Library Calls	MLC	High	
	Infinite Loops	IL	Avg	
	Use of Send/Transfer	ST	Avg	
Mishandled Events	DoS (Unbounded Operation)	DOS1	High	
	Mishandled Exception	ME	High	
	Assert Violation	AV	Avg	
Gas Depletion	Multiple Sends	MSS	High	
	DoS	DOS2	Avg	
Bad Programming Practices & Language Weaknesses	Right-to-Left Override	RLO	High	
	Unchecked Return Value	URV	Avg	
	Call Stack Depth	CSD	Avg	EIP-150 2016
	Locked Ether	LE	Avg	
	Incorrect ERC20	ERC20	Avg	
	Inline Assembly	ASM	Avg	
	Unsafe Recast	UR	Avg	
	Variable Shadowing	VSH	Low	v0.6.0
	Block Information	BI	Low	
	Implicit Visibility	IV	Low	
	Complex Pragmas	PRA	Low	
Incorrect Control Flow	Transaction Order Dependency	TOD	High	
	Short Address	SA	Avg	
	Timestamp Dependency	TD	Low	
Arithmetic Issues	Integer Overflow	IOF	High	v0.8.0
	Integer Underflow	IUF	Avg	v0.8.0
	Division by zero	DBZ	Low	v0.4.0
Improper Access Control	Accessible Self-Destruct	ASD	High	
	Tainted Self-Destruct	TSD	High	
	Unrestricted Write	UW	High	
	Suicide Contract	SUC	Avg	
	Tx Origin	TXO	Avg	
	Unprotected Ownership	UO	Avg	

Dataset Statistics

For each paper we can compare vulnerabilities, number of contracts in the dataset, origin of the data and other statistics

		Dataset statistics				Dataset used				Labeling												
Ref	Name	Vulnerabilities	# vulnerabilities	# contracts per class	Balanced	Smote/tomek	#contracts	SmartBugs Wild	Solidifi	ESC	Other	Custom	SecuriFI	Smartcheck	Vandal	Dedaub	Osiris	Solhint	Remix	Manual	Other	Not specified
[10]	-	IOF LLC ASD/TSD/UW/SUC/TXO/UO RE ERC20 DOS2 UR	8	no		6000						✓										✓
[44]	HGAT	RE TD IOF IUF	4	yes	✓	7018		✓														
[32]	-	RE TXO IOF IUF CSD ME TD TOD RE ASM TD LLC TSD	13	no	✓	✓	n.a.	✓		✓												
[46]	-		-	no		160012 +587 +9369		✓		✓	✓			✓								
[9]	Bi-GGNN	BI DC IOF RE SUC SA TD TOD URV	-	no		9369+964 +143+7000		✓	✓	✓												
[77]	MODNN	IOF IUF CSD TOD TD RE AV TXO RE ASM TD LLC	12	no		18000		✓	✓	✓	✓		✓			✓						
[75]	CBGRU	IL RE IOF CSD TD IUF	6	yes	✓	>10000		✓														
[76]	SPCBIG-EC	RE TD IL IOF IUF CSD	6	yes	✓	✓	>10000					✓	✓									
[36]	-	RE RE/DC/LLC/MLC/IL/ST URV ERC20 ERC20 TD	-	yes		3328						✓										✓
[72]	xFuzz	RE TXO DC	3	yes		7391					✓		✓						✓			
[31]	CodeNet	RE LLC TD TXO	4	yes	✓	201376		✓		✓			✓									
[27]	CNN-BiLSTM	TXO URV ERC20 ERC20 TD	-	yes		1733						✓										✓
[6]	Eth2Vec	RE TD ERC20 MSS/DOS2 IV IOF IUF	-	no		95152							✓	✓								
[33]	SmartConDetect	-	23	no		10000							✓	✓								
[57]	ABCNN	RE IOF/IUF/DBZ TD	3	no		8632					✓											
[73]	DeeSCVHunter	RE TD	2	no		40932				✓												✓
[42]	ESCORT	CSD RE MSS ASD DOS1 TSD TOD AV	8	yes	✓	93497						✓				✓						
[70]	Peculiar	RE	1	no		203713		✓				✓									✓	
[29]	-	IOF RE ME BI UO	5	yes		2297058 +32499+24						✓										✓
[41]	CGE	RE TD IL	-	no		40932+4170				✓	✓											✓
[79]	TMP, DR-GCN	RE TD IL	3	no		40932+4170				✓	✓											✓
[50]	BLSTM-ATT	RE	-	yes	✓	4000						✓										✓
[67]	ContractWard	IOF IUF TOD CSD TD RE	6	yes	✓	49502						✓										
[37]	Soliaudit	IOF IUF CSD TOD TD RE AV TXO RE ASM URV TD LLC ASD/TSD	-	yes	✓	17979						✓							✓			
[47]	-	IUF RE ASD LE ERC20 MSS RE TXO URV LLC ME VSH ASM ERC20 PRA	-	yes		1013						✓										
[56]	-	IOF IUF TOD CSD TD RE	6	yes	✓	✓	3000					✓										

Table 3. Comparing different detectors according to considered vulnerabilities and dataset used.

Method Info

Information useful for comparison/design of better methods

	Pre-processing					ML model										Dataset																		
Ref	Name	AST	CFG	word2vec	ngram	network	other	custom	LR	DT	SVM	RF	kNN	Ensemble	NN	CNN	RNN	LSTM/GRU	Transformer	GraphNN	Attention	Bidirectional	Other	# Vulnerabilities	# Contracts	SmartBugs	Wild	Other	Custom	Exec Time	Avg F1-score	Compares to ML		Link
[10]	-	✓	✓			✓								✓						✓				8	6000		✓	n.a.	90.57	[79], [70]	[75]		✓	
[44]	HGAT	✓	✓					✓													✓	✓		4	7018	✓		1.04	84.25					
[32]	-			✓		✓												✓	✓			✓		13	n.a.	✓	✓	n.a.	96.5	[26, 33, 41, 47, 57, 58, 67, 73, 79]				
[46]	-		✓		✓	✓								✓										-	169968		✓	✓	3.27	97	[24, 26, 42]			
[9]	Bi-GGNN	✓	✓			✓												✓		✓	✓	✓		-	17458	✓	✓	n.a.	91.1	[50, 58]				
[77]	MODNN				✓	✓	✓							✓										12	18000	✓	✓	✓	n.a.	94.8	[6, 37, 79]			
[75]	CBGRU			✓		✓									✓		✓					✓		6	>10000	✓		n.a.	89.93	[50, 70, 73, 79]			✓	
[76]	SPCBIG-EC			✓		✓									✓		✓				✓	✓		6	>10000		✓	9.8	90.79	[6, 73, 79]			✓	
[36]	-				✓	✓	✓	✓	✓	✓		✓												-	3328		✓	n.a.	75.3	-				
[72]	xFuzz	✓	✓	✓						✓				✓										3	7391	✓	✓	30	-	-				✓
[31]	CodeNet							✓								✓								4	47518	✓	✓	0.14	97.63	-				
[27]	CNN-BiLSTM					✓										✓		✓				✓		-	1733		✓	n.a.	83.63	-				
[6]	Eth2Vec	✓																				✓		-	95152		✓	0.371	57.5	[47]			✓	
[33]	SmartConDetect					✓	✓													✓				23	10000		✓	n.a.	90.9	[6], [79]				
[57]	ABCNN			✓		✓										✓					✓			3	8632		✓	<1	87.66	-				
[73]	DeeSCVHunter							✓									✓	✓	✓	✓		✓	✓	2	40973		✓	n.a.	83.4	[79]				✓
[42]	ESCORT				✓										✓									8	93497		✓	0.20	95	[26, 30, 58, 67]				
[70]	Peculiar	✓				✓														✓				1	203713	✓	✓	n.a.	92	[79]				✓
[29]	-		✓																		✓			5	>2M		✓	0.47	n.a.	-				
[41]	CGE							✓						✓							✓			-	40932		✓	n.a.	85.43	[79]				
[79]	TMP, DR-GCN	✓																		✓		✓		3	40932		✓	n.a.	77.13					
[50]	BLSTM-ATT			✓		✓												✓			✓	✓		-	4000		✓	n.a.	88.26					✓
[67]	ContractWard			✓							✓	✓	✓	✓	✓									6	49502		✓	4	97	-				
[37]	Soliaudit			✓	✓				✓	✓	✓	✓	✓	✓	✓	✓								-	17979		✓	n.a.	90.4	-				✓
[47]	-	✓				✓					✓	✓	✓	✓	✓									-	1013		✓	<0.001	-	-				
[56]	-			✓							✓	✓	✓	✓										6	3000		✓	4	93	-				

Table 2. Comparing different detectors according to pre-processing technique, models, dataset and results.

Emergred Problems

- Machine Learning-related:

- Scalability
- Interpretability
- Replicability

- Comparison-related:

- Lack of comparison with related work
- Missing results for single vulnerability
- Using different metrics
- Heterogeneity of used datasets
- Vulnerabilities nomenclature
- Results inconsistency

- Usability-related:

- Missing or high inference time
- Vulnerability location
- Source code vs. bytecode
- Risk of vulnerabilities

- Dataset-related:

- Limited number of contracts
- Untreated unbalanced classes
- Unclear dataset creation process
- Labeling methods

Work in Progress

Formalization of vulnerabilities

Collection of representative smart contracts

Creation of benchmark dataset with different methods (generation, manual labelling, semi-supervised, ...)

Testing of new ML models

Conclusion

Static Analyzers are not trustworthy

Manual labeling leads to inconsistencies too

Need for formalization of vulnerabilities and benchmark dataset

Machine learning can still be considered the best option given the low inference time and the wide range of vulnerabilities it is possible to detect

Thank you for your attention

