Blockchain Research @University of Coimbra

Nuno Laranjeiro

cnl@dei.uc.pt

The current blockchain team

- 2 Professors

- 3 PhD students
- 3 MSc students

- Former members
  - 2 MSc students

Main topics

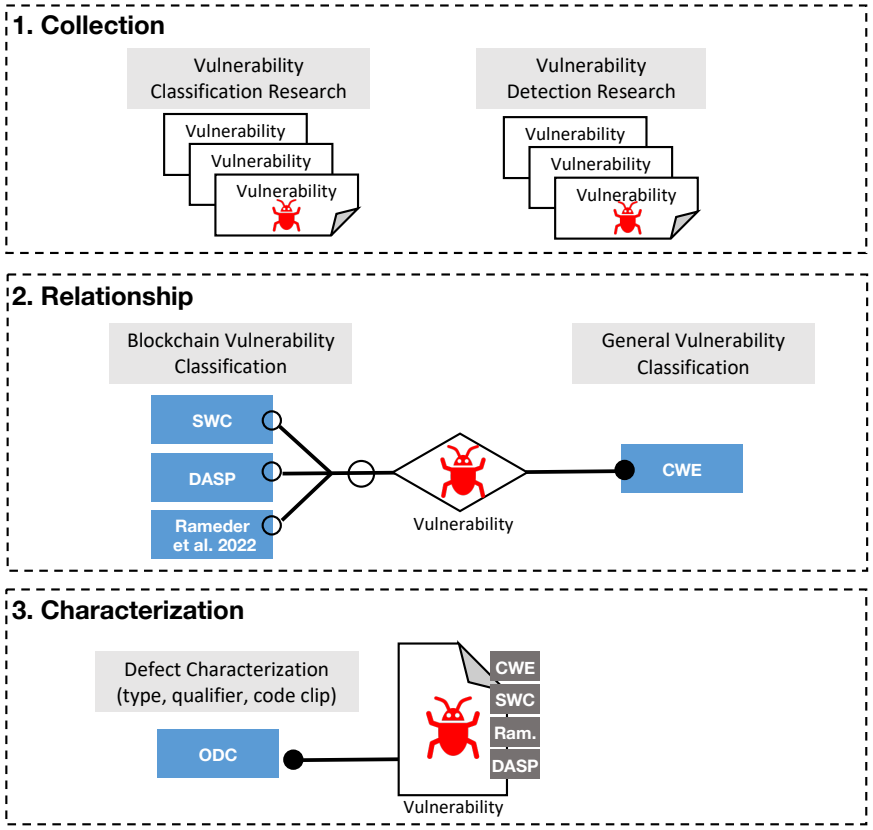
1) Study and systematization of smart contract vulnerabilities
2) Assessment of smart contract vulnerability detection tools
3) Development of a new highly effective vulnerability detection tool
4) Definition of blockchain transaction revocation models

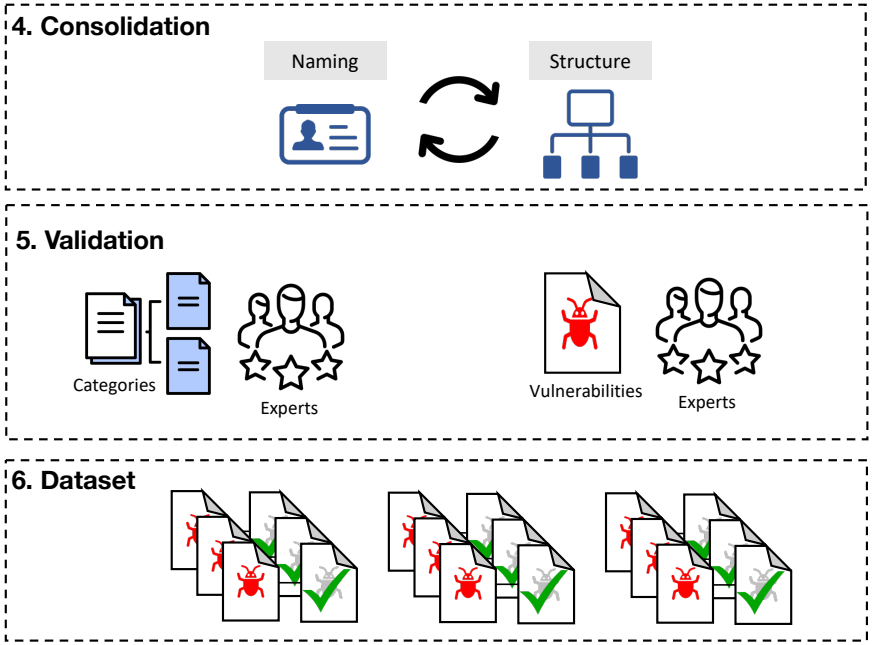
Vulnerability taxonomy - **openscv**

- Vulnerabilities associated with huge financial costs
- Increasing number of new vulnerabilities being discovered
- Huge number of vulnerability detection tools being developed
- Existing schemes (DASP, SWC) outdated, static, insufficient on detail
- **OpenSCV**: An Open Hierarchical Taxonomy for Smart Contract Vulnerabilities – **https://openscv.dei.uc.pt**

Vidal, F.R., Ivaki, N. & Laranjeiro, N. OpenSCV: an open hierarchical taxonomy for smart contract vulnerabilities. Empirical Software Engineering 29, 101 (2024)

Building OpenSCV





Assessing vulnerability detection tools

- Benchmark for assessing and comparing different types of tools
  - Static tools
  - Dynamic tools

- We are working on a vulnerability injector
  - Support for current vulnerabilities
  - Easily extensible to new vulnerabilities
  - Use of generative AI to generate new forms of vulnerabilities

New vulnerability detection tool

- Based on an ensemble of state-of-the-art tools.
- Several criteria involved, possibly conflicting
- Use of machine learning models that work over the data that results from the verification tools execution
  - Added value retrieved from their false alarms
  - Analysis of code metrics related with the software bugs
  - Runtime metrics (e.g., CPU or memory usage)

Transaction revocation

- When a vulnerability is found a smart contract cannot be fixed.
  - It must be terminated and a new one is deployed
- Wrong transactions (incorrect transfer of funds)
- Malicious transactions (illegal content stored in the blockchain)
- Mechanisms for transaction revocation exist
- New mechanism
  - Preventive (use ML for high risk transactions)
  - Reactive (redacting, pruning, erasing)

Analyzing the Impact of Elusive Faults on Blockchain Reliability

CS1– General impact of known faults in diferent types of contracts

CS2 – Effectiveness of fault detection tools

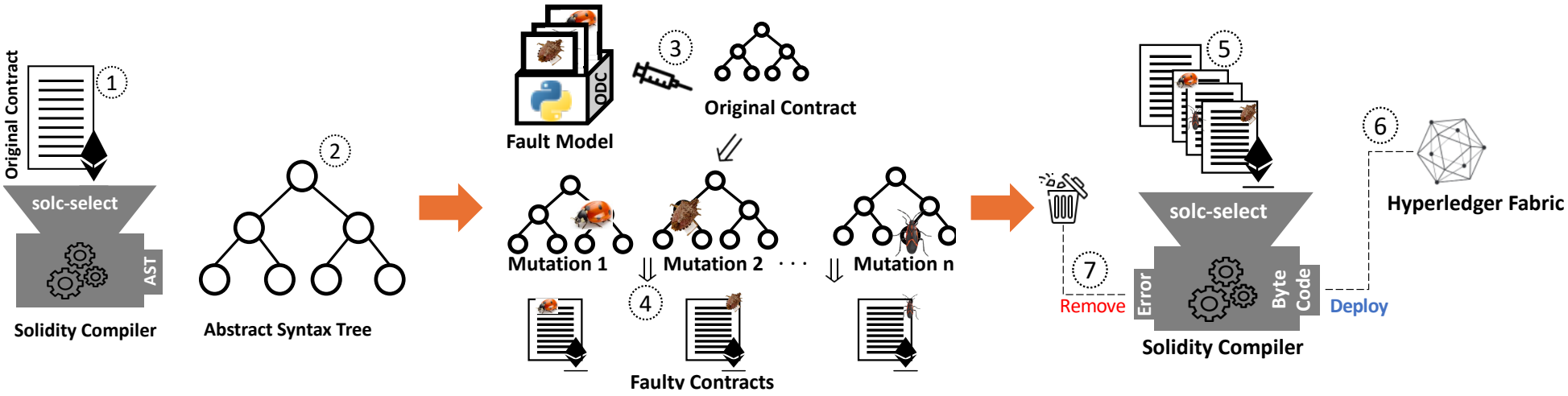
CS3 – Impact of faults that escape detection

Smart contract faults

- Mapped to SWC
- 400 contracts
- 15,949 faulty contracts

Defect Class	Defect Nature	Defect Name	Defect Identifier
Assignment	Missing	Initialization of Storage Variables/Pointers (Uninitialized Storage Pointer) **(MISP)**	A_MISP
		Initialization of Local Variable **(MILV)**	A_MILV
		Initialization of State Variables **(MISV)**	A_MISV
		Constructor **(MC)**	A_MC
		Compiler Version **(MCV)**	A_MCV
	Wrong	Arithmetic Expression Used In Assignment **(WVAE)**	A_WVAE
		Integer Sign **(WIS)**	A_WIS
		Integer Truncation **(WIT)**	A_WIT
		Value Assignment With Too Many Digits **(WVATMD)**	A_WVATMD
		Value Assigned To Contract Address **(WVAA)**	A_WVAA
		Constructor Name **(WCN)**	A_WCN
		Variable Type (e.g., byte[]) **(WVT)**	A_WVT
		Declaration Of Invariant State Variable **(WDISV)**	A_WDISV
		Variable Name (Variable Shadowing) **(WVN)**	A_WVN
Checking	Missing	"require" On Transaction Sender **(MRTS)**	CH_MRTS
		"require" On Input Variable(s) **(MRIV)**	CH_MRIV
		"require" OR Subexpression On Transaction Sender **(MROTS)**	CH_MROTS
		"require" OR Subexpression On Input Variable(s) **(MROIV)**	CH_MROIV
		"require" AND Subexpression On Transaction Sender **(MRATS)**	CH_MRATS
		"require" AND Subexpression On Input Variable(s) **(MRAIV)**	CH_MRAIV
		Check On Gas Limit **(MCHGL)**	CH_MCHGL
		Check On Arithmetic Operation **(MCHAO)**	CH_MCHAO
		Check On Suicide Functionality **(MCHSF)**	CH_MCHSF
	Wrong	"require" For Authorization (Authorization Through tx.origin) **(WRA)**	CH_WRA
Interface	Missing	Visibility modifier of state variables (implicit visibility) **(MVMSV)**	I_MVMSV
		Function Visibility Modifier **(MFVM)**	I_MFVM
	Wrong	Visibility (public) for private/internal function **(WVPF)**	I_WVPF
Algorithm	Missing	"if" construct on transaction sender plus statements **(MITSS)**	AL_MITSS
		"if" construct on input variable(s) plus statements **(MIIVS)**	AL_MIIVS
	Wrong	Use of require, assert, and revert **(WRAR)**	AL_WRAR
		Exception Handling **(WEH)**	AL_WEH
	Extraneous	Continue-statements in do-while-statements or for **(ECSWS)**	AL_ECSWS
Function	Missing	Withdraw function **(MWF)**	F_MWF
		Inheritance **(MINHERITANCE)**	F_MINHERITANCE
	Wrong	Inheritance and inheritance Order **(WIO)**	F_WIO
	Extraneous	Inheritance **(EINHERITANCE)**	F_EINHERITANCE

Fault injection approach

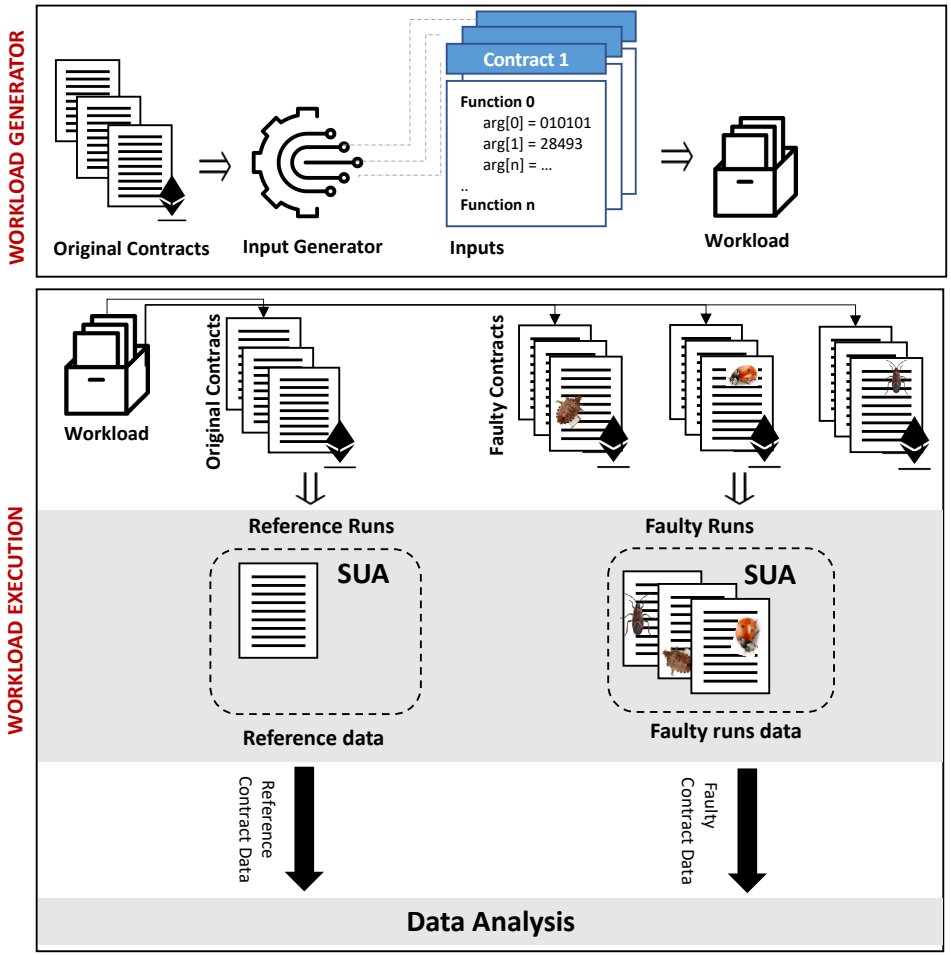


- Execution:

Up to 1500 calls per function

- Metrics:

transaction execution time, reverted transactions, CPU/memory usage)

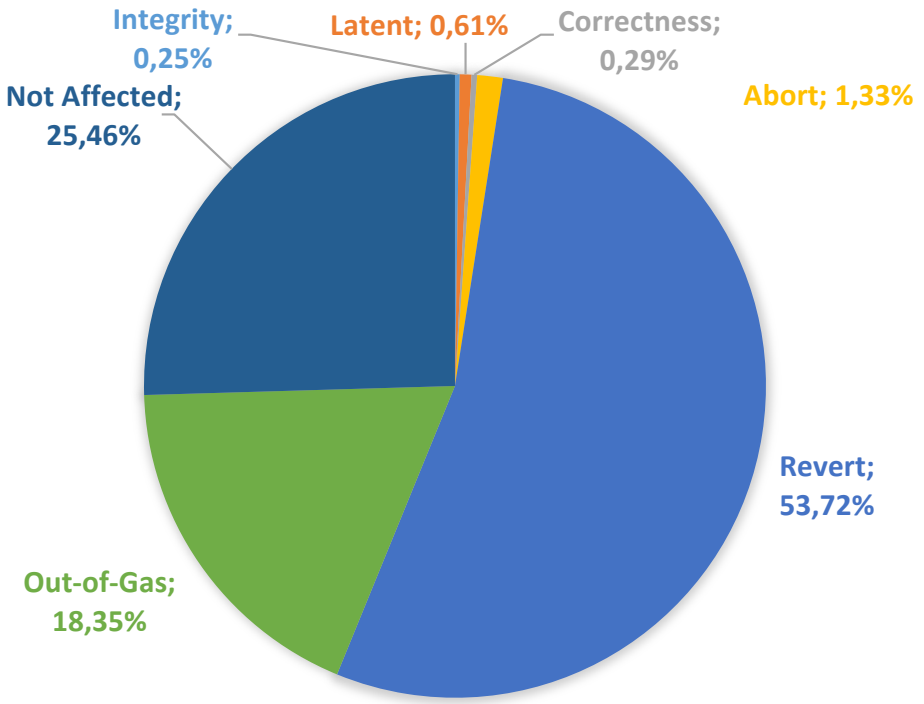


Failure modes

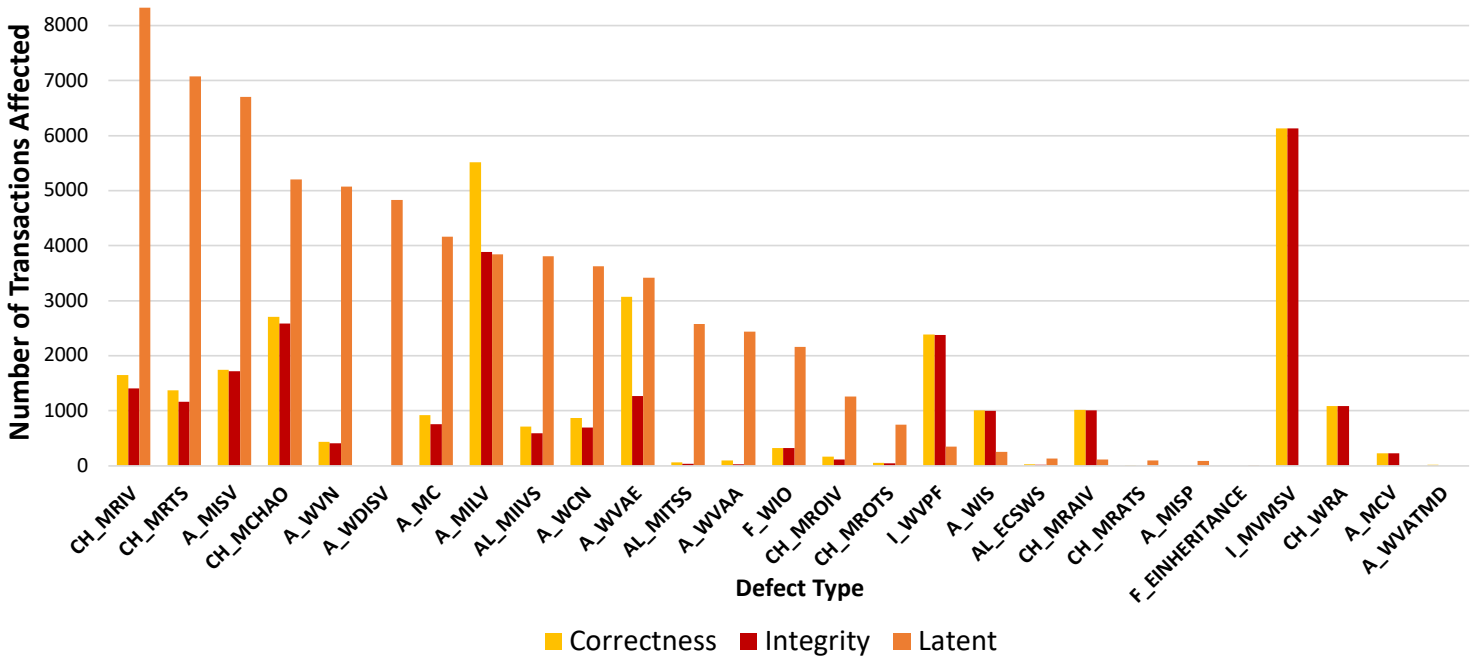
Failure Modes	Transaction not concluded	Incorrect return value or transaction result	Incorrect ledger state
Abort	●		
Revert	●		
Out-of-gas	●		
Correctness		●	
Integrity		●	●
Latent integrity			●

CS1 – Faults' impact overview

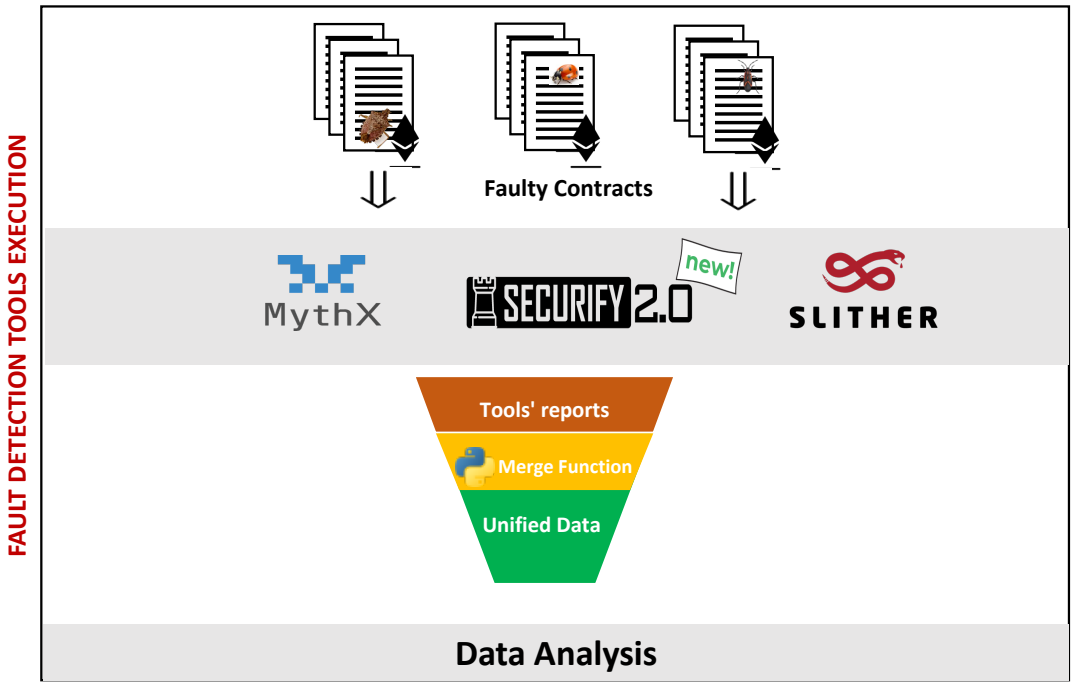
- **>10M transactions (25% no effect)**
- Revert and out of gas are prevalent
- Most critical ones (integrity, latent, correctness) <2.5%



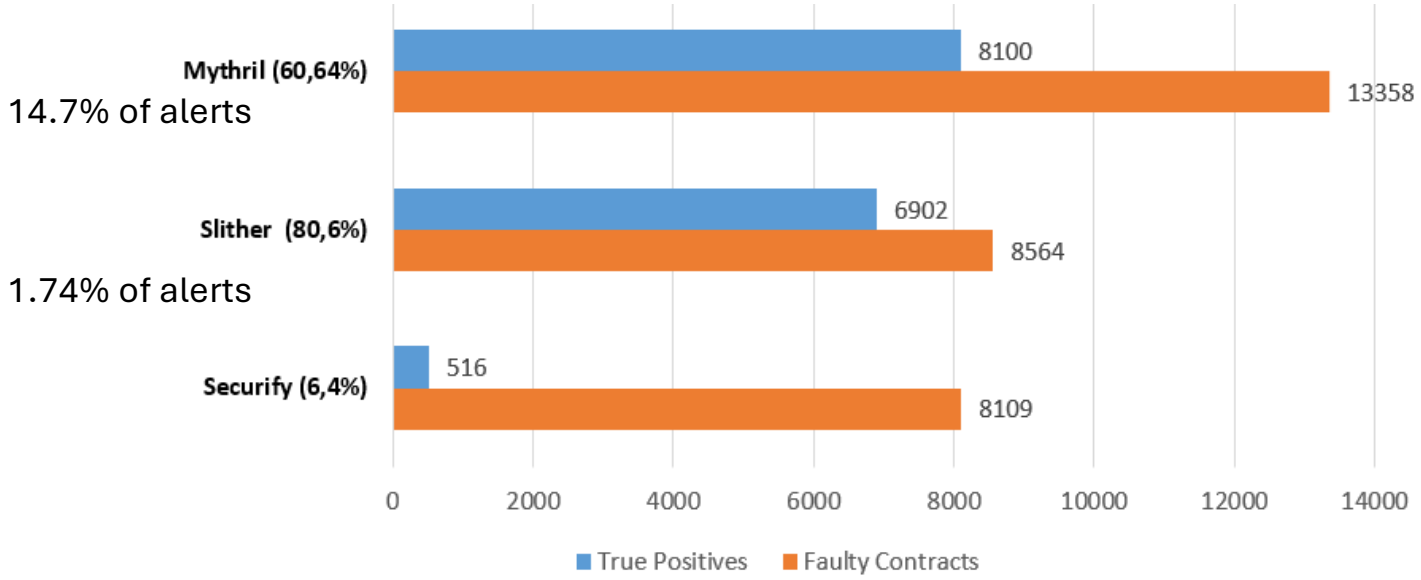
CS1 – Faults associated with more severe failures



CS2 – Tools' effectiveness

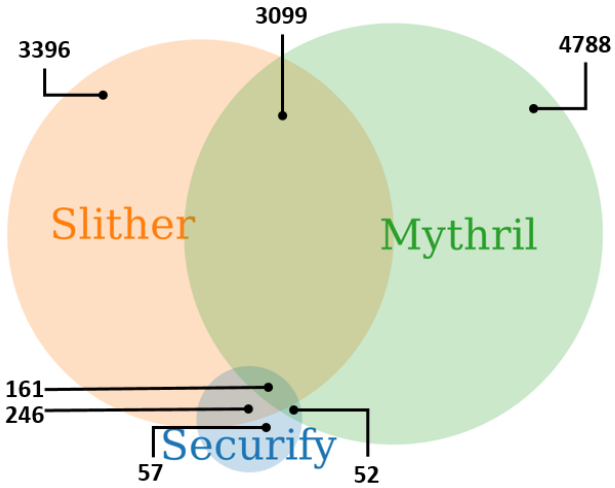


CS2 – Tools effectiveness

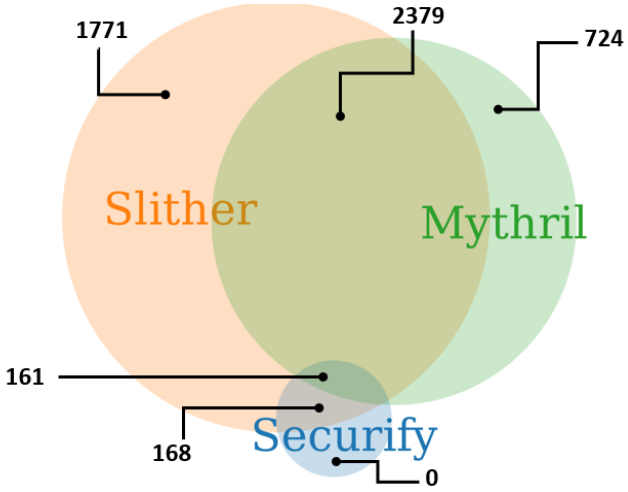


- There are other properties, e.g., speed: Security -> Slither -> Mythril

CS2 – Intersecting detection capabilities



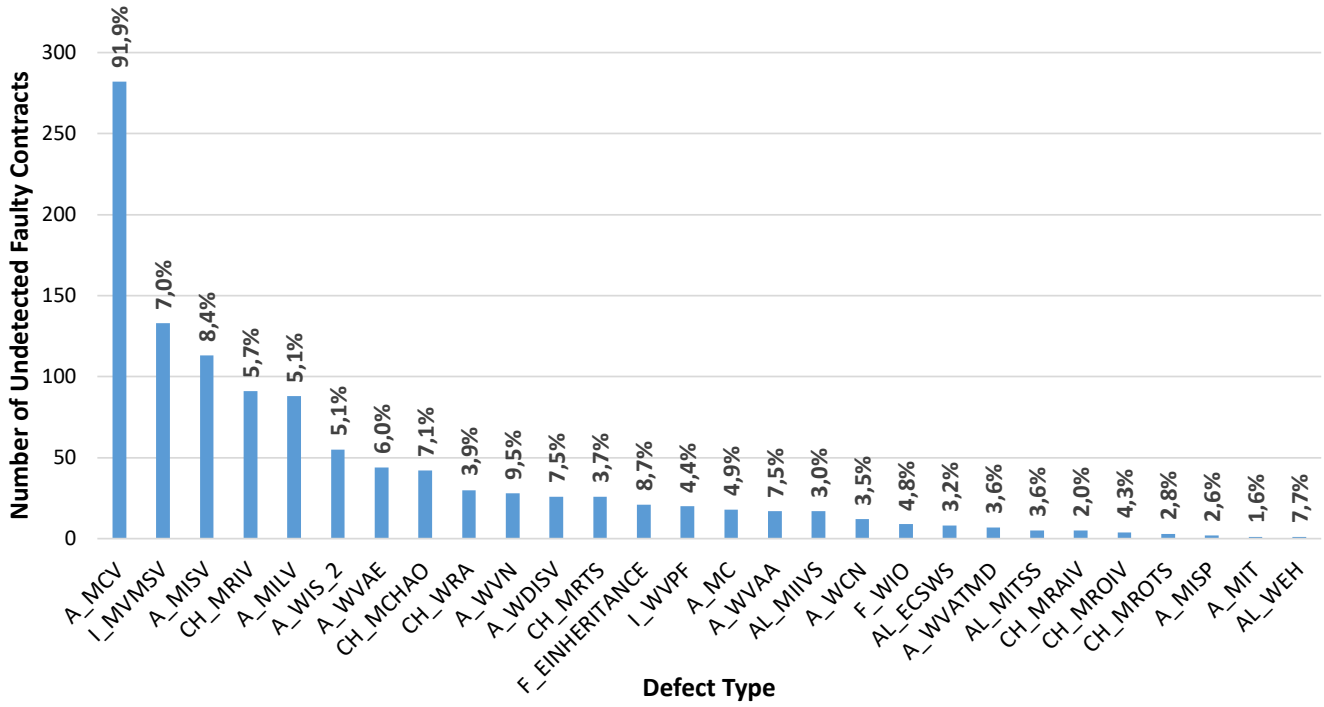
Union of all faults the tools should detect



Faults common to the 3 tools

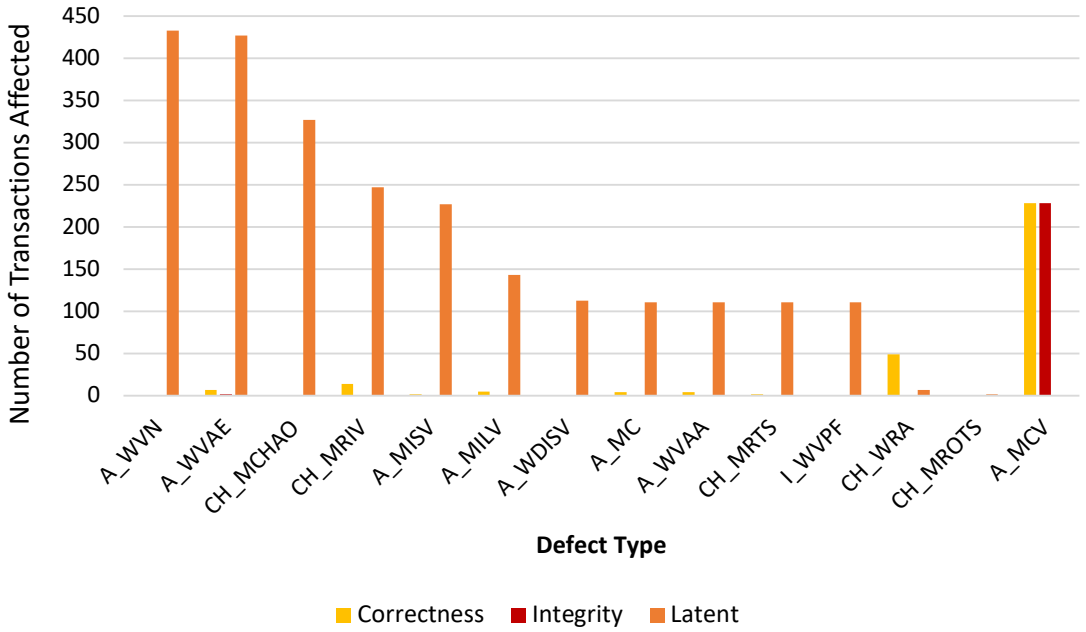
CS3 – Undetected faulty contracts

- 9% of the faulty contracts escape detection by any of the tools



CS3 – Impact of the most severe faults

- Faults generating the most severe failures are either of type assignment or checking



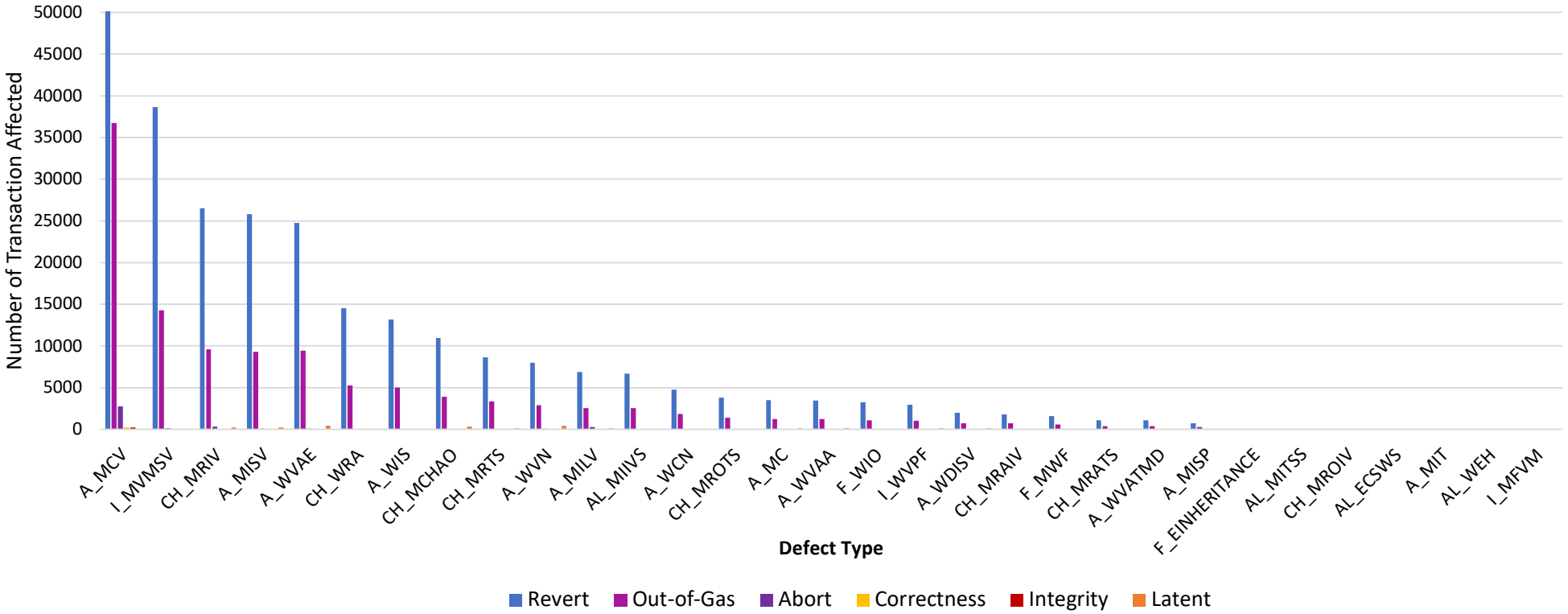
A few highlights

- No failures in about ¼ of the faulty contracts
- **Revert failures** in about ½ of the faulty contracts
- **Out-of-gas** in about 1/5 of the faulty contracts
- *missing require on input variable*, the third fault most frequently injected, is responsible for most *Latent failures*, which is the most severe failure mode
- Faults generating the most severe failures are either of type assignment or checking
- ¾ of the fault types escaped detection (elusive) and are associated with severe failures

Questions?

Nuno Laranjeiro
cnl@dei.uc.pt

CS3 – Affected transactions



CS3 – Undetectable faulty contracts impact

- Lower number of transactions with no Effect (11% vs 25%)