2023 International Conference on Blockchain and Trustworthy Systems

The Best of Both Worlds: Integrating Semantic Features with Expert Features for Smart Contract Vulnerability Detection

Xingwei Lin[1], Mingxuan Zhou[2], Sicong Cao[2], Jiashui Wang[1,3], and Xiaobing Sun[2]

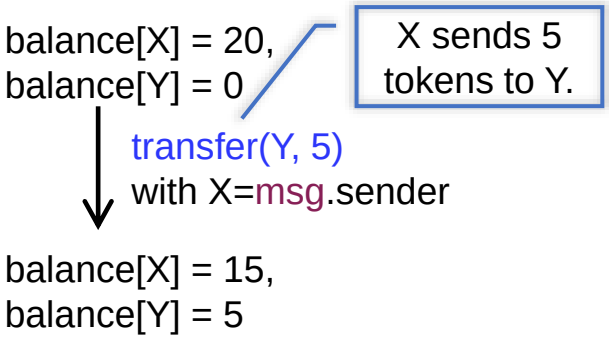
[1]Ant Group
[2]Yangzhou University
[3]Zhejiang University

Smart Contract

- Digital contract written in programming languages.
  - e.g., Decentralized Finance, food supply chain (IBM Food Trust).
- Send transactions by invoking functions in smart contracts.

```
1  function transfer (address to, uint value) public
2  returns (bool) {
3    require (balance[msg.sender] >= value);
4    balance[msg.sender] -= value;
5    balance[to] += value;
6    return true;
7  }
```

Solidity Function



Importance of Securing Smart Contracts

- Immutable once deployed.
- Huge financial damage once exploited.

(2016)

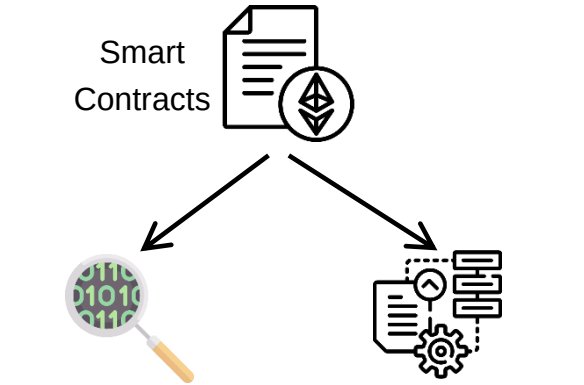
(2018)

(2017)

(2020)

Existing Solutions

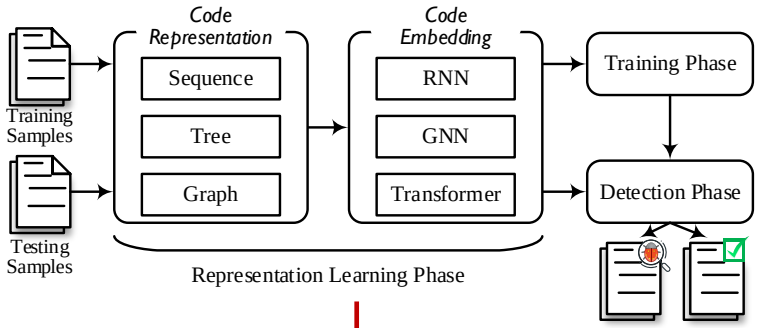
Traditional Approaches



High FPs

Path Explosion

Learning-based Approaches

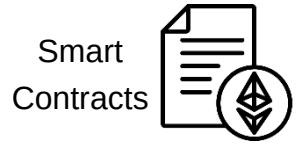


Promising while unsatisfactory

Existing Solutions

Traditional Approaches

Smart Contracts



Static Analysis

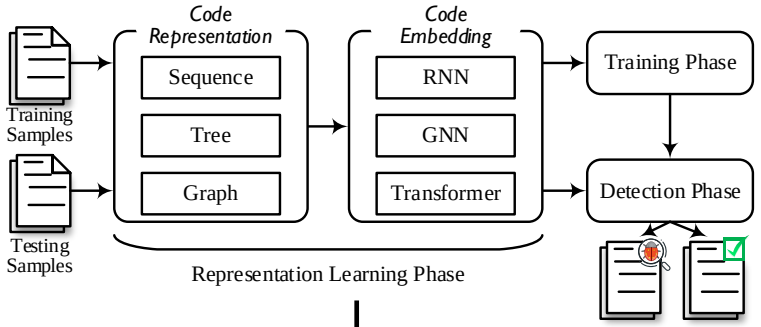
High FPs

Prior Expert Knowledge

Symbolic Execution

Path Explosion

Learning-based Approaches



Promising while unsatisfactory

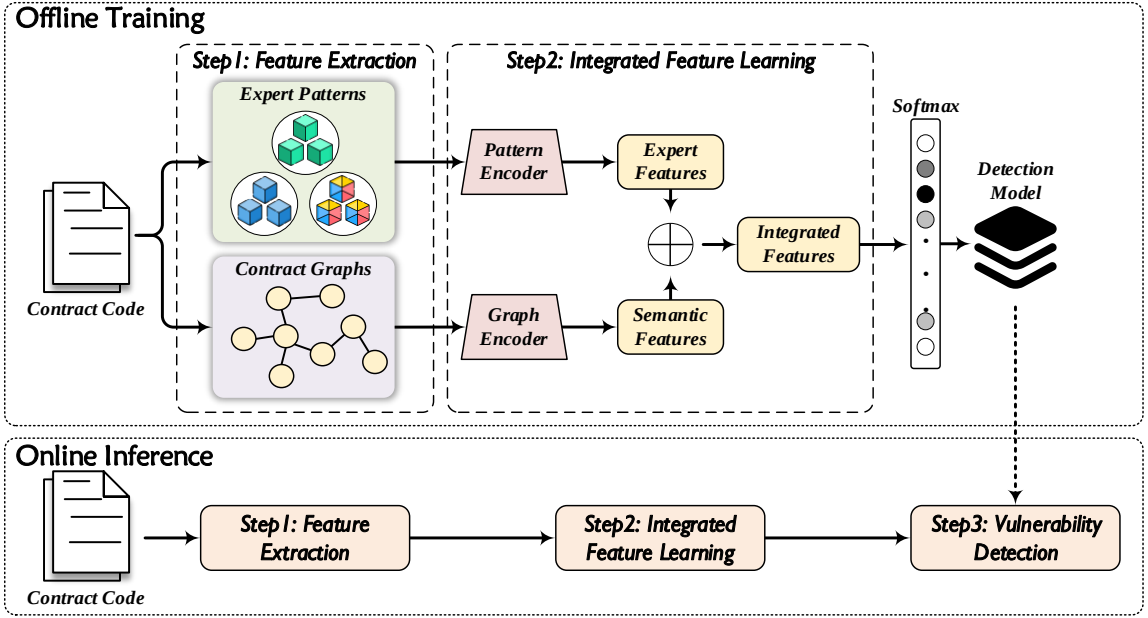
Implicit Vulnerability Patterns

Can we combine the strengths of pattern- and DL-based approaches while mitigating each others' weaknesses?

Our approach: SmartFuSE

Static Analysis + Graph-based Representation Learning

- Extracting expert features and semantic features from statistical data and source code.
- Fusing expert features and semantic features for vulnerability detection



Detail: Feature Extraction

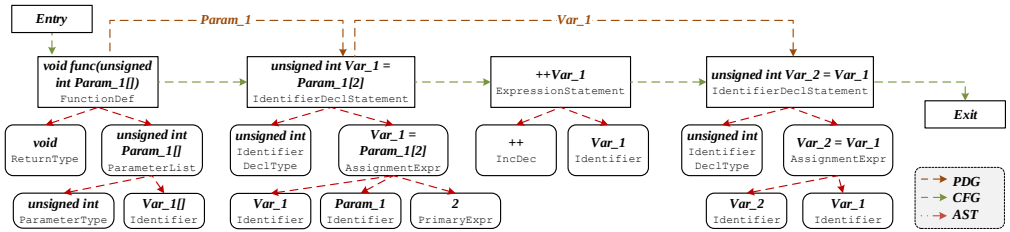
Common expert features

Vulnerability Type	Expert pattern	Security-Critical Operations
Reentrancy	enoughBalance callValueInvocation balanceDeduction	call.value invocation a function that contains call.value the variable: correspond to user balance
Timestamp dependence	timestampInvocation timestampAssign timestampContaminate	block.timestamp invocation block.number invocation a variable: affect critical operation
Infinite loop	loopStatement loopCondition selfInvocation	for while self-call function

```
1  function withdraw(uint amount) public {
2     if (credit[msg.sender] >= amount) {
3       require (msg.sender.call.value(amount)());
4       credit[msg.sender]-=amount;*str1;
5     }
6  }
```

```
1  function() payable {
2     if (msg.sender >= 10 finney) {
3          bytes20 bH = ripemd160(block.timestamp);
4          if (bH[0] ==0) {
5            uint8 bM = ((bH[1] & 0x01 !=0)? 1:0);
6            uint256 bTI = (msg.sender * 100) * bM;
7          }
8     }
9  }
```

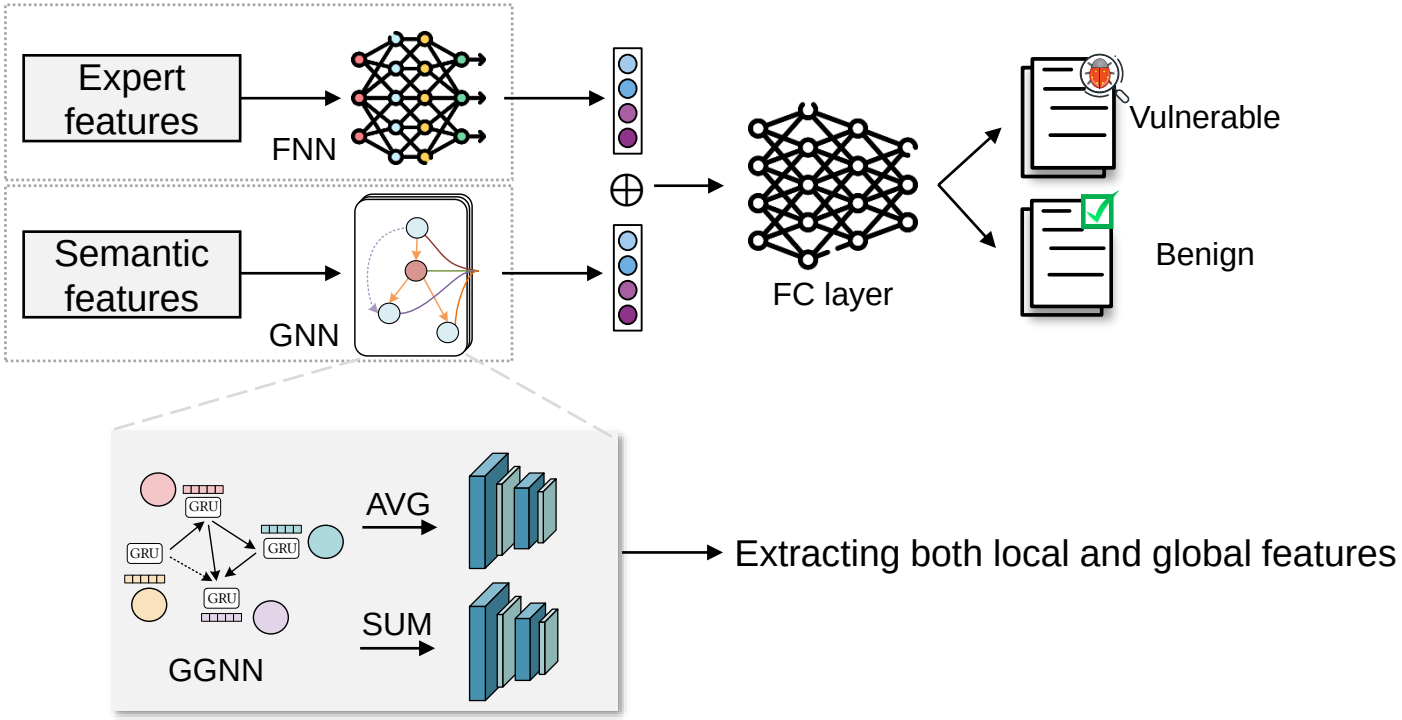
Code-centric semantic features



Slice Criteria	Example
block info	*block.number, block.gaslimit*
delegatecall	*address.delegatecall*
arithmetic operations	$*, +, -$
external function call	*address.call*
selfdestruct operation	*selfdestruct*
input address parameters	*address input_address*
block timestamp	*block.timestamp*
low level call operation	*address.call*

Program slicing

Detail: Integrated Feature Learning



Evaluation Setup

- **Benchmark:** Ethereum Smart Contracts (ESC) + (VNT chain Smart Contracts)
  - https://github.com/Messi-Q/Smart-Contract-Dataset

- Compared with 5 analysis-based approaches and 2 DL-based approaches
  - **Analysis-based:** Oyente, Mythril, Smartcheck, Securify, Slither
  - **DL-based:** Peculiar, TMP

- Used 4 common evaluation metrics: Accuracy, Precision, Recall, and F1-score

- 10-fold cross validation

Evaluation Results

RQ1: Effectiveness

Method	Accuracy	Precision	Recall	F1-score
Oyente	57.3	41.1	42.8	41.9
Mythril	53.9	64.7	36.4	46.6
Securify	50.5	53.2	55.2	54.2
Smartcheck	37.8	59.4	43.5	50.2
Slither	61.9	63.1	58.4	50.7
Peculiar	82.7	55.2	41.6	47.4
TMP	85.0	83.9	66.5	74.2
SmartFuSE	**91.4**	**88.6**	**94.3**	**91.4**

Result: Overall, SmartFuSE outperforms all of the five referred analysis-based detectors and two DL-based approaches.

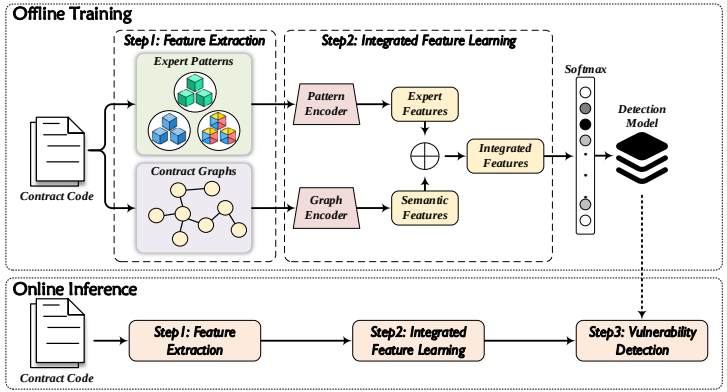
RQ2&3: Ablation Study

Setting	Accuracy	Precision	Recall	F1-score
Expert Features	86.9	84.3	90.2	87.1
Semantic Features	83.2	81.5	88.6	84.9
SmartFuSE	**91.4**	**88.6**	**94.3**	**91.4**

Setting	Accuracy	Precision	Recall	F1-score
Sum Pooling	75.7	74.7	83.6	87.1
Avg Pooling	80.1	78.4	87.4	84.9
Global Attention Pooling	83.8	81.6	89.5	87.1
Self Attention Pooling	87.6	84.2	92.6	84.9
SmartFuSE	**91.4**	**88.6**	**94.3**	**91.4**

Result: The combination of expert features and semantic features, as well as our graph representation learning with hybrid pooling layer, contribute significantly to the performance of SmartFuSE.

Thanks for listening!



蚂蚁集团 ANT GROUP

揚州大學 YANGZHOU UNIVERSITY

浙江大學 ZHEJIANG UNIVERSITY