An Efficient and Scalable Smart Contract Checker

Shangyu Xie

Vulnerabilities of Smart Contract

1. **Integer Overflow**. Integer overflow vulnerabilities occur when a computed value is too large for the type attached to the value.
2. **External Call To Fixed Address.** An external contract can take over the control flow due to an unchecked call of the return value.
3. **Exception State** (Assert Violation). A bug exists in the contract or that assert is used incorrectly.

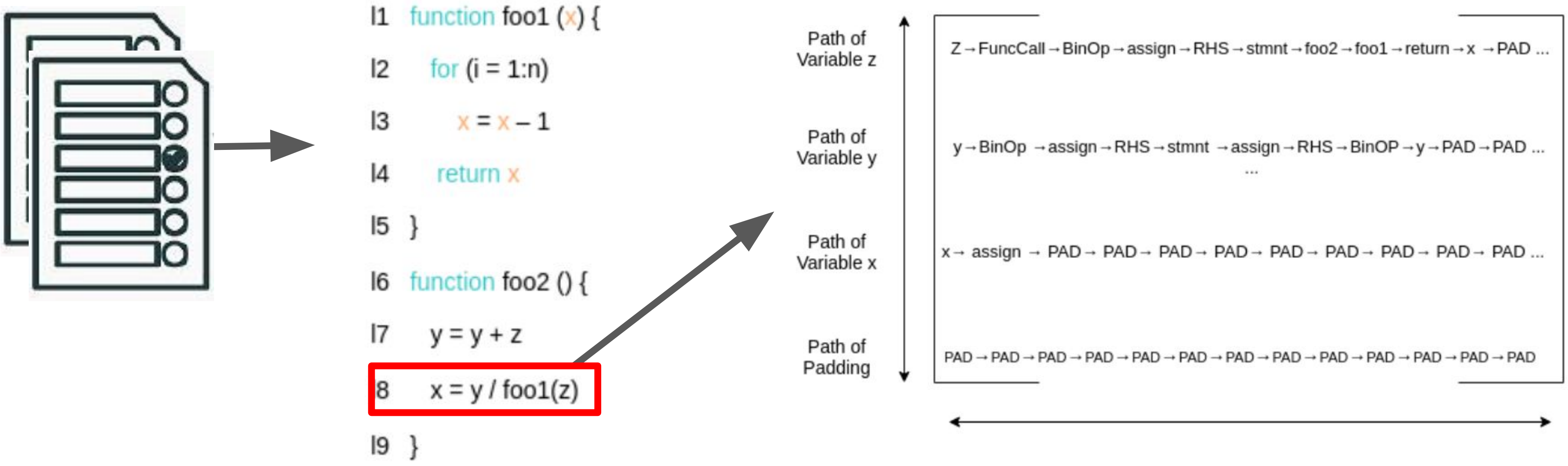
https://mythx.io/detectors/
https://swcregistry.io/docs/SWC-101
https://swcregistry.io/docs/SWC-104

Modeling Smart Contract-Abstract Syntax Tree

1. Program main components (tokens)
    a. Variables
    b. Operators
    c. Function calls
2. Control and Data Path (**CDP**)
    a. A path corresponding to one token is the part of the AST origination from the corresponding node and terminating at the previous usage of the considered token.
3. **Line-level** Representations to model both data and control dependencies.
    a. Path-attention Model
    b. Attention can be summarized as a vector of importance weights describing the power of each token in terms of discrimination of vulnerabilities.

code2vec: Learning Distributed Representations of Code

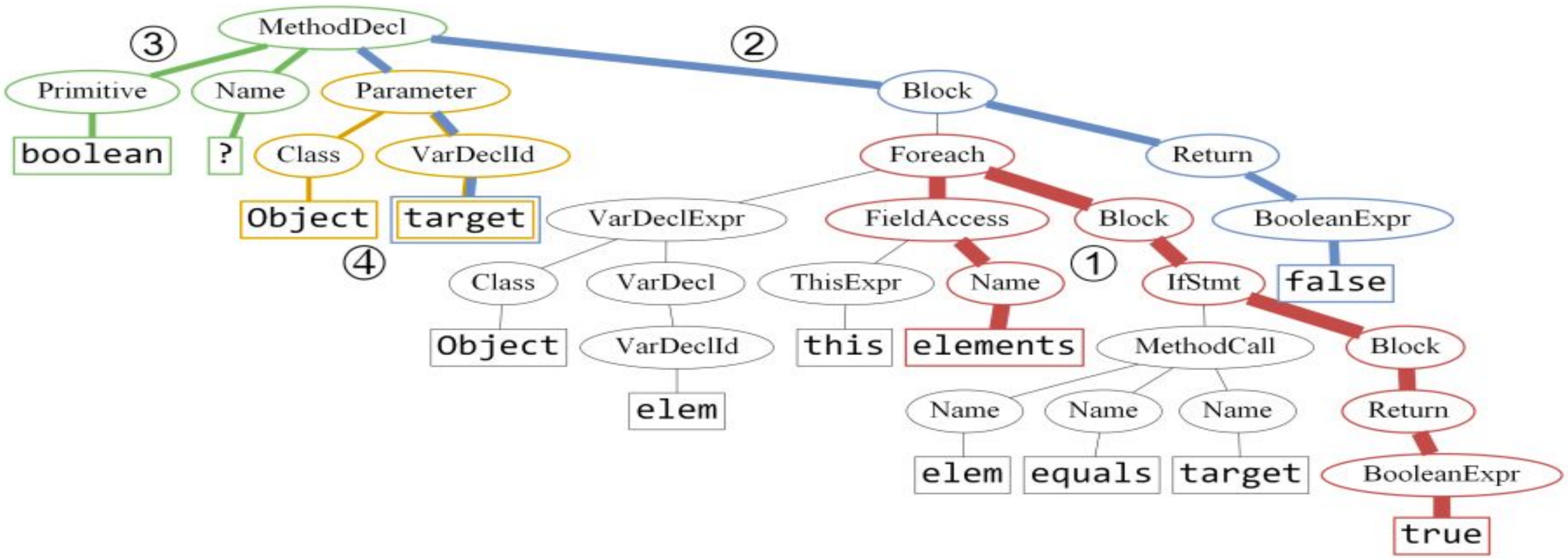
Feature Embedding for Smart Contract



2-dimension for each code lines

1. Extract AST representations
2. CDP for variables

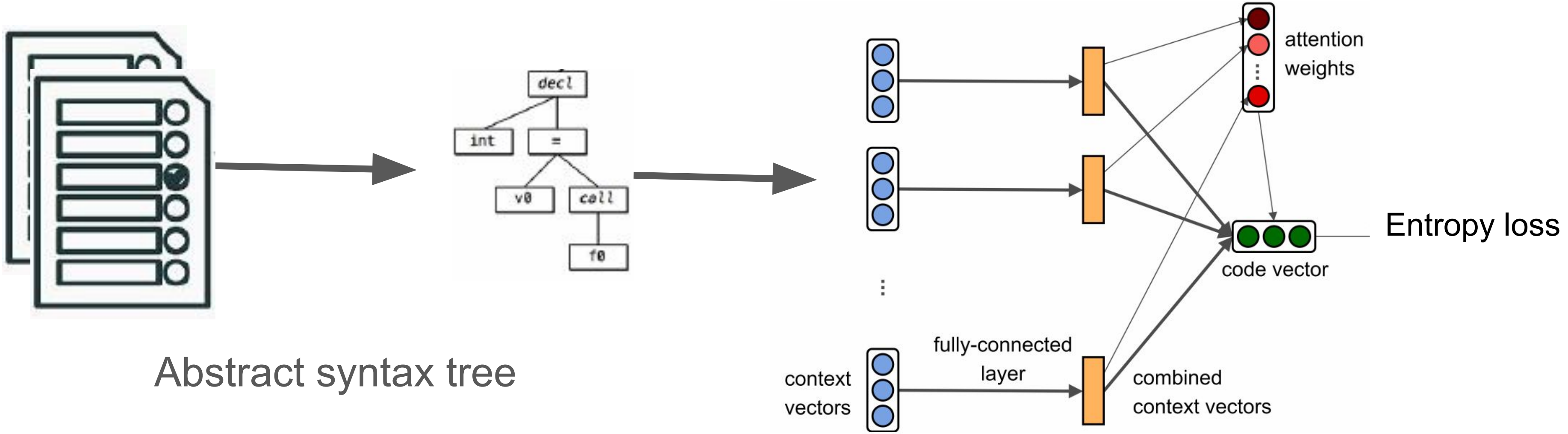
Path Attention Model (to be extended)



The top-4 attended paths on the AST, learned via a model. The width of each colored path is proportional to the attention it was given.

Training Architecture

Line-level Embedding Representation



Abstract syntax tree

Added Attention Layer in Training

Unbalanced Dataset

1. Bugged codes segment is a very small portion of dataset
2. False Positive Rate (the unavailability of sufficient examples of bugged smart contract, which can be usually treated as low likelihood pattern detection)

Example. 100 Animal of Pictures for Classification. Only 5 are sheeps, the remaining 95 are pigs.

Multiple Instances Learning (MIL)

1. Objective Function
    a. standard supervised classification problems using support vector machine (SVM); k is number of instances; w is the training mode to learn, y_i is the label

$$\min_{\mathbf{w}} \quad \frac{1}{k}\sum_{i=1}^{k} \overbrace{max(0, 1 - y_i(\mathbf{w}.\phi(x) - b))}^{①} + \frac{1}{2}\left\|\mathbf{w}\right\|^2$$

    b. Bugged codes’ location can be unknown

Real-world Anomaly Detection in Surveillance Videos, CVPR’17

Deep MIL Ranking Model

Challenges

1. not obvious how to assign 1/0 labels to the bugged smart contract and classify which is which.
2. low likelihood pattern detection

Idea

1. Scoring function (f) to enable the normal smart contract obtains lower bug scores than the bugged smart contract

$$\max_{i \in \mathcal{B}_a} f(\mathcal{V}_a^i) > \max_{i \in \mathcal{B}_n} f(\mathcal{V}_n^i)$$

Ranking Model (Continue)

$$l(\mathcal{B}_a, \mathcal{B}_n) = \max(0, 1 - \max_{i \in \mathcal{B}_a} f(\mathcal{V}_a^i) + \max_{i \in \mathcal{B}_n} f(\mathcal{V}_n^i))$$

$$+\lambda_1 \overbrace{\sum_i^{(n-1)} (f(\mathcal{V}_a^i) - f(\mathcal{V}_a^{i+1}))^2}^{①} + \lambda_2 \overbrace{\sum_i^{n} f(\mathcal{V}_a^i)}^{②},$$

1. ① indicates smoothing part; ② indicates the sparse item
2. the error is back-propagated from the maximum scored smart contract in both bugged and normal bag of smart contract

expect that the learning network will learn a generalized model to predict high scores for bugged smart contracts, the score of range [0, 1]

Further Classification/detection (Symbolic Execution)

1. ML model classify the bugged code quickly and efficiently
2. The bug type need to be classified further
    a. Use detection method directly
    b. Design another classification model (for classifying the type)

Implementation

1. Evaluation Metric

$$Precision = \frac{true\,positive}{true\,positive + false\,positive}$$

$$Recall = \frac{true\,positive}{true\,positive + false\,negative}$$

$$F1\text{-}score = 2 \times \frac{Recall \times Precision}{Recall + Precision}$$

$$Accuracy = \frac{true\,positive + true\,negative}{Num.\,of\,total\,samples}$$

2. Labeling Datasets (choose source datasets and fitable tools)
3. Benchingmarks. (Other Models, random forest trees.)