Fuzzy Hashing of EVM Bytecode

Raphael Nußbaumer

October 2020

1. Algorithmen
2. Vergleichsmethoden,
3. Resultat der Evaluation
4. Dabei sind auch jene Ansätze interessant, bei denen nichts herauskommt.

1 Abstract

This thesis explores fuzzy hashing methods to compute similarities between EVM byte-codes. For this purpose a set of python utilities was implemented and an data-set for evaluation generated.

2 Goal

The goal was to compare different similarity measures and pre-processing steps and evaluate how they respond to changes in the codes. This should be accomplish with a set of reusable python utilities.

3 Data sets

3.1 Solc Versions and Options

To evaluate the similarity measures I selected a set of 12 solidity smart contracts and compiled them with different solc version and compiler options. Necessary changes where made to the source code to ensure compatibility with the various solc versions.

The following solc version where used:

1. `0.5.16`
2. `0.6.12`
3. `0.7.6`
4. `0.8.4`

To evaluate the effect of optimization I applied the following options:

1. `{ enabled: false, runs: 200 }`
2. `{ enabled: true, runs: 0 }`
3. `{ enabled: true, runs: 200 }`
4. `{ enabled: true, runs: 999999 }`

Further the contracts where compiled with ABI encoding v1 and v2.

4 Pre-Processing

4.1 Segmentation

4.2 Skeletonization

4.3 Op-Code Filtering

5 Hashing Methods

5.1 ssdeep - Context Triggered Piecewise Hashes (CTPH)

5.2 Op-Code Frequency

5.3 LZJD - Binary-Hashing

This is how a cite[1] looks like.

6 Results

References

[1] Andrew H Sung et al. "Static analyzer of vicious executables (save)". In: *20th Annual Computer Security Applications Conference*. IEEE. 2004, pp. 326–334.