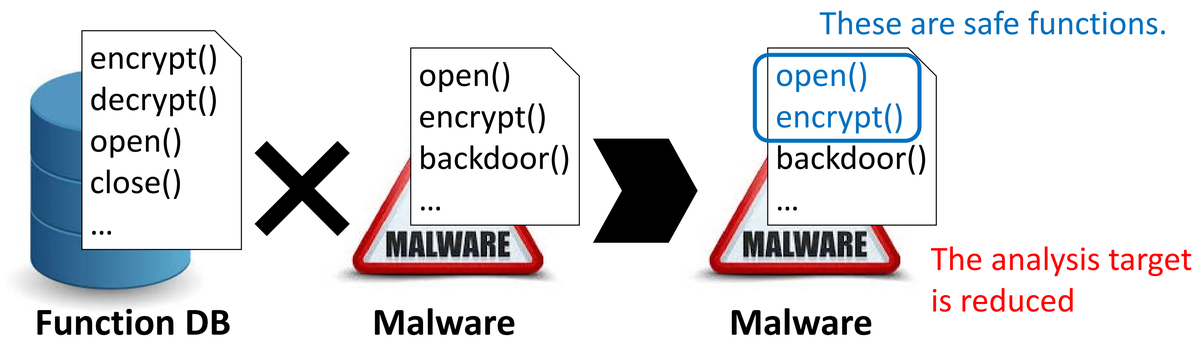
Measuring Similarity of Binary Programs using Hungarian Algorithm

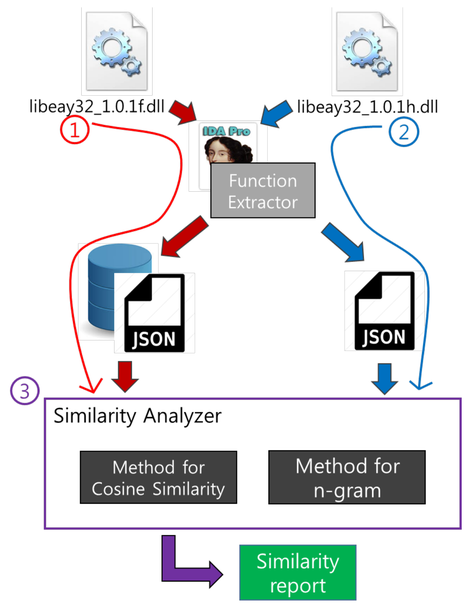
Yeongcheol Kim*, Seokwoo Choi** and Eun-Sun Cho*
*Department of Computer Science and Engineering, Chungnam National University, Korea
{aree91,eschough}@cnu.ac.kr
**NSR, Korea
seogu.choi@gmail.com

Motivation



We aim to notice the binaries of well-known functions which are safe in the sense of malicious behaviors, so as to enable analyzing suspicious codes effectively by excluding spotted functions(like open-source functions) in the analysis process.

Similarity Analysis Flow



① First, extract function information from binary program which is database.

② Next, extract function information from analysis target.

③ Finally, conduct analysis for similarity measure.

Proposed Analysis Method

Step 1) After generating an n-gram set from the instructions that constitute the function, a matrix is generated with the length of the LCS(Longest Common Subsequence) calculated by applying LCS algorithm to the elements between the sets.

Step 2) Apply Hungarian algorithm* to the generated matrix, and calculate the indices of the matrix that constitute the largest sum of elements that do not select rows and columns as duplicates.

Step 3) Detect the same function more precisely using indices calculated through Method I and Method II

*Patrick P.F. Chan and Christian Collberg, *A Method to Evaluate CFG Comparison Algorithms*, QSIC 2014

ex) n=3, threshold=0.6

step1 Function A : [push, mov, push, push, pop, pop, ret]
➔ [push, mov, push, push, pop]

n-gram set of Function A
- 0 : [push, mov, push]
- 1 : [mov, push, push]
- 2 : [push, push, pop]

Function B : [push, mov, push, pop, ret]

n-gram set of Function B
- [push, mov, push] : 0
- [mov, push, pop] : 1
- [push, pop, ret] : 2

step2 3 x 3 matrix

LCS	B-0	B-1	B-2
A-0	3	2	0
A-1	2	2	0
A-2	2	0	2

Hungarian algorithm

	B-0	B-1	B-2
A-0	3	2	0
A-1	2	2	0
A-2	2	0	2

n-gram similarity
$= (3 + 2 + 2) / 3^2$
$= 7/9$

step3

4	3	1	2	3
3	4	3	2	1
2	3	3	3	2
3	2	2	3	1
4	4	2	3	4

The similarity is measured as 5/5 = 1

Method I : Using counts of indices on the relation of $y = x$

2	5	4	3	4	4	5
4	4	6	5	4	3	2
3	2	4	5	4	5	4
4	5	4	4	5	4	5
3	2	3	4	4	4	3
3	4	3	3	2	4	5
4	2	4	4	3	4	4

If m = 2, the similarity is 5 / (7-2 + 1) = 5/6.
If m = 3, the similarity is 4 / (7-3 + 1) = 4/5.

Method II : Using frequencies of consecutive indices keeping the slope 1 between indices

Discussions and Conclusions

- We conduct experiments on functions from two different versions (1.0.1f and 1.0.2h) of OpenSSL DLL files.
- F-measure (the harmonic mean of precision and recall):
  (1) not more than 0.329, when we apply only Hungarian algorithm on LCSs of two n-gram sets, but omitting Step 3 without applying method I or II
  (2) up to 0.517 when we complete the whole steps from Step 1 to Step 3, including both method I and method II.
  ➔ ***a good sign of feasibility.***
- The proposed method reduces the amount of information needed and the analysis time, and also expected to be robust even when the order of mnemonics is modified by block, because considering flow of instructions in a block (by using n-grams) as well as between blocks (by using the slop value of the indices.)
  ➔ **saving binary analysis time for malware detection.**