Supplement of relevant materials

Here are some supplementary materials regarding the review comments for VulMCI:

Regarding the issue of datasets

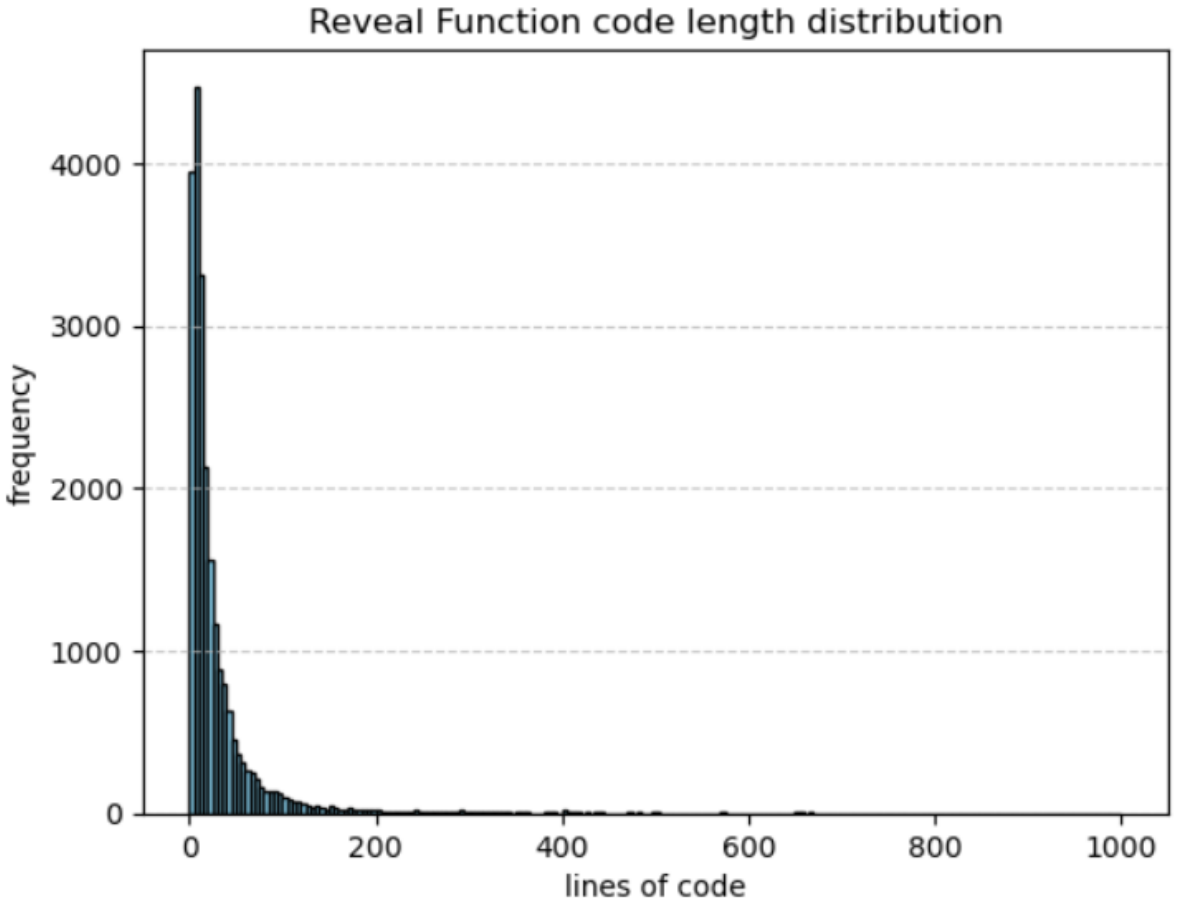
Reveal

Size represents the threshold length for the CNN input code, and dim represents the size of the vector embedding dimension, determined by the sent2vec pre-trained model. Here, 128 and 700 are compared.

> sent2vec training command:
> ./fasttext sent2vec -input processed_sentences_Reveal_normalized.txt -output your_model -minCount 8 -dim 128 -epoch 9 -lr 0.2 -wordNgrams 2 -loss ns -neg 10 -thread 20 -t 0.000005 -dropoutK 4 -minCountLabel 20 -bucket 4000000 -maxVocabSize 750000 -numCheckPoints 10

方法	remark	FPR	FNR	F1	Pr	Re	ACC
VulMCI_no_splicing	baseline	5.162	85.971	16.957	21.429	14.029	87.467
VulMCI_size=400_dim=700		2.235	58.683	50.735	65.714	41.317	92.461
VulMCI_size=200_dim=128		3.539	44.012	58.898	62.126	55.988	92.658
VulMCI_size=400_dim=128		**1.459**	**56.587**	**55.133**	**75.521**	**43.413**	**93.361**
AMPLE	SOTA	-	-	48.48	51.06	46.15	92.71

The experiment found that with dim=128, a balance was achieved between time cost and detection effectiveness. Additionally, the length of the Reveal function mainly falls within 200 lines, and with concatenated lines, it generally does not exceed 400 lines. Through experiments with thresholds of 200, 300, 400, 500, and 1000, we determined that the optimal size is 400.

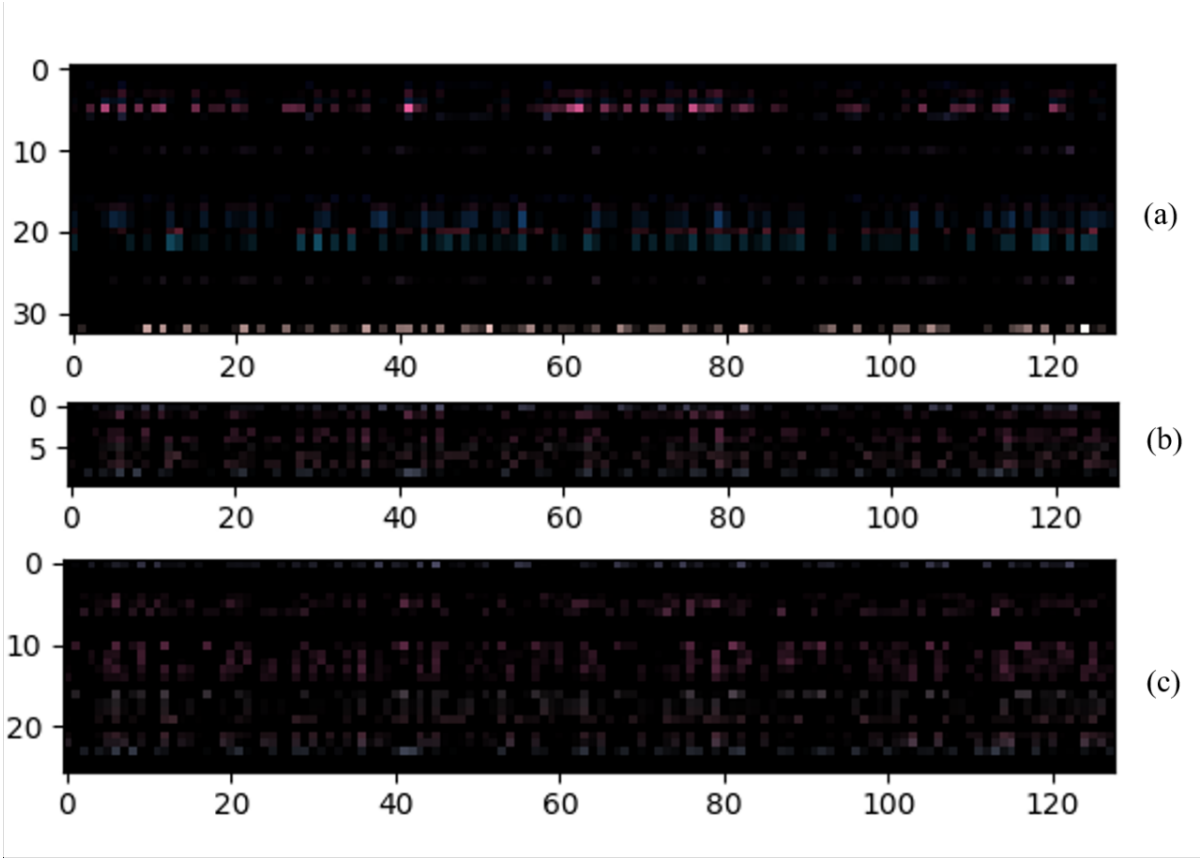


Comparison of the difference between the data before and after pixel-row oversampling

Image generation and analysis

In the figure, we demonstrate the changes in the image at three stages. First, (a) shows the generation of the vulnerability code image directly using CPG node relationships. This method produces images with high sparsity, where more than half of the black areas represent information-scarce parts, and the contrast of the color dots is not conducive to retaining vulnerability features during convolution. By converting the CPG into a CMG (code line mapping graph, mapping token nodes to code lines), we reduced the redundant information of the nodes. Without losing code semantics, this reduced the image size by more than two-thirds and increased color density, as shown in (b). Finally, to further enrich the code semantic information, we adopted a pixel row oversampling method, making the values between adjacent pixel rows more continuous. This results in an image with a certain degree of gradual change. At this point, we have

obtained a streamlined yet information-rich three-channel image (c).

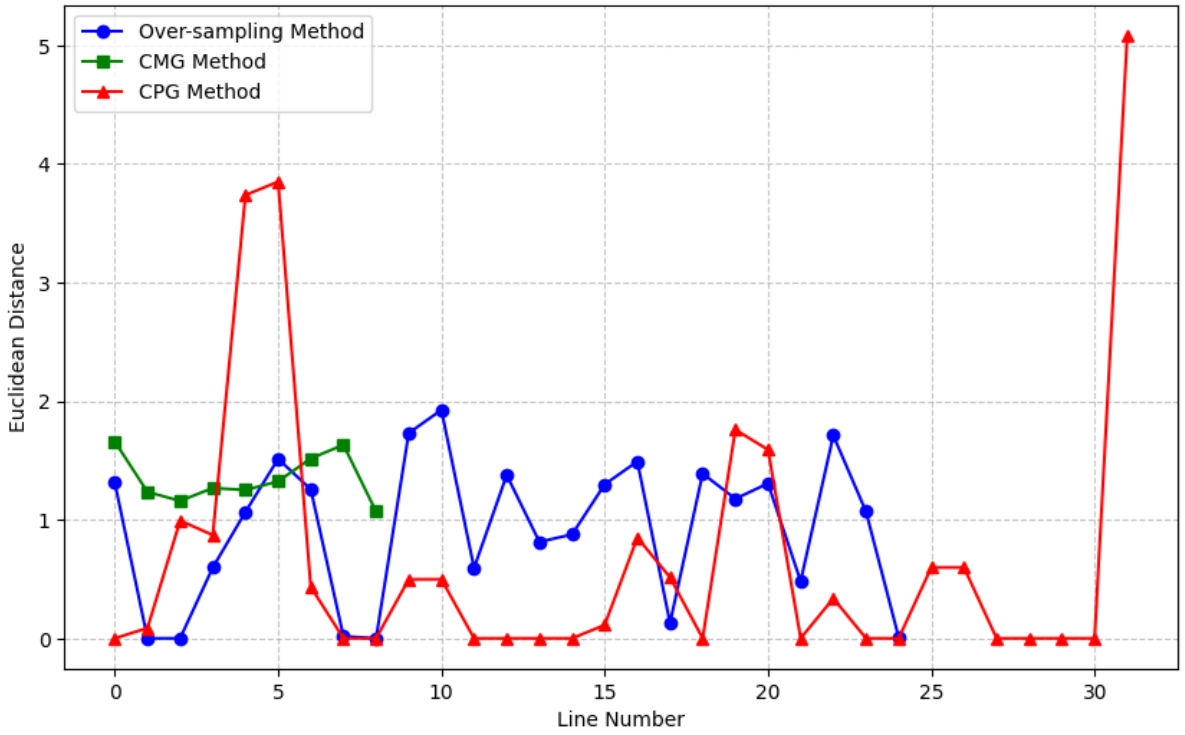


Based on the comparison of the three methods for processing vulnerability images, we further conducted data analysis to verify the effect of pixel row oversampling on enhancing code continuity. In the figure, we calculated the Euclidean distance for the three channels of the vulnerability feature images under each processing method. These three channels correspond to different centrality metrics. To visualize the distribution of Euclidean distances between row vectors under different processing methods, we selected the results from the degree centrality channel for display, as shown in Figure 7. The horizontal axis represents the row number of the row vectors, and the vertical axis represents the Euclidean distance. The smaller the distance, the closer the adjacent code rows are, indirectly proving the enhancement of code continuity.

First, the method of directly using CPG nodes (labeled "CPG Method") shows large fluctuations, with 14 row vectors having a distance of 0. This phenomenon can be attributed to the presence of discontinuous color dots and continuous zero vectors, corresponding to the characteristics of the vulnerability image mentioned earlier.

Second, the processing method using only CMG nodes (labeled "CMG Method") results in fewer row vectors due to the simplification of redundant information from CPG. The overall Euclidean distance ranges from 1 to 2, which is relatively high. To enrich code semantics and enhance vulnerability features, we introduced pixel row oversampling (labeled "Over-sampling Method"). Since sent2vec uses context for vector embedding, the changes in embedding results are reflected in the fluctuations of the Euclidean distance. Despite the fluctuations, nearly half of the distances are less than 1, demonstrating improved continuity between pixel rows.

It should be noted that in the Euclidean distance graph of pixel row oversampling, there are a few points with a distance of 0. This is because some unimportant code rows have centrality metrics of None. In the algorithm logic of pixel row oversampling, this value is retained, resulting in zero vectors when multiplied by the row vectors.


Over-sampling Method
CMG Method
CPG Method
Euclidean Distance
Line Number
0
1
2
3
4
5
0
5
10
15
20
25
30