

AUTHORS: Benjamin Steenhoek <benjis@iastate.edu>, Wei Le

TITLE: Refactoring Programs to Improve the Performance of Deep Learning for Vulnerability Detection

PROBLEM

- Limited/imbalanced vulnerability datasets hurt model performance
- Synthetic data does not represent real-world code
- Existing automatic labeling is often noisy or biased
- Manual labeling is expensive

CONTRIBUTION

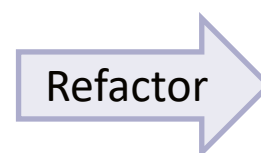
- General technique: refactoring for data augmentation
- Heuristics targeting vocabulary and data imbalance issues
- Implement and evaluate on dataset of C programs with SOTA models Devign [1] and ReVeal [2]

SOLUTION

- Refactor programs to increase diversity [3]
- Behavior is unchanged
- Apply varied sequences of refactorings to generate diverse programs
- Refactor buggy programs to rebalance dataset
- Increased test F1 by 2.12 (Devign) and 1.33 (ReVeal)

```
1 int main()
2 {
3     int x = 0;
4     int y = 3; // PermuteStmt
5     x = 10; // RenameVariable
6
7     // InsertNoop
8
9     // LoopExchange
10    for (int i = 0; i < x; i ++){
11
12        y -= i;
13        switch (y) // SwitchExchange
14        {
15            case 5:
16                x += 10;
17                break;
18            case 3:
19                x += 25;
20                break;
21        }
22    }
23    return x * y;
24 }
```

Original program



```
1 int main()
2 {
3     int y = 3;
4     int foo = 0;
5     foo = 10;
6
7     char *bar = "baz";
8
9     int i = 0;
10    while(i < foo)
11    {
12        y -= i;
13        if (y == 5)
14        {
15            foo += 10;
16        }
17        else if (y == 3)
18        {
19            foo += 25;
20        }
21        i ++;
22    }
23    return foo * y;
24 }
```

Refactored program

LIMITATIONS AND FUTURE WORK

Limitations

- Limited search space for refactoring
- Only implemented 5 operations for C

Future work

- Unsound mutations for greater diversity
- More languages or refactoring operations for more general usage

RELATED WORK

[1] Zhou, Y. et al. "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks." NIPS (2019).

[2] Chakraborty, S. et al. "Deep Learning based Vulnerability Detection: Are We There Yet?" IEEE TSE (2021).

[3] Rabin, M. R. I. et al. "Evaluation of Generalizability of Neural Program Analyzers under Semantic-Preserving Transformations." ArXiv (2019).