IOWA STATE UNIVERSITY
Department of Computer Science

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)

```c
int main()
{
    int x = 0;
    int y = 3; // PermuteStmt
    x = 10; // RenameVariable

    // InsertNoop

    // LoopExchange
    for (int i = 0; i < x; i ++)
    {
        y -= i;
        switch (y) // SwitchExchange
        {
            case 5:
            x += 10;
            break;
            case 3:
            x += 25;
            break;
        }
    }
    return x * y;
}
```

Original program

Refactor

```c
int main()
{
    int y = 3;
    int foo = 0;
    foo = 10;

    char *bar = "baz";

    int i = 0;
    while(i < foo)
    {
        y -= i;
        if (y == 5)
        {
            foo += 10;
        }
        else if (y == 3)
        {
            foo += 25;
        }
        i ++;
    }
    return foo * y;
}
```

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).

Master’s thesis. Unpublished as of date.