УДК 004.021

ПРИМЕНЕНИЕ МЕТОДА РЕКУРРЕНТНЫХ ДЕРЕВЬЕВ К АНАЛИЗУ ЦИКЛОВ В БАЙТ-КОДЕ JVM

Гречишкина Д. (Университет ИТМО)
Научный руководитель – к.т.н., доцент Корнеев Г.А.
(Университет ИТМО, факультет информационных технологий и программирования)

Статический анализ кода – важная и полезная, однако сложная задача. Особенно сложной частью является анализ циклов. В работе предлагается расширение метода рекуррентных деревьев в применении к анализу циклов в байт-коде JVM.

Введение. Анализ потока исполнения используется не только компилятором для оптимизации исполнения, но и для поиска уязвимостей в программах, нахождения ошибок в коде и генерации тестов, являясь важным методом статического анализа. Значительную сложность при анализе вызывают циклы: во-первых, в общем случае нельзя определить, завершится ли цикл или нет; во-вторых, вложенные циклы потенциально генерируют экспоненциальное число возможных путей исполнения, что существенно ограничивает множество программ, поддающихся анализу за разумное время. Например, анализатор Angr, использующийся для поиска уязвимостей в бинарном коде, прекращает анализ при получении более чем 256 активных путей исполнения. Некоторые анализаторы поступают по-другому: разворачивают первые несколько итераций цикла и анализируют только их. Если число итераций цикла не очевидно (например, для цикла while), то невозможно обнаружить ошибки, проявляющие себя на последних итерациях, потому что невозможно развернуть цикл полностью. Таким образом, поиск более эффективных подходов к анализу циклов является актуальной задачей.

Основная часть. Предлагаемый метод анализа основывается на предположении, что существует некоторое достаточно широкое множество циклов, которые можно преобразовать в канонический вид методом рекуррентных цепочек. Цикл имеет канонический вид, если существует некоторая переменная цикла, имеющая в начале цикла значение ноль, которая с каждой итерацией увеличивается на единицу, а все остальные переменные из тела цикла зависят только от нее. Таким образом цикл может быть инстанциирован на любой итерации, поскольку переменные в его теле зависят только от номера итерации, но не от предыдущих итераций.

Построение рекуррентной цепочки для переменной u алгоритмом, предложенным Себастьяном Попом, Филиппом Клауссом и другими, представляет собой обход в глубину по графу потока исполнения, представленному в SSA форме, начиная с первой встречи объявления u, до повторной встречи этого объявления. В процессе поиска запоминается и накапливается (в виде рекуррентного дерева) информация о том, как переменная u изменяется в течение цикла.

Собственно, решаемая задача – символьное вычисление такого дерева, решение рекуррентной зависимости. В библиотеке SymKt были реализованы методы, символьно вычисляющие рекуррентные деревья, если это возможно и осмысленно. Далее, с помощью этой библиотеки и библиотеки kfg, реализован алгоритм, который для данного цикла производит следующие действия: строит рекуррентные деревья для каждой переменной из тела цикла и символьно вычисляет эти деревья. Если в цикле и всех его вложенных циклах все деревья, соответствующие переменным, вычислены (то есть для каждой переменной известна нерекуррентная формула), то алгоритм перестраивает цикл следующим образом: в граф потока исполнения добавляется новая переменная, соответствующая номеру итерации и

имеющая неопределенное значение. Для каждой переменной из тела цикла по соответствующей ей формуле генерируются инструкции. Далее Kex по измененному дереву строит формулы для SMT-решателя и генерирует тесты для класса.

Выводы. Предложенное расширение метода рекуррентных деревьев позволит увеличить покрытие кода тестами.

Гречишкина Д. (автор) Подпись

Корнеев Г.А. (научный руководитель) Подпись