А.С. Юмаганов

МЕТОД ФОРМИРОВАНИЯ БАЗИСНОЙ БИБЛИОТЕКИ ФУНКЦИЙ В ЗАДАЧЕ ПОИСКА ПОХОЖИХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ КОДА

(Самарский университет)

Введение

Решение задачи поиска похожих последовательностей кода в исполняемых файлах может быть использовано для поиска известных уязвимостей в программном обеспечении (ПО) [1], поиска плагиата кода [2], детектирования вредоносных программ [3].

Настоящая работа является продолжением прежних исследований автора [4], посвященных разработке метода решения задачи поиска похожих последовательностей кода в исполняемых файлах. В данной работе рассматривается способ формирования базисной библиотеки, используемой данным методом.

Описанный в предыдущей работе автора [4] метод поиска похожих функций использует следующие базовые понятия:

- текущая библиотека – набор функций исследуемого исполняемого файла;
- архивные данные – набор известных функций и их описаний через библиотеку базисных функций;
- базисная библиотека – вспомогательный набор функций, применяемый для сравнения функций архивных данных и текущей библиотеки.

Задача, решаемая этим методом, формулируется следующим образом: для заданной функции текущей библиотеки найти наиболее релевантную функцию из архивных данных. Описание функций в данном методе формируется на основе пространственного положения команд процессора каждой функциональной группы [5] в теле функции.

Данный метод поиска похожих функций состоит из трех этапов. На первом этапе происходит представление функций архивных данных через библиотеку базисных функций и сохранение полученных описаний в архивной базе данных (БД). На втором этапе аналогичным образом формируется БД текущей библиотеки. На заключительном третьем этапе осуществляется непосредственно поиск похожих функций, в результате которого для заданной функции текущей библиотеки формируется упорядоченный по критерию евклидовой близости список функций архивных данных.

Базисная библиотека функций, используемая в данном методе, содержит 50 функций, выбранных вручную из различных исполняемых файлов. В данной работе представлен альтернативный способ формирования базисной библиотеки, позволяющий добиться улучшения качественных показателей поиска. В основе данного способа лежит использование не реальных функций, а искусственно сгенерированных функций.

Генерация функций базисной библиотеки

Для генерации функций базисной библиотеки используется генетический алгоритм [6], который применяется для решения различных оптимизационных задач. Для описания работы данного алгоритма используется терминология биологической науки – генетики. Особь представляет собой потенциальное решение оптимизационной задачи, она кодируется *хромосомой*. Множество особей составляет *популяцию*. Работа алгоритма сводится к поиску оптимального решения задачи в процессе эволюции популяций, осуществляемой с помощью некоторых генетических операций. Данный алгоритм является эвристическим и состоит из четырех основных этапов: инициализация, селекция, скрещивание, мутация. Рассмотрим каждый из этих этапов подробнее.

На первом этапе работы алгоритма создается исходная популяция. Для этого необходимо описать структуру хромосомы и случайным образом заполнить ее содержимое для каждой особи исходной популяции. Для генерации функции базисной библиотеки будем использовать хромосому, имеющую следующую структуру. Пусть K – размер функции, M – число функции в базисной библиотеке. Тогда вектор размерности $K * M$, каждый элемент которого представляет собой номер группы команд процессора, будем использовать в качестве хромосомы, описывающей набор из M функций базисной библиотеки. В данной работе $K = 100$, $M = 24$.

На этапе селекции для каждой хромосомы из текущей популяции вычисляется значение целевой функции (fitness function), которое определяет приспособленность особи. Чем выше значение целевой функции, тем с большей вероятностью данная особь получит потомство. В данной работе в качестве целевой функции используется усредненное значение средней точности поиска [4] при использовании в качестве текущей библиотеки двух различных версий библиотеки librtiff: 4.0.8 и 3.9.7. В качестве архивных данных использовались функции библиотеки libtiff 4.0.3, а в качестве функций базисной библиотеки – соответственно поочередно каждая из хромосом.

Процесс селекции заключается в выборе наиболее приспособленных особей, остальные особи «погибают» Для отбора приспособленных особей используется метод рулетки (roulette wheel method), при котором вероятность выбора особи тем выше, чем больше значение ее целевой функции. Вероятность выбора особи рассчитывается по формуле:

$$p_i = \frac{F_i}{\sum_{n=1}^{N} F_n},$$

где p_i – вероятность выбора особи i, F_i – значение целевой функции для особи i, N – количество особей в популяции.

На следующем этапе осуществляется скрещивание особей. Для каждой «выжившей» на этапе селекции особи А случайным образом подбирается пара

В. Случайным образом выбирается точка скрещивания, диапазон значений которой ограничен отрезком [1, …, $K*M-1$], где $K*M$ – длина хромосомы. Эта точка разбивает хромосомы родителей на две части. Затем формируются две новых хромосомы путем обмена частей хромосом родителей. Из двух полученных хромосом случайным образом выбирается одна хромосома, и полученный потомок заменяет своего родителя А в текущей популяции. Скрещивания особи осуществляется с заданной вероятностью $p_{crossover}$.

На четвертом этапе генетического алгоритма к особям применяется следующая генетическая операция – мутация. Мутация так же осуществляется с заданной вероятностью $p_{mutation}$. Случайным образом выбирается некоторая позиция в хромосоме особи и соответствующий данной позиции элемент хромосомы меняется на случайно выбранное значение номера группы команд процессора, отличного от исходного значения.

Этапы селекции, скрещивания и мутации выполняются в цикле. Условием выхода из цикла является достижение заданного числа поколений. Так как при выполнении описанных выше генетических операций возможно уменьшение значения целевой функции у потомков, хромосома особи, обладающая наибольшим значением целевой функции в поколении, сохраняется. Результатом работы алгоритма будет хромосома, значение целевой функции которой является максимальным среди лучших хромосом каждого поколения.

В данной работе использовались следующие параметры генетического алгоритма: вероятность скрещивания $p_{crossover}=0.75$, вероятность мутации $p_{mutation}=0.05$, число особей в поколении $N=10$, число поколений $Gen_{num}=75$.

Экспериментальные исследования

В ходе проведения экспериментальных исследований при описанных выше параметрах генетического алгоритма лучшее значение целевой функции было получено в 50 поколении – $F_{50}=0.8111$. Значение целевой функции для базисной библиотеки, составленной из отобранных вручную «реальных» функций различных исполняемых файлов, вычисленной на тех же данных оказалось равным $F_r=0.8031$.

Сравнение средней точности поиска похожих функций при использовании различных базисных библиотек было произведено на других исходных данных: архивные данные были представлены функциями библиотеки curl версии 7.6.3, а функции текущей библиотеки – функциями различных других версий библиотеки curl. Результаты экспериментальных исследований представлены в таблице 1.

Анализ полученных результатов показывает, что представленный в данной работе метод генерации функций базисной библиотеки позволяет повысить показатель средней точности поиска представленного в работе [4] метода поиска похожих последовательностей кода.

Таблица 1. Сравнение способов генерации базисной библиотеки

Текущая библиотека	Средняя точность поиска, используя оригинальный способ формирования базисной библиотеки	Средняя точность поиска, используя базисную библиотеку, сформированную с помощью представленного в данной работе метода
libcurl 7.5.4	0.7828	**0.7886**
libcurl 7.5.6	0.8550	**0.8626**
libcurl 7.5.9	0.8851	**0.8929**
libcurl 7.6.0	0.9036	**0.9089**

Заключение

В работе представлен метод генерации функций базисной библиотеки, которая используется для формирования векторного описания функций исполняемых файлов в представленной ранее работе. Представлены результаты экспериментальных исследований, демонстрирующие преимущество представленного метода формирования базисной библиотеки над способом ее формирования, используемым в оригинальной работе.

Литература

[1] David, Y. Tracelet-based code search in executables / Y. David, E. Yahav // Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation. - 2013. - DOI:10.1145/2594291.2594343.

[2] Hemel, A. Dolstra Finding software license violations through binary code clone detection / A. Hemel, K.T. Kalleberg, R.E. Vermaas // Proceedings of the 8th International Working Conference on Mining Software Repositories. – 2011. – P. 63-72. – DOI: 10.1145/1985441.1985453.

[3] Kruegel, C. Polymorphic worm detection using structural information of executables / C. Kruegel, E. Kirda // Proceedings of the 8th International Conference on Recent Advances in Intrusion Detection. – 2005. – P. 207-226. – DOI: 10.1007/11663812_11.

[4] Юмаганов, А.С. Метод поиска похожих последовательностей кода в исполняемых бинарных файлах с использованием беспризнакового подхода / А.С. Юмаганов, В.В. Мясников // Компьютерная оптика. – 2017. – Т. 41, № 5. – С. 756-764. – DOI: 10.18287/2412-6179-2017-41-5-756-764.

[5] x86 Assembly language reference manual [Электронный ресурс]. – 2010. – Режим доступа: https://docs.oracle.com/cd/E19253-01/817-5477/817-5477.pdf

[6] Holland, J.H. Adaptation in natural and artificial systems / J.H. Holland // University of Michigan Press, Ann Arbor. – 1975. – 96 p.