# Поиск клонов фрагментов в исполняемом коде

***Сирунян Вааган Телемакович***

*Студент*

*Филиал МГУ имени М. В. Ломоносова в г. Ереване, Ереван, Армения*

*E-mail:* sirunyan@ispras.ru

***Научный руководитель*** *— Вылиток Алексей Александрович*

Разработчики программного обеспечения часто копируют и вставляют фрагмент кода (последовательность строк кода) в проект. Исследования показали, что до 20 процентов кода могут являться похожими фрагментами (клонами) [1]. Данная работа посвящена поиску клонов фрагментов исполняемого кода. Обнаружение клонов фрагмента исполняемого кода может использовтся для анализа вредоносного программного обеспечения, обнаружения ошибок, выявления нарушений авторских прав и т. д.

Существует ряд методов и инструментов поиска клонов исполняемого кода — основанные на сравнении хешей кода [2], метрические [3,4] и семантические [5,6] методы. Методы описанные в работах [2-4] не требуют большой вычислительной мощности и работают сравнительно быстро. Однако они не учитывают поток управления и поток данных (частично используется только в [3]). Этот пробел восполняется в работе [5], где используется динамическое символьное выполнение. Однако, такой метод немасштабируемый и непригодный для анализа нескольких мегабайтов исполняемых файлов (ИФ).

Целью данной работы является создание метода поиска клонов фрагментов кода, который учитывает поток управления, поток данных и также реализовать инструмент, который может масштабироваться для анализа нескольких десятков мегабайт ИФ за несколько часов.

На вход метод принимает фрагмент исполняемого кода (адреса начала и конца) в первом ИФ и второй ИФ для поиска в нем клонов фрагмента. Предложенный метод состоит из следующих шагов: 1) дизассемблирование обоих ИФ с помощью дизассемблера IDA Pro [7]; 2) приведение ассемблера в промежуточное представление REIL, предоставляемое средой BinNavi [8]; 3) построение графа зависимостей программы фрагмента (ГЗП) и ГЗП [9] для всех функций второго ИФ, вершинам ГЗП соответствуют инструкции REIL, а ребрам – зависимости по данным и по управлению; 4) поиск изоморфных подграфов ГЗП фрагмента во всех ГЗП второго ИФ, используя раз-

работанный эвристический алгоритм; 5) фрагменты исполняемого кода, соответствующие полученным подграфам, выводятся как клоны фрагмента. Описанный метод основан на методе из [10], предназначенном для обнаружения клонов на уровне функций.

Алгоритм хорошо зарекомендовал себя для поиска клонов фрагментов: средний результат на тестовых ИФ — обнаружение 95% фрагмента. Дальнейшей работой является использование разработанного метода для поиска семантических ошибок и для поиска вредоносного кода.

**Литература**


1. Roy C.K., Cordy J.R. An empirical study of function clones in open source software systems // Proceedings of the 15th Working Conference on Reverse Engineering, 2008, P. 81–90.
2. Jang J., Brumley D. Bitshred: Fast, scalable code reuse detection in binary code // CyLab, 2009, P. 28.
3. Bruschi D., Martignoni L., Monga M. Code normalization for self-mutating malware // IEEE Security & Privacy, 2007, P. 46–54.
4. Sæbjørnsen A., Willcock J., Panas T., Quinlan D., Su Z. Detecting code clones in binary executables // Proceedings of the 18th International Symposium on Software Testing and Analysis, 2009, P. 117–128.
5. Ramtine Tofighi-Shirazi, Maria Christofi, Philippe Elbaz-Vincent, Thanh-Ha Le. DoSE: Deobfuscation based on Semantic Equivalence // SSPREW-8, San Juan, United States, 2018.
6. Liu B., Huo W., Zhang Ch., Li W., Li F., Piao A., Zou W. $\alpha$Diff: cross-version binary code similarity detection with DNN // Proceedings of the 2018 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18), 2018, Montpellier, France. ACM, New York, NY, USA, 12 pages.
7. IDA Pro: `https://www.hex-rays.com/products/ida`
8. BinNavi: `https://www.zynamics.com/binnavi.html`
9. Ferrante J., Ottenstein K.J., Warren J.D. The Program Dependence Graph and Its Use in Optimization // ACM Transactions on Prgramming Languages and Systems, 1987, Vol. 9 № 3.
10. Aslanyan H.K. Effective and Accurate Binary Clone Detection // Mathematical Problems of Computer Science, 2017. T. 48. P. 64–73.