Метод извлечения алгоритма программы из трассы машинных инструкций

Кошкин Вячеслав Александрович

Старший лаборант

Институт системного программирования им. В. П. Иванникова РАН, Москва, Россия

E-mail: trickyfox371@ispras.ru

Научный руководитель *— Падарян Вартан Андроникович*

Развитие технологий программирования приводит к усложнению программного обеспечения (ПО) и увеличению его размеров. Анализ безопасности кода современного ПО с целью обнаружения потенциально эксплуатируемых ошибок и недекларированных возможностей подразумевает глубокое изучение реализуемых в нем алгоритмов. Для изучение свойств алгоритмов требуется наличие развитого инструментария извлечения алгоритма из бинарного кода, упрощения и сокращения объема анализируемого кода. Отсутствие подобного инструментария негативно сказывается на результатах анализа безопасности кода, либо приводит к невозможности его выполнения. Таким образом, извлечение алгоритма из бинарного кода программы является неотъемлемым этапом реверс инжиниринга ПО, а задачи, связанные с развитием соответствующих инструментов, приобретают все большую актуальность [1–2].

В контексте данной работы под извлечением алгоритма программы из бинарного кода подразумевается выделение среди всех инструкций подмножества, принадлежащего анализируемому алгоритму, а также последующее построение высокоуровневого представления алгоритма. В качестве высокоуровневого представления алгоритма может выступать, например, код на высокоуровневом языке, полученный при декомпиляции, либо граф потока управления. Такие представления алгоритма получаются путем статического анализа кода. Их основными недостатком является отсутствие информации времени выполнения, и, как следствие, полученный алгоритм является неточным. Изучение его свойств приведет к некорректным результатам анализа безопасности ПО.

В работе предложен метод извлечения алгоритма программы из бинарного кода, основанный на динамическом анализе кода в автономном режиме. В основе метода лежит алгоритм отслеживания потока данных в прямом и обратном направлениях. Построение высокоуровневого представления алгоритма выполняется итеративным

двухэтапным процессом. На каждом этапе строится свое представление алгоритма: на первом этапе — функциональная схема слайса [3], а на втором — схема выполнения алгоритма.

Функциональная схема слайса является иерархической блок–схемой, в которой инструкции, принадлежащие анализируемому алгоритму, объединяются в более крупные функциональные элементы. Схема выполнения алгоритма является более высокоуровневой блок–схемой, в которой основными функциональными элементами являются модели функций — высокоуровневые описания на уровне имени и параметров. Схема выполнения алгоритма имеет более простую структуру, лишенную иерархичности, и меньший объем, чем функциональная схема слайса, поскольку включает в себя только модели функций и их параметры. Разработанные способы представления извлеченных алгоритмов могут быть использованы как для ручного, так и для автоматического анализа кода.

Предложенный метод построения высокоуровневого представления алгоритма на каждой итерации включает следующие шаги: 1) построение функциональной схемы слайса; 2) анализ функциональной схемы слайса и создание моделей функций; 3) построение схемы выполнения алгоритма. В случае, если после выполнения третьего шага была получена блок–схема удовлетворительного качества и нужного уровня детализации, итеративный процесс извлечения алгоритма прекращается, в противном случае выполняется дополнительная итерация с учетом новых созданных моделей функций.

В докладе будут раскрыты некоторые технические детали реализации каждого шага итеративного процесса построения блок–схемы, открытые вопросы и актуальные направления для дальнейших исследований, а также продемонстрированы результаты работы предложенного метода.

Литература

1. Caselden D., Bazhanyuk A., Payer M., McCamant S., Song D. HI–CFG: Construction by Binary Analysis and Application to Attack Polymorphism // In Proceedings of the European Symposium on Research in Computer Security, Egham, UK, 2013, P. 164–181.
2. Vaughn M., Reps T. A Generating-Extension-Generator for Machine Code // arXiv preprint arXiv:2005.06645. 2020.
3. Bugerya A. B., Kulagin I. I., Padaryan V. A., Solovev M. A., Tikhonov A. Yu. Recovery of High–Level Intermediate Representations of Algorithms from Binary Code // In Proceedings of Ivannikov Memorial Workshop (IVMEM), Veliky Novgorod, 2019, P. 57–63.