С.С. Гайсарян, В.А. Падарян, А.Ю. Тихонов

КОМБИНИРОВАННЫЙ АНАЛИЗ БИНАРНОГО КОДА В ЗАДАЧАХ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ ⋆

г. Москва, Институт системного программирования РАН.

На протяжении ряда лет в ИСП РАН ведется разработка интегрированной среды анализа бинарного кода [1]. Подход к анализу, который она автоматизирует, предполагает согласованное применение динамического и статического анализа.

Изначальные требования ориентировали подход на решение задачи извлечения отдельных алгоритмов из кода программ и высокоуровневого представления этих извлеченных алгоритмов, что является типовым действием при анализе программных реализаций. Впоследствии было поставлено дополнительное требование: задача должна решаться в тех случаях, когда применение обычных средств анализа затруднено или невозможно. Это требование обусловило применение динамического post-mortem анализа трасс выполнения, полученных из полносистемного симулятора, позволяющих корректно анализировать потоки данных. На основе этого анализа выполнялось выделение кода.

Развитие подхода привело к выводу, что качественное решение задачи выделения алгоритма невозможно в рамках одного только статического или динамического подхода. Принципиальным недостатком динамического анализа является то, что «виден» только тот код, который выполнялся и попал в трассу. Незадействованные фрагменты кода выпадают из рассмотрения. Статический подход не гарантирует восстановление кода (из-за запаковки, изменения с течением времени и т.п.) и его структуры (из-за косвенной адресации). Более того, для понимания работы кода зачастую требуется знать состояние «типового» контекста, значений регистров и памяти. Отсутствие контекста вынуждает применять консервативный анализ, рассматривая все потенциально допустимые ситуации, что негативно сказывается на точности получаемых результатов.

Центральной идеей комбинированного анализа является построение по трассам статического представления, способного отражать некоторые динамические свойства. Для каждого адресного пространства виртуальной памяти строится граф: расширенный типизированными ребрами межпроцедурный граф потока управления. Вершинами являются базовые блоки, состоящие из последовательности машинных команд, ребра – четырех типов: вызов функции, возврат из функции, передача управления внутри тела функции, переход на новое поколение кода.

Важное свойство предложенного представления – способность адекватно отражать изменения кода, возникающие с течением времени. Причем изменения могут быть обусловлены произвольными причинами: загрузка динамической библиотеки, jit-компиляция, и, в том числе, распространение в памяти системы кода вредоносного ПО. Другое полезное свойство представления заключается в возможности его дополнения блоками и ребрами, восстановленными из других трасс, что позволяет отчасти компенсировать основной недостаток динамического подхода: ограниченное покрытие кода.

Предложенный подход к дополнению статического представления является более общим, чем дополнение на основе статического дизассемблирования незадействованных переходов [2], поскольку позволяет включать код, задействованный

⋆ Работа поддержана грантом РФФИ 11-07-00450-а

с помощью косвенной адресации, размещенный в обработчиках прерываний и исключений.

Построенное на основе трасс выполнения статическое представление позволяет применять адаптированный для машинного кода программный слайсинг [3], эффективно выделяя код отдельных алгоритмов на основе восстановленных зависимостей по данным и управлению.

Другим практическим применением анализа бинарного кода является поиск ошибок в широком понимании этого термина. Нарушение соглашений уровня модели вычислителя или языка программирования приводит к дефектам реализации: разыменованию нулевого указателя, делению на ноль, переполнению буфера, ошибкам работы с динамической памятью. Нарушение соглашений более высокого уровня приводит к таким ошибкам, как блокировки потоков выполнения, нарушение конфиденциальности обрабатываемой информации и т.п.

Значительный интерес представляют две конкретные задачи: автоматическое построение эксплоитов для бинарного кода и поиск нарушений конфиденциальности. Перспективные методы решения первой неоднократно рассматривались в открытых публикациях. Как правило, они основаны на механизме смешанного символьно-конкретного выполнения. Для решения второй задачи известные публикации предлагают только анализ помеченных данных, причем точность этого анализа неудовлетворительна, если не учитывать обратимость преобразований, выполняемых над данными.

В рамках среды анализа доступна инфраструктура, поддерживающая решение этих задач. Разработано внутреннее представление низкого уровня Pivot и средства трансляции машинного кода различных целевых архитектур в это представление [4]. Код Pivot используется для построения системы логических уравнений решателя Z3. Поддерживается построение систем уравнений с целью нахождения условий срабатывания незадействованных условных переходов. В состоянии исследования находится вопрос построения систем уравнений, описывающих переполнение буферов памяти, применительно как к динамическому, так и статическому анализу. В последнем случае используется восстановленное по трассам статическое представление программы.

Литература

1. В.А. Падарян, А.И. Гетьман, М.А. Соловьев. Программная среда для динамического анализа бинарного кода. // Труды Института Системного Программирования, том 16, 2009, стр. 51-72
2. Domagoj Babić, Lorenzo Martignoni, Stephen McCamant, and Dawn Song. Statically-directed dynamic automated test generation. // In Proc. of the 2011 Int. Symposium on Software Testing and Analysis (ISSTA '11). ACM, New York, NY, USA, pp. 12-22.
3. Андрей Тихонов, Вартан Падарян. Применение программного слайсинга для анализа бинарного кода, представленного трассами выполнения. // Материалы XVIII Общероссийской научно-технической конференции «Методы и технические средства обеспечения безопасности информации», 2009, стр. 131.
4. В.А. Падарян, М.А. Соловьев, А.И. Кононов. Моделирование операционной семантики машинных инструкций. // Программирование, №3, 2011, стр. 50-64.