Нанесение водяных знаков на программное обеспечение

Архипов Иван Сергеевич

Санкт-Петербургский Государсвенный Университет

группа 21.M04-мм
Научный руководитель: д.ф-м.н., профессор А.Н.Терехов
Консультант: старший преподаватель Уральского федерального университета, А.Е.Сибиряков

Обзор водяных знаков

- Статические водяные знаки внедрены в код и/или данные компьютерной программы
- Динамические водяные знаки хранят информацию в состояниях выполнения программы

Характеристики водяных знаков

- Надёжность
- Требуемый объём ресурсов
- Невидимость
- Защита частей кода, а не всего проекта целиком
- Устойчивость

Новая архитектура

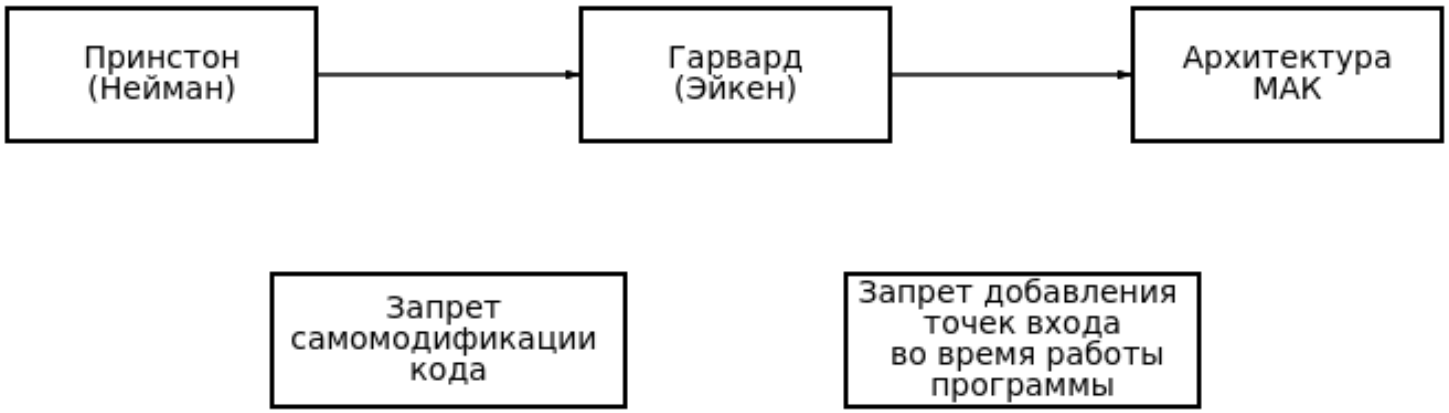

Рис.: Архитектурные концепции

Новые возможности

- В Гарвардской архитектуре нельзя определить точки входа программы. Из-за этого нельзя “раздвинуть” линейные участки кода и перемешать их, так как мы не можем заранее сказать, придёт ли управление в данную точку
- В новой архитектуре этого недостатка нет, что открывает широкие возможности, например, для запутывания кода. В том числе в данной архитектуре открываются новые возможности для нанесения водяных знаков на программное обеспечение

Водяные знаки в новой архитектуре

- Можно “раздвинуть” линейные участки кода, а в них записать водяной знак. Способ нанесения водяного знака зависит от его цели. Например, если нужен видимый водяной знак, то можно просто его туда записать, если нужен невидимый, то нужен способ его спрятать
- Хочется иметь способ наносить целый комплекс водяных знаков с разными характеристиками

Каркасный водяной знак

- Необходим способ понимать, что в данной области записан водяной знак. Этот способ и есть каркасный водяной знак
- На данный момент предлагается использовать редкие битовые последовательности. Но такие последовательности легко найти, потому хочется иметь способ их спрятать

Вероятностные характеристики ПО

- Оказалось, что программный код имеет вполне определённые вероятностные характеристики. Например, группой М.В.Баклановского было выяснено, что нулей в коде примерно 60%, а единиц примерно 40%
- Чтобы спрятать водяной знак, нужно “подогнать” эмпирические статистики такие, как у обычного кода. Для этого нужно эти статистики найти
- Также для сокрытия водяного знака можно использовать такие “редкие” последовательности, которые не всегда встречаются в исполняемом файле. Для этого нужно провести анализ частоты нахождения различных битовых последовательностей в коде

Предшествующая работа

Данная тема рассматривалась в курсовой работе Смирнова Дениса Павловича в 2018 году.

- Был сделан хороший обзор водяных знаков
- Вычислены некоторые статистики, однако недостаточное количество
- Отсутствие прототипа внесения и извлечения водяных знаков

Цель и задачи

Целью работы является разработка, на основе концепции MAK, прототипа фреймворка для работы с водяными знаками в программном обеспечении. Для достижения обозначенной цели были поставлены следующие задачи:

- обзор подходов к внесению водяных знаков в программное обеспечение, изучение концепции MAK (включая ограничения концепции), а также предшествующей работы по этому проекту;
- создание архитектуры фреймворка;
- прототипная реализация фреймворка;
- проведение экспериментального исследования с реализованным прототипом (выбор тестового приложения, анализ полученных результатов).

Архитектура фреймворка[1]

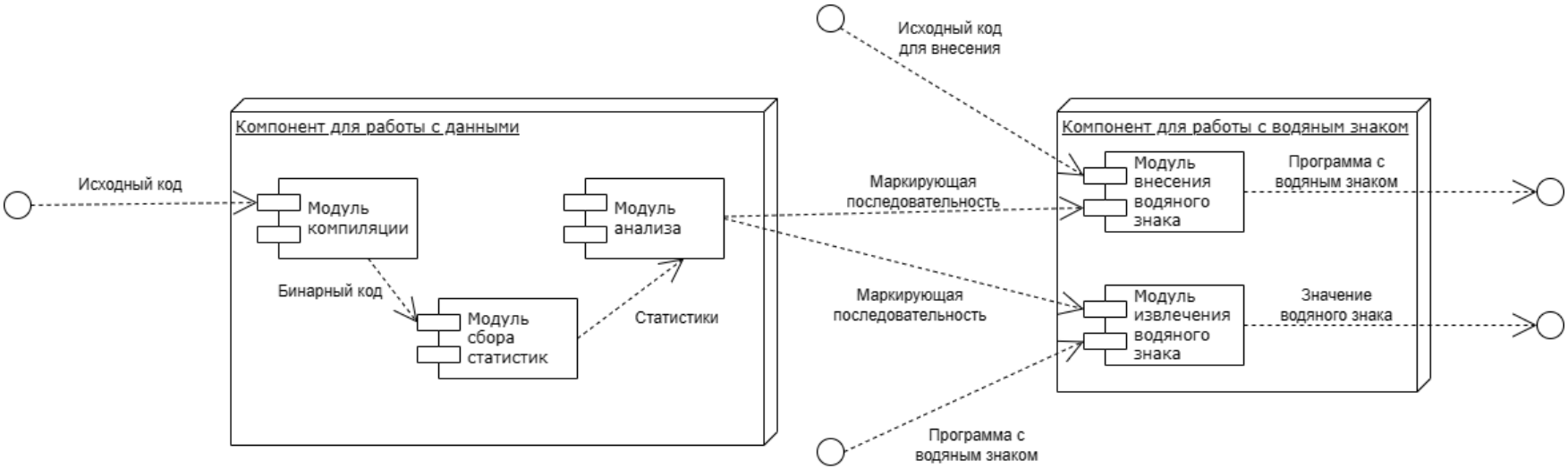

Рис.: Архитектура фреймворка

[1]Компонент для работы с данными расположен в репозитории: `https://github.com/IvanArkhipov1999/Binary-code-statistics-research`
Компонент для работы с водяным знаком расположен в репозитории: `https://github.com/IvanArkhipov1999/watermark-prototype`

Модуль компиляции

Модуль компиляции принимает на вход путь до исходных файлов, компилятор и флаги, и преобразует исходный код в бинарный. Модуль реализован на языке Python.

Модуль сбора статистик

Модуль сбора статистик для каждого бинарного файла извлекает секцию кода и считает доли вхождения бинарных последовательностей в этот файл. На выходе предоставляется csv-файл со всеми долями бинарных последовательностей.
Модуль реализован на языке Python.

Модуль анализа

Модуль анализа на вход получает csv-файл с долями последовательностей.
Модуль делает следующее:

- считает среднее, стандартное отклонение, медиану, минимальное и максимальное значения для каждой доли каждой последовательности;
- проверяет принадлежность распределения долей последовательности нормальному распределению по критерию Колмагорова-Смирнова;
- ищет "редкие" последовательности, подходящие для маркировки водяного знака.

Модуль реализован на языке R.

Модуль внесения водяного знака

Модуль внесения водяного знака получает на вход исходный файл с кодом, значение водяного знака и компилятор. Алгоритм внесения можно разбить на два этапа:

1. Выделение памяти под водяного знак. Для этого в ассемблерное представление программы вносятся необходимое количество мусорных команд, память для которых будет использована для водяного знака.
2. Внесение водяного знака. Вместо мусорных команд в бинарном представлении вносится маркирующая последовательность, длина сообщения и значение водяного знака.

На данный момент модуль поддерживает язык C++ и архитектуру x86_64, но возможно расширение.
Модуль реализован на языке Python.

Модуль извлечения водяного знака

Модуль извлечения водяного знака получает на вход бинарный файл, по определённой маркирующей каркасный водяной знак последовательности находит вложенное сообщение и выводит его.
На данный момент модуль поддерживает язык C++ и архитектуру x86_64, но возможно расширение.
Модуль реализован на языке Python.

Выбор данных для экспериментов[2]

- В качестве данных для анализа была выбрана тестовая база GCC.
- Для компиляции был выбран один из наиболее популярных на данный момент компиляторов gcc.
- Для компиляции был выбран формат elf.
- Компиляция производилась для архитектур x86_64, arm64 и mips64el.
- Компиляция была произведена с флагом -std=gnu++11.
- Рассматривались все последовательности длиной до 10 включительно.
- Результаты работы не гарантируют, что полученные в ходе анализа статистики можно будет использовать для других случаев. Однако работа показывает, что скорее всего при выборе другого языка, компилятора, платформы и других характеристик метод будет работать.

[2] Весь анализируемый датасет лежит в отдельном репозитории `https://github.com/IvanArkhipov1999/Binaries-dataset`

Архитектура x86_64[3]

- Подтвердилось утверждение о том, что в среднем в бинарном коде нулей 60%, а единиц 40%.
- Оказалось, что брать в качестве маркирующих последовательности исключительно из единиц – не самая удачная идея.
- Было найдено 13 подпоследовательностей длиной 8, 9 и 10 бит, распределение долей которых соответствует нормальному.
- Было найдено 508 редких подпоследовательностей длиной 8, 9 и 10 бит. Процент частоты нахождения в коде таких подпоследовательностей самый разный, что позволяет регулировать невидимость водяного знака.

[3]Полный перечень полученных данных лежит в репозитории `https://github.com/IvanArkhipov1999/Binaries-dataset`

Архитектура arm64[4]

- По сравнению с архитектурой x86_64 в коде arm64 содержится немного больше нулей. Подтверждается соотношение количества нулей к единицам примерно как к 3:2.
- Оказалось, что брать в качестве маркирующих последовательности исключительно из единиц – не самая удачная идея.
- Всего было найдено 45 подпоследовательностей длиной 5, 6, 7, 8, 9 и 10 бит, распределение долей которых соответствует нормальному, что существенно больше по сравнению с архитектурой x86_64 и mips64el.
- Было найдено 517 редких подпоследовательностей длиной 8, 9 и 10 бит. Процент частоты нахождения в коде таких подпоследовательностей самый разный, что позволяет регулировать невидимость водяного знака.

[4]Полный перечень полученных данных лежит в репозитории https://github.com/IvanArkhipov1999/Binaries-dataset

Архитектура mips64el[5]

- По сравнению с рассмотренными выше архитектурами в mips64el содержится значительно больше нулей. Нулей примерно 70%, а единиц 30%.
- Оказалось, что брать в качестве маркирующих последовательности исключительно из единиц – не самая удачная идея.
- Всего было найдено 7 подпоследовательностей длиной 8, 9 и 10 бит, распределение долей которых соответствует нормальному.
- Всего найдено 406 "редких" подпоследовательностей, что значительно меньше, чем в случае x86_64 и arm64. Процент частоты нахождения в коде таких подпоследовательностей самый разный, что позволяет регулировать невидимость водяного знака.

[5]Полный перечень полученных данных лежит в репозитории `https://github.com/IvanArkhipov1999/Binaries-dataset`

Результаты

- Выполнено сравнение статических и динамических водяных знаков. Изучена концепция MAK - её возможностей, достоинств и недостатков. Выполнено описание водяных знаков в MAK, приведены примеры и ограничения их применимости. Изучены результаты предыдущей работы по проекту.
- В рамках создания архитектуры фреймворка выделена компонента для работы с данными (модуль компиляции, модуль сбора статистик и модуль анализа) и компонента для работы с водяным знаком (модуль внесения водяного знака и модуль извлечения водяного знака). Фреймворк поддерживает C++ и архитектуру x86_64, имеются возможности расширения на другие языки и архитектуры процессоров.

Результаты

- Создан прототипный вариант разработанной архитектуры. Компонент работы с данными был создан на языках Python (модули компиляции и сбора статистик) и R (модуль анализа) и поддерживает любые языки и архитектуры процессора. Компонента работы с водяным знаком реализован на Python.
- Для апробации разработанного фреймворка был выбран набор тестов для g++. Было подготовлено по 3884 бинарных файлов для архитектур x86_64, arm64 и mips64el и найдены необходимые данные. Для тестирования компонента работы с водяным знаком был реализован CI с несколькими тестовыми примерами.

Результаты

- Результаты работы были представлены на всероссийской конференции СПИСОК-2023.
- Работа над данной темой привела к новому решению binary provenance problem на основе статистического подхода.
- Решения binary provenance problem посвящены определению высокоуровневых свойств программы и её происхождения (компилятор, язык, флаги компиляции и другое) по бинарному представлению.
- На данный момент новый подход заинтересовал лабораторию одной крупной китайской компании.