Нанесение водяных знаков на программное обеспечение

Архипов И. С., магистр математико-механического факультета СПбГУ, arkhipov.iv99@mail.ru

Аннотация

В докладе приводится исследование нового подхода внесения водяного знака в программное обеспечение на основе концепции МАК.

Введение

Для многих компаний, занимающихся разработкой программного обеспечения, пиратство программных продуктов является огромной проблемой. По оценкам, убытки от пиратства в индустрии в 2009 году составили более 50 миллиардов долларов [1]. С распространением интернет-технологий данная проблема становится всё более острой.

Для решения этой проблемы была разработана технология под названием «цифровой водяной знак». Это некая информация, встраиваемая в цифровой продукт без повреждения его эксплуатации. Водяные знаки стали популярны для защиты мультимедийных объектов, таких как изображения, видео и аудио. Активно ведутся разработки цифровых знаков применительно к программному обеспечению.

Как правило, водяные знаки программного обеспечения предназначены прежде всего для защиты авторских прав на цифровой продукт. Для этого водяной знак может содержать информацию о правообладателе продукта, чтобы доказать авторство, или о покупателе программного обеспечения, чтобы имелась возможность выявить нарушителя. Также водяные знаки могут использоваться для проверки целостности данных.

Работа с водяным знаком состоит из внесения водяного знака, то есть встраивания дополнительной информации в цифровой продукт, эксплуатации продукта без нарушения функциональности и распознания водяного знака.

На кафедре системного программирования СПбГУ в исследовательской группе под руководством М.В.Баклановского была разработана концепция МАК[1], открывающая новые возможности для преобразования программного кода. В рамках этой концепции ими был предложен новый подход нанесения водяных знаков.

[1] `https://youtu.be/n-YBy9iQnw4`. Accessed: 01-05-2023.

В докладе представлены результаты исследования, необходимые для реализации данного подхода, и прототип внесения и извлечения водяного знака.

Обзор

Статические и динамические водяные знаки

Водяные знаки могут быть классифицированы как статические и динамические. Статические водяные знаки внедрены в код и/или данные компьютерной программы. Динамические водяные знаки хранят информацию в состояниях выполнения программы.

В качестве примеров статических водяных знаков можно привести подходы, основанные на замене кода и/или данных в программе, перестановке кода, распределении регистров, изменении графа потока управления программы. Имеются и более специфичные техники.

В качестве примеров динамических водяных знаков можно привести такие методы, как "Пасхальные яйца внедрение водяных знаков в потоки и другие.

Характеристики водяных знаков

При работе с водяными знаками стоит чётко понимать, какими свойствами они обладают. В связи с этим в современной литературе [2] сложились следующие характеристики водяных знаков:

- Надёжность. Необходимо, чтобы его распознавание не имело ложных срабатываний.
- Требуемый объём ресурсов. Дополнительные память и время для работы водяного знака.
- Невидимость. Отвечает за сложность нахождения злоумышленником.
- Защита частей кода, а не всего проекта целиком.
- Устойчивость. Сохранение водяного знака при незначительных изменениях программы.

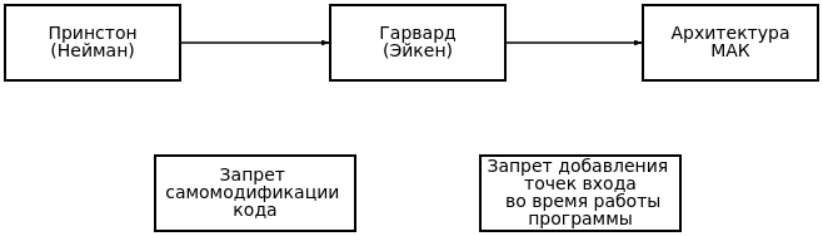

Рис. 1: Архитектурные концепции

Концепция МАК

На кафедре системного программирования СПбГУ в исследовательской группе под руководством М.В.Баклановского была разработана концепция МАК[2] (рис. 1). В ней на программы накладывается ещё одно важное ограничение: все возможные адреса переходов в программе должны быть известны до её исполнения. Авторы МАК характеризуют это ограничение как "запрет добавления точек входа во время работы программы".

В Гарвардской архитектуре нельзя определить точки входа программы. Из-за этого нельзя "раздвинуть" линейные участки кода, то есть оставить между линейными участками пустую память в секции кода, перемешать их, записать что-нибудь в промежутки между линейными участками, так как мы не можем заранее сказать, придёт ли управление в точку с конкретным адресом.

Водяные знаки в концепции МАК

В концепции МАК возможно "раздвинуть" линейные участки кода, то есть оставить между линейными участками пустую память в секции кода. В эту пустую память становится возможным записать водяной знак.

Способ нанесения водяного знака зависит от его цели. Если нужен видимый водяной знак, то можно просто его туда записать. Если нужен невидимый водяной знак, то можно замаскировать его под код. Если есть необходимость сделать водяной знак трудно извлекаемым, то можно применить обфускацию.

[2] `https://youtu.be/n-YBy9iQnw4`. Accessed: 01-05-2023.

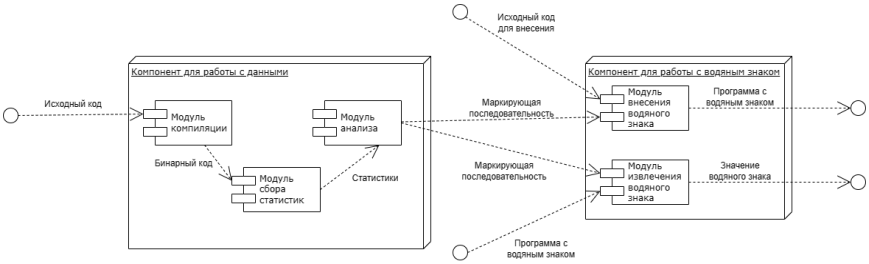

Рис. 2: Архитектура фреймворка

Нужен способ наносить целый комплекс водяных знаков с разными характеристиками. Для этого необходим способ понимать, что в данной области записан водяной знак. М.В.Баклановским и его группой такой способ был назван "каркасным водяным знаком".

Для маркировки водяного знака в конкретной секции кода предлагается использовать определённые "редкие" битовые последовательности, встречающиеся не во всех (или вообще не встречающиеся) бинарных файлах.

Для сокрытия водяного знака можно использовать статистические характеристики бинарного кода. В области расположения значения водяного знака можно добавить не несущие информации биты для обеспечения в данной области кода таких же статистических характеристик, как у обычного бинарного кода. Такая техника не даст злоумышленнику обнаружить водяной знак по отличающимся статистикам.

Цели и задачи

Целью работы является разработка, на основе концепции МАК, 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 МАК (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

Фреймворк состоит из двух компонент: компонент для работы с данными[3] и компонент для работы с водяным знаком[4]. Первый компонент содержит модуль компиляции, модуль сбора статистик и модуль анализа. Второй компонент содержит модуль внесения водяного знака и модуль извлечения водяного знака. Схематично архитектура фреймворка представлена на рис. 2.

Особенности реализации

В данном разделе кратко описана функциональность каждого модуля, его входные и выходные данные.

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

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 анализа на вход получает csv-файл с долями последовательностей и 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.

[3] `https://github.com/IvanArkhipov1999/Binary-code-statistics-research`. Accessed: 01-05-2023.

[4] `https://github.com/IvanArkhipov1999/watermark-prototype`. Accessed: 01-05-2023.

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 водяной 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.

Модуль извлечения водяного знака получает на вход бинарный файл, по определённой маркирующей каркасный водяной знак последовательности находит вложенное сообщение и выводит его.

Эксперименты

Выбор данных для экспериментов

В качестве данных для анализа была выбрана тестовая база GCC. Весь подготовленный набор данных лежит в отдельном репозитории[5].

Для компиляции был выбран один из наиболее популярных на данный момент компиляторов gcc.

Данные были подготовлены в формате elf.

Компиляция производилась для архитектур x86_64, arm64 и mips64el для демонстрации работоспособности подхода в различных архитектурах.

При сборе статистик и поиске "редких" последовательностей рассматривались доли всех бинарных последовательностей длиной до 10 включительно.

Архитектура x86_64

Для архитектуры x86_64 было найдено 13 подпоследовательностей длиной 8, 9 и 10 бит, распределение долей которых соответствует нормальному и 508 "редких" последовательностей длиной 8, 9 и 10 бит.

[5] `https://github.com/IvanArkhipov1999/Binaries-dataset`. Accessed: 01-05-2023.

Процент частоты нахождения в коде таких подпоследовательностей самый разный, что позволяет регулировать невидимость водяного знака. Полный перечень полученных данных лежит в отдельном репозитории[6].

Архитектура arm64

Для архитектуры arm64 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. Также было найдено 517 "редких" последовательностей длиной 8, 9 и 10 бит. Процент частоты нахождения в коде таких подпоследовательностей самый разный, что позволяет регулировать невидимость водяного знака.

Архитектура mips64el

Для архитектуры mips64el было найдено 7 подпоследовательностей длиной 8, 9 и 10 бит, распределение долей которых соответствует нормальному. Также найдено 406 "редких" подпоследовательностей, что значительно меньше, чем в случае x86_64 и arm64. Процент частоты нахождения в коде таких последовательностей самый разный, что позволяет регулировать невидимость водяного знака.

Заключение

В рамках данной работы был реализован прототип нанесения и извлечения водяного знака в концепции МАК и проведена его апробация.

Список литературы

[1] Business Software Alliance. Piracy impact study 2010: The economic benefits of reducing software piracy, 2010.

[2] Lim, Hyun-Il. "A performance comparison on characteristics of static and dynamic software watermarking methods."Indian Journal of Science and Technology 8.21 (2015): 1-6.

[6] `https://github.com/IvanArkhipov1999/Binaries-dataset`. Accessed: 01-05-2023.