Сычев Д.И. Жуган А.Е. Обфускация Kotlin программ с помощью техники Control Flow Flattening // Международный журнал информационных технологий и энергоэффективности. – 2023. –
Т. 8 № 4(30) с. 22–29

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004

ОБФУСКАЦИЯ KOTLIN ПРОГРАММ С ПОМОЩЬЮ ТЕХНИКИ CONTROL FLOW FLATTENING

[1]Сычев Д.И., [2]Жуган А.Е.
Санкт-Петербургский государственный университет телекоммуникаций имени профессора М.А. Бонч-Бруевича, Санкт-Петербург, Россия (193232, г. Санкт-Петербург, пр. Большевиков, 22, к. 1), e-mail: [1]s.denis_2001@mail.ru, [2]art.zhugan@yandex.ru

В эпоху, когда безопасность программного обеспечения и защита интеллектуальной собственности имеют первостепенное значение, обфускация исходного кода для предотвращения реверс инжиниринга стала критически важным аспектом разработки программного обеспечения. В данной статье исследуется применение техники Control Flow Flattening, мощного метода запутывания кода, к программам, написанным на Kotlin. Kotlin, современный и набирающий популярность язык программирования, известный своим лаконичным синтаксисом и функциональной совместимостью с Java. Однако особенности языка также создают уникальные проблемы, когда дело доходит до обфускации кода/
Ниже рассматривается концепция Control Flow Flattening, обращая внимание на её преимущества по сравнению с другими методами обфускации. Затем углубимся в особенности реализации Control Flow Flattening в Kotlin, описав требования, инструменты и пошаговый процесс. Сюда входит синтаксический анализ исходного кода Kotlin, идентификация и преобразование структур потока управления, а также повторная сборка кода.

Ключевые слова: Обфускация, Control Flow Flattening Kotlin, защита програмного обеспечения.

OBFUSCATION OF KOTLIN PROGRAMS USING THE CONTROL FLOW FLATTENING TECHNIQUE

[1]Sychev D.I., [2]Zhugan A.E.
St. Petersburg State University of Telecommunications named after Professor M.A. Bonch-Bruevich, St. Petersburg, Russia (193232, St. Petersburg, Bolshevikov Ave., 22, room 1), e-mail: [1]s.denis_2001@mail.ru, [2]art.zhugan@yandex.ru

In an era where software security and intellectual property protection are of paramount importance, obfuscating source code to hinder reverse engineering and tampering has become a critical aspect of software development. This paper explores the application of control flow flattening, a powerful obfuscation technique, to Kotlin programs. Kotlin, a modern and increasingly popular programming language, is known for its concise syntax and interoperability with Java. However, the language's features also present unique challenges when it comes to obfuscation.
We begin by providing an overview of Kotlin and the concept of control flow flattening, highlighting its benefits and comparison with other obfuscation techniques. We then delve into the implementation of control flow flattening in Kotlin, detailing the requirements, tools, and step-by-step process. This includes parsing the Kotlin source code, identifying and transforming control flow structures, and reassembling the flattened code. Through case studies and examples, we demonstrate the efficacy of control flow flattening in obfuscating Kotlin programs and analyze the performance and security implications of the technique.

Finally, we discuss the limitations of control flow flattening in Kotlin, potential countermeasures against obfuscation, and opportunities for future research and development. Our findings underscore the importance of software obfuscation in contemporary software development and encourage further exploration of advanced techniques to protect Kotlin programs from reverse engineering and unauthorized tampering.

Keywords: Obfuscation, Control Flow Flattening Kotlin, software protection.

Введение

В современном цифровом мире программное обеспечение повсеместно распространено и играет важную роль в различных отраслях и секторах. В результате защита интеллектуальной собственности программного обеспечения и предотвращение несанкционированного доступа к исходному коду стали критическими важными проблемами. Обфускация программного обеспечения — это метод, используемый для достижения вышеописанных целей путем преднамеренного изменения исходного кода или двоичных файлов программы, что делает ее более сложной для понимания, реверс инжиниринга или модификации без ущерба для ее функциональности. [1, 2]

Kotlin, разработанный JetBrains, — это современный язык программирования со статической типизацией, завоевавший значительную популярность благодаря своей лаконичности, выразительности и полной совместимости с Java. С ростом числа разработчиков, использующих Kotlin для различных приложений, в том числе для написания программ, на базе операционной системы Android, потребность в эффективных методах обфускации в экосистеме Kotlin становится более важной, чем когда-либо. Хотя Kotlin предлагает ряд функций, упрощающих разработку, эти же функции могут создавать проблемы при обфускации исходного кода.

Control Flow Flattening — это мощный метод обфускации, который направлен на то, чтобы скрыть логику программы путем преобразования структур потока управления, таких как циклы, условные операторы и вызовы функций, в единую сложную структуру управления. Это серьезно затрудняет анализ и реверс инжиниринг программы, предлагая дополнительный уровень безопасности и защиты интеллектуальной собственности. Ниже исследуется применение Control Flow Flattening к программам на Kotlin, подробно описывая процесс реализации.

1. Язык программирования Kotlin

Kotlin предлагает различные преимущества по сравнению с традиционными языками программирования, такими как Java. Некоторые из ключевых преимуществ включают в себя:

- *Краткость*: синтаксис Kotlin позволяет разработчикам выражать свои намерения с помощью меньшего количества строк кода, что делает код более читабельным и удобным для сопровождения.
- *Безопасность*: Kotlin имеет встроенные null safety функционал, снижающие вероятность NullPointerExceptions, распространенного источника ошибок в Java.
- *Выразительность*: Kotlin включает в себя различные функции, такие как лямбда-выражения, функции расширения и интеллектуальные приведения, которые позволяют разработчикам писать более выразительный и эффективный код.

- *Совместимость*: Kotlin может беспрепятственно взаимодействовать с кодом Java, позволяя разработчикам использовать существующие библиотеки Java и постепенно переносить существующие проекты на Kotlin без проблем с совместимостью.
- *Поддержка:* JetBrains, компания, стоящая за Kotlin, предоставляет отличную поддержку инструментов, включая IntelliJ IDEA, которая предлагает расширенные функции Kotlin для редактирования кода, рефакторинга и отладки.

Несмотря на свои многочисленные преимущества, Kotlin также сталкивается с уникальными проблемами, когда дело доходит до обфускации. Некоторые из проблем включают в себя:

- *Встроенные функции*: Kotlin позволяет разработчикам использовать встроенные функции, которые интегрируются непосредственно в вызывающий код во время компиляции. Эта функция может усложнить процесс обфускации, поскольку требует преобразования встроенного кода в контексте вызывающей функции.
- *Рефлекция:* поддержка отражения в Kotlin, которая позволяет исследовать и модифицировать структуру программы во время выполнения, может затруднить сокрытие логики и поведения программы посредством запутывания.
- *Вывод типов*: продвинутая система вывода типов Kotlin может затруднить запутывание программы, поскольку она требует сохранения правильной информации о типе в процессе обфускации.

2. Control flow flattening

Техника обфускации Control flow flattening включает в себя преобразование циклов, условных выражений и вызовов функций в единую сложную управляющую структуру, обычно реализуемую с использованием оператора switch и цикла диспетчера. Вместе, это затрудняет для злоумышленника следование точному порядку выполнения программного кода и понимания предполагаемого поведения программы.[3] Control flow flattening — это метод, не являющийся исключительным для конкретного языка программирования, что делает его подходящим для обфускации большого количества программ, написанных на различных языках программирования, включая Kotlin. По сравнению с другими доступными методами обфускации, Control flow flattenin является более устойчивым к автоматизированным инструментам деобфускации, поскольку он вводит нетривиальные преобразования, которые трудно вернуть вспять.[2] Так же, для лучшей защиты программного обеспечения, Control flow flattening можно использовать в сочетании с другими методами запутывания для создания многоуровневой стратегии защиты исходного кода.

Существует множество доступных методов обфускации, каждый из которых имеет свои сильные и слабые стороны. Ниже приведены наиболее распространенные методы:

Лексическая обфускация: переименование переменных, функций и классов, чтобы сделать код труднее для чтения и понимания. Хотя этот метод может быть прост в реализации, злоумышленнику относительно легко обойти его с помощью автоматизированных инструментов.

Обфускация данных: изменение представления данных, например, шифрование строк или изменение структур массивов, чтобы скрыть их значение или использование. Этот метод

может помочь защитить конфиденциальные данные, но может быть не так эффективен против опытных реверс-инженеров. [4]

Виртуализация кода: преобразование кода в промежуточное представление, которое выполняется специальной виртуальной машиной, что усложняет анализ. Хотя этот метод может обеспечить надежную защиту, он также может привести к значительным потерям производительности. [5]

3. Реализация Control flow flattening в Kotlin

Применение Control flow flattening к реальному языку программирования включает преобразование сложных структур потока управления в единую плоскую структуру. Этот процесс может иметь ряд проблем из-за определенных языковых конструкций и особенностей, усложняющих преобразование.

Циклы, такие как while, do или for, могут быть сложными для сглаживания, поскольку они включают несколько итераций с различными условиями. Сглаживание цикла требует преобразования его в плоскую структуру, которая сохраняет исходную функциональность, но затрудняет анализ и понимание кода злоумышленниками. Это включает в себя замену заголовков цикла операторами if, сохранения исходной логики потока и управление ключевой переменной.

```
var i = 0
while (i < 5) {
    println("Iteration: $i")
    i++
}
```

Рисунок 1 – Часть кода содержащего цикл while

```
var i = 0
var loopCondition = true
while (loopCondition) {
    if (i < 5) {
        println("Iteration: $i")
        i++
    } else {
        loopCondition = false
    }
}
```

Рисунок 2 – Код, Обфусцированный с помощью метода Control Flow Flatекштg.

В обфусцированной версии (Рисунок 2) исходный цикл while заменяется сглаженным циклом while и оператором if. Состояние цикла контролируется переменной ключевой loopCondition, что затрудняет анализ и понимание общей логики программы.

Ключевое слово when в Kotlin, схожее с оператором switch в C++, может быть сложно сгладить из-за спецефичного синтаксиса (Рисунок 3). Для обработки конструкций переключателей процесс преобразования должен гарантировать, что сведенный код сохраняет исходное поведение и правильно реализует изменения потока управления, продиктованные метками case.

```
val number = 3
when (number) {
    1 -> println("One")
    2 -> println("Two")
    3 -> println("Three")
    else -> println("Unknown")
}
```

Рисунок 3 – Ключевое слово when

```
val number = 3
var loopCondition = true
while (loopCondition) {
    loopCondition = when (number) {
        1 -> {
            println("One")
            false
        }
        2 -> {
            println("Two")
            false
        }
        3 -> {
            println("Three")
            false
        }
        else -> {
            println("Unknown")
            false
        }
    }
}
```

Рисунок 4 – Обфусцированная версия кода,

В запутанной версии исходная конструкция switch заменена уплощенным циклом while и вложенным оператором when. Условием цикла управляет переменная loopCondition, которая гарантирует сохранение исходного поведения, но затрудняет анализ кода.

Применение выравнивания потока управления к программам на Kotlin может потребовать обращения к специфичным для языка функциям, таким как встроенные функции. Kotlin поддерживает встроенные функции, которые позволяют компилятору заменять вызовы функций фактическим телом функции в месте вызова во время компиляции. Это может повысить производительность, но создает проблемы при применении выравнивания потока управления. Поэтому, разработчикам рекомендовано не использовать явные функции без явной необходимости.

Так же, Kotlin поддерживает вывод типов, что позволяет компилятору автоматически определять тип переменной или выражения на основе их использования. Это может усложнить процесс обфускации, поскольку при преобразовании структур потока управления необходимо учитывать предполагаемые типы.

Благодаря рассмотрению этих специфичных для Kotlin функций и предоставлению примеров кода сглаживание потока управления может быть эффективно применено к программам на Kotlin, что сделает их более устойчивыми к обратному проектированию и анализу.

Вывод

Выше были рассмотрены проблемы, возникающие при применении выравнивания потока управления к языку программирования Kotlin, предложены возможные решения и примеры кода для обработки специфичных для Kotlin функций.

Эффективность сглаживания потока управления можно повысить, комбинируя его с другими методами обфускации и постоянно обновляя и улучшая процесс.

Хотя сглаживание потока управления не является абсолютным решением вышеописанных уязвимостей, это важный шаг к обеспечению безопасности программ Kotlin. Решая проблемы и ограничения, связанные с этим методом, и изучая новые разработки в области обфускации кода, разработчики могут повысить безопасность и устойчивость своих программ на Kotlin к реверс-инжинирингу и вредоносным атакам.

Список литературы

1. Красов А. В. и др. Алгоритмы и методы защиты программного кода на базе обфускации //I-methods. – 2020. – Т. 12. – №. 1. – С. 1-12.
2. Шариков, П. И. Методика обфускации байт-кода java-приложения с целью его защиты от атак декомпиляцией / П. И. Шариков // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1: Естественные и технические науки. – 2022. – № 1. – С. 64-72
3. Гельфанд А. М., Гвоздев Ю. В., Штеренберг С. И. Исследования недостатков языков высокоуровнего программирования для осуществления скрытого вложения в исполнимые файлы //Актуальные проблемы инфотелекоммуникаций в науке и образовании. – 2015. – С. 295-297.

4. Цветков, А. Ю. Обеспечение безопасности в клиент-серверном Java приложении для учета и автоматической проверки лабораторных работ / А. Ю. Цветков, М. Е. Шалаева, М. А. Юрченко // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО 2019) : сборник научных статей VIII Международной научно-технической и научно-методической конференцияи : в 4 т., Санкт-Петербург, 27–28 февраля 2019 года. Том 1. – Санкт-Петербург: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, 2019. – С. 756-761.
5. Цветков, А. Ю. Поиск уязвимостей в программном обеспечении / А. Ю. Цветков, Ю. Б. Эллауи // Актуальные проблемы инфотелекоммуникаций в науке и образовании : сборник научных статей: в 4х томах, Санкт-Петербург, 24–25 февраля 2021 года / Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича. Том 1. – Санкт-Петербург: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, 2021. – С. 684-688.
6. Синельщиков В. С., Цветков А. Ю. Защита персональных данных на предприятии //Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО 2021). – 2021. – С. 653-657.
7. Цветков А. Ю., Рузманов Е. Ю. Рассмотрение тестирования на проникновение в задачах защиты информации //ББК 3 П27. – 2021. – С. 55.
8. Гельфанд А. М., Гвоздев Ю. В., Штеренберг С. И. Исследования недостатков языков высокоуровнего программирования для осуществления скрытого вложения в исполнимые файлы //Актуальные проблемы инфотелекоммуникаций в науке и образовании. – 2015. – С. 295-297.
9. Зимин А. Е., Косов Н. А. Обеспечение информационной безопасности в процессе создания и использования программ для ЭВМ //Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО 2017). – 2017. – С. 343-348.

References

1. Krasov A. V. et al. Algorithms and methods for protecting program code based on obfuscation //I-methods. - 2020. - T. 12. - No. 1. - pp. 1-12.
2. Sharikov, P. I. Java application bytecode obfuscation technique to protect it from decompilation attacks / P. I. Sharikov // Bulletin of the St. Petersburg State University of Technology and Design. Series 1: Natural and technical sciences. - 2022. - No. 1. - P. 64-72
3. Gelfand A. M., Gvozdev Yu. V., Shterenberg S. I. Investigations of high-level programming languages for hidden investment in executable files //Actual problems of infotelecommunications in science and education. - 2015. - pp. 295-297.
4. Tsvetkov, A. Yu. Security in a client-server Java application for accounting and automatic verification of laboratory work / A. Yu. Tsvetkov, M. E. Shalaeva, M. A. Yurchenko // Actual problems of infotelecommunications in science and education (APINO 2019): collection of scientific articles of the VIII International scientific-technical and scientific-methodical conference: in 4 volumes, St. Petersburg, February 27–28, 2019. Volume 1. - St. Petersburg: St. Petersburg State University of Telecommunications. prof. M.A. Bonch-Bruevich, 2019. - pp. 756-761.

5. Tsvetkov, A. Yu. Search for vulnerabilities in software / A. Yu. Tsvetkov, Yu. B. Ellaui // Actual problems of infotelecommunications in science and education: collection of scientific articles: in 4 volumes, St. Petersburg, February 24–25, 2021 year / St. Petersburg State University of Telecommunications. prof. M.A. Bonch-Bruevich. Volume 1. - St. Petersburg: St. Petersburg State University of Telecommunications. prof. M.A. Bonch-Bruevich, 2021. - pp. 684-688.
6. Sinelshchikov V. S., Tsvetkov A. Yu. Protection of personal data at the enterprise // Actual problems of infotelecommunications in science and education (APINO 2021). - 2021. - pp. 653-657.
7. Tsvetkov A. Yu., Ruzmanov E. Yu. Consideration of penetration testing for information protection //BBK 3 P27. - 2021. - pp. 55.
8. Gelfand A. M., Gvozdev Yu. V., Shterenberg S. I. Investigation of the shortcomings of high-level programming languages for the implementation of hidden attachments to executable files // Actual problems of infotelecommunications in science and education. - 2015. - pp. 295-297.
9. Zimin A. E., Kosov N. A. Ensuring information security in the process of creating and using computer programs // Actual problems of infotelecommunications in science and education (APINO 2017). - 2017. - pp. 343-348.