**Министерство образования и науки Российской Федерации**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Владимирский государственный университет имени Александра Григорьевича

и Николая Григорьевича Столетовых» (ВлГУ)

Институт информационных технологий и электроники

Кафедра информатики и защиты информации

**ОТЧЕТ**

**О ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**Разработка обфускатора языка C**

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н. доцент кафедры ИЗИ

подпись, дата Ю.М. Монахов

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ студент группы ИБ-122

подпись, дата Д.С. Чугров

Владимир 2025

Оглавление

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Стр. |
| Введение | | | 3 |
| Глава 1. Уточнение требований | | | 5 |
| 1.1. |  | Анализ и декомпозиция предметной области | 5 |
| 1.2. |  | Обзор аналогов | 6 |
|  | 1.2.1. | Obfuscator - LLVM | 6 |
|  | 1.2.2. | Themida | 6 |
|  | 1.2.3. | Tigress | 6 |
|  | 1.2.4. | Сравнение аналогов | 7 |
|  | 1.2.5. | Вывод из сравнения аналогов | 7 |
| 1.3. |  | Техническое задание | 7 |
|  | 1.3.1. | Тип обфускатора | 7 |
|  | 1.3.2. | Шифрование идентификаторов | 7 |
|  | 1.3.3. | Изменение потока управления программы | 8 |
|  | 1.3.4. | Язык программирования | 8 |
|  | 1.3.5. | Платформа | 8 |
| Глава 2. Обфускатор языка C | | | 9 |
| 2.1. |  | Обоснование выбора языка разработки и библиотек | 9 |
| 2.2. |  | Структура обфускатора языка С | 9 |
| 2.3. |  | Лексический анализатор языка С | 9 |
| 2.4. |  | Синтаксический анализатор языка С | 9 |
| 2.5. |  | Преобразователь графа состояния кода | 10 |
| 2.6. |  | Генератор «мертвого» и избыточного кода | 10 |
| 2.7. |  | Преобразователь идентификаторов и шифрование строк | 11 |
| 2.8. |  | Усложнение константных выражений | 11 |
| 2.9. |  | Приложение | 11 |
| Глава 3. Результат обфускации | | | 13 |
| Список использованной литературы | | | 15 |
| Ссылка на исходный код обфускатора языка C | | | 16 |

Введение

Обычно на ранних этапах разработки программного обеспечения перед компанией - разработчиком стоит задача непосредственно реализация его основных компонентов. Основные ресурсы идут на организацию эффективного взаимодействия разработчиков и построение понимаемого им программного кода проекта. Вопрос о его защите пока не ставится.

Однако, приближаясь к завершающей стадии разработки, проблема защиты исходного кода программного обеспечения становится актуальной.

Во- первых, риск раскрытия информации об алгоритмах и принципах работы ПО. Подобная утечка является особенно критичной для компаний, чья конкурентоспособность напрямую зависит от уникальности разработки.

Во - вторых, создание пиратского ПО, которое может нанести репутационный и финансовый ущерб. Более того, подобные нелегальные копии могут испортить пользовательский опыт.

В - третьих, при несанкционированном внесение изменений в исходный код могут привести нарушения функциональной составляющей программы.

Учитывая все выше перечисленное, становится очевидной необходимость использования средств защиты кода ПО. Они изменяют структуру и маскируют логику программы, тем самым усложняя анализ исходного кода. При этом его работоспособность сохраняется.

Анализируя все факторы, данная проблема является актуальной и применение компанией - разработчиком таких программных комплексов, обеспечивающих затрудненное понимание исходного кода и его работы ПО, носит необходимый характер.

Предметной областью данного проекта является разработка программного средства защиты исходного кода.

Цель данной разработки за время производственной практики создать программное обеспечение, способное изменять структуру исходного кода приложения клиента, маскируя основную его логику и усложняя анализ. Для достижения этой цели должны быть решены следующие задачи:

1) Разработать лексический анализатор языка С;

2) Реализовать систему управления потоком кода;

3) Разработать механизмы обфускации, включая изменение структуры кода, шифрование строк и добавление мусорного кода для усложнения анализа;

4) Обеспечить сохранение функциональности программы после применения методов обфускации.

Глава 1. Уточнение требований

1.1. Анализ и декомпозиция предметной области

Обфускация - это приведение исходного кода к такому виду, при котором сохраняется его функциональность, но затрудняется его анализ и понимание алгоритмов работы. Соответственно обфускатор кода - это программное средство, позволяющие производить подобные манипуляции с кодом. Основная цель обфускации - приведение кода в такое состояние, при котором его понимание становится практически невозможным.

В своей работе А.В. Чернов приводит следующие способы обфускации:

1. Преобразования, изменяющие только внешний вид программы. К ним относятся такие действия как удаление комментарий и переименование идентификаторы.

2. Преобразования структур данных. Они включают себя изменение иерархии наследования классов, преобразование массивов.

3. Преобразование потока управления программы. Представляет собой изменение графа потока к состоянию несводимому.

4. Внесение недостижимого кода. Достигается добавлением фрагментов, которые никогда не выполнятся.

5. Добавление «мертвого» кода.

6. Внесение избыточного кода.

7. Преобразования, нацеленные против методов декомпиляции.

Как правило, обфускация не рассматривается в качестве полноценного метода защиты кода, потому что она не предотвращает нелегальное использование кода, а лишь затрудняет взлом. Она чаще всего применяется в паре с шифрованием для усиления защиты кода. Более того, при использовании определенных запутывающих методов есть вероятность увеличения времени выполнения программы.

1.2. Обзор аналогов

1.2.1. Obfuscator - LLVM

**Obfuscator-LLVM** – это набор инструментов для обфускации кода. Он основан на фреймворке LLVM (Low Level Virtual Machine), соответственно имеет встроенный компилятор. Здесь обфускация происходит на промежуточном уровне представления.

Среди реализованных функций можно выделить следующие:

* добавление блока кода в начале основной части программы с дальнейшей реализацией условного перехода к нужным инструкциям;
* основная часть программы случайным образом заполняется ненужными инструкциями. (Реализована генерация ложных условий, бесполезных циклов, глобальных и локальных переменных);
* замена стандартных бинарных операторов (сложение, вычитание и логические операторы) более сложными инструкциями/последовательностями инструкций, результат которых функционально эквивалентен. Реализован случайный выбор замены. Данный метод доступен только для целочисленных операций.

1.2.2. Themida

Данное ПО представляет собой коммерческий продукт от компании Oreans Technologies. Основная задача программы - защита исполняемых файлов операционной системы Windows от нелегального использования, копирования и других видов вмешательств. Действует на уровне машинного кода.

1.2.3. Tigress

Tigress - инструмент, предназначенный для защиты программ на языке С. Данный обфускатор представляет собой транслятор «исходный код в исходный код», на вход исходный код программы на языке C и возвращает модифицированный исходный код, который функционально эквивалентен оригиналу, но значительно сложнее для анализа и понимания.

1.3 Сравнение аналогов

Можно выделить следующие сравнительные характеристики:

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Obfuscator - LLVM | Themida | Tigress |
| Уровень обфускации | Промежуточный уровень | Уровень машинного кода | Исходный код в исходный код |
| Влияние на производительность | Может снижаться из-за сложных преобразований | Минимальное влияние | Может снижаться из-за добавления кода |
| Платформа | Кроссплатформенный | Windows | Linux |
| Тип лицензии | Бесплатная | Есть платная версия и бесплатная демо-версия | Бесплатная |

1.4 Вывод из сравнения аналогов

Исходя из сравнения аналогов выше, можно сделать вывод, что на данный момент отсутствуют специализированные средства для платформы Windows, которые позволяют напрямую изменять исходный код программ с сохранением их функциональности и при этом выдавать на выходе эквивалентные файлы.

1.3. Техническое задание

1.3.1 Тип обфускатора

Необходимо разработать обфускатор языка С, представляющий собой транслятор исходного кода в исходный код. На вход поступают файлы программы, на выходе - эквивалентные.

1.3.2. Шифрование идентификаторов

Обфускатор должен выполнять замену понятных идентификаторов на бессмысленные наборы символов, при этом должна сохраняться их уникальность и место их использования.

1.3.3. Изменение потока управления программы

Необходимо предлагать пользователю степень изменения графа состояния программы для усложнения анализа её логики. Возможные настройки:

* Легкий режим - минимальное влияние на производительность;
* Средний режим - умеренное влияние на производительность;
* Агрессивный режим - существенное влияние на производительность.

В зависимости от выбора режима генерируется определенное количество «мертвого», недостижимого и избыточного кода от исходного в процентном соотношении.

1.3.4. Язык программирования

Для реализации обфускатора следует использовать язык программирования Java с использованием инструмента для создания лексического анализатора и генератора код ANTLR.

1.3.5. Платформа

Обфускатор должен быть приспособлен для работы на операционной системе Windows.

Глава 2. Обфускатор языка C

2.1. Обоснование выбора языка разработки и библиотек

Для реализации обфускатора был выбран язык разработки Java, т.к. он позволяет запускать приложения на различных платформах без необходимости вносить изменения в код. Более того, Java работает на виртуальной машине (JVM), которая эффективно управляет ресурсами и предоставляет высокую производительность приложений.

Также необходим такой инструмент как ANTLR. Он позволяет генерировать парсеры и анализаторы на основе реализованной грамматики. Данное средство поможет облегчить разработку лексического и синтаксического анализаторов, а также позволит использовать Listener для построения логики обфускатора.

2.2. Структура обфускатора языка C

Для эффективного приведение кода в такое состояние, при котором его понимание становится практически невозможным обфускатор должен состоять из следующих частей:

* Лексический анализатор языка C;
* Синтаксический анализатор языка C;
* Преобразователь графа состояния кода;
* Генератор «мертвого» и избыточного кода;
* Преобразователь идентификаторов и шифрование строк;
* Приложение.

2.3. Лексический анализатор языка C

Лексический анализатор - первая стадия обфускации. Он выполняет обработку исходного кода программы и разбивает его на лексемы: ключевые слова, идентификаторы, литералы, операторы и разделители.

Лексический анализатор реализован с помощью генераторов парсеров ANTLR. Соответственно грамматика языка C написана в файле C.g4. На ее основе сгенерированы лексер и парсер, необходимые для остальных частей обфускатора, и находятся в папке Tokens.

2.4. Синтаксический анализатор языка C

Синтаксический анализатор - следующая часть обфускатора. Он принимает поток токенов от лексического анализатора для проверки на соответствие синтаксису языка программирования C.

Реализовано с помощью сгенерированных файлов на основе грамматики.

2.5. Преобразователь графа состояния кода

За первоначальное построение графа состояния кода отвечает класс Graph. Он принимает на вход синтаксическое дерево, полученное в результате работы синтаксического анализатора. По ходу его обхода каждой части приписывается соответствующая метка графа. Также, в зависимости от степени обфускации добавляются специальные метки для генерации «мертвого» и избыточного кода.

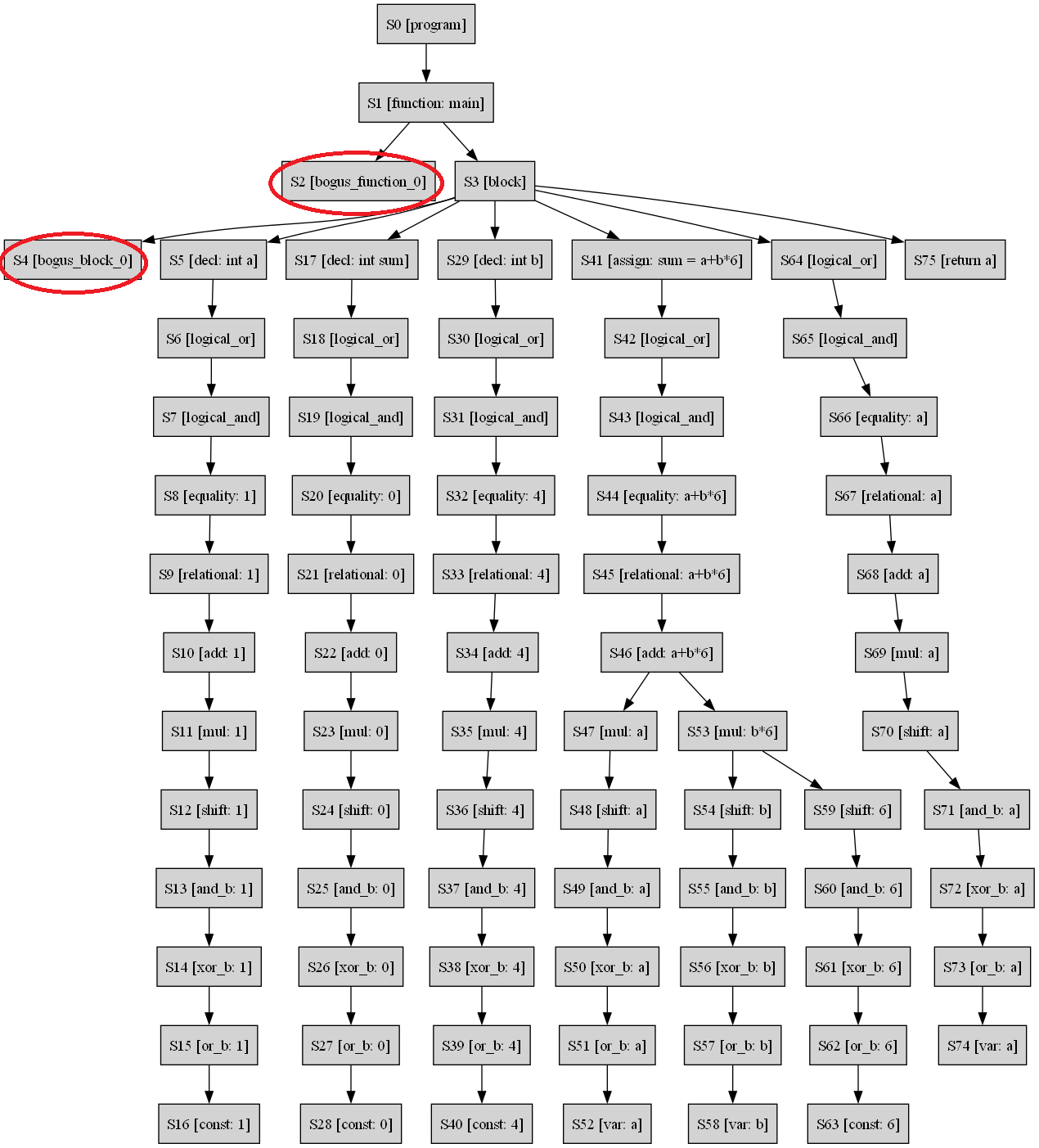


Рис. 1. Граф состояния кода с добавленными метками для генерации кода

2.6. Генератор «мертвого» и избыточного кода

За работу генератора «мертвого» и избыточного кода отвечают классы Generation и CRandomCodeGenerator. Первый - анализирует полученный граф состояния кода, считывая метки для генерации кода. Благодаря им вызываются методы из CRandomCodeGenerator для вставки конкретного вида «мертвого» и избыточного кода.

К ним относятся:

* Объявление новых переменных и действий с ними;
* Генерация циклов и условий;
* Вывод сообщений, не представляющих никакой полезной информации;
* Создание новых методов.

2.7. Преобразователь идентификаторов и шифрование строк

Данная часть является заключительным этапом обфускации. Полученный модифицированный код снова проходит лексический и синтаксический анализ для извлечения идентификатор переменных, методов и строк.

Шифрование строки происходит с помощью метода XOR с генерированным ключом.

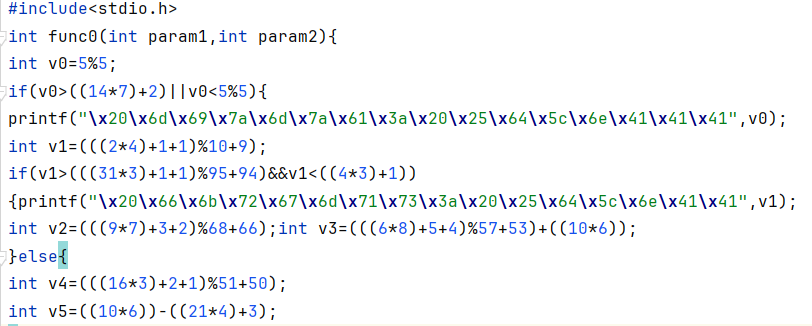


Рис. 2. Часть преобразованного кода

2.8. Усложнение константных выражений

Одновременно с последним этапом происходит усложнение константных выражений. В ходе лексического и синтаксического анализа можно извлечь значения выражений и разложить их на более сложные, в результате дающие тоже самое.

2.9. Приложение

Для удобства использования обфускатора реализовано приложение, принимающие файлы с кодом на языке программирования C. Его запуск происходит с помощью соответствующего файла .exe.

Для начала пользователю необходимо выбрать степень офускации (от 1 до 3).

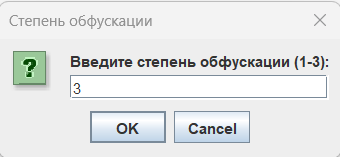


Рис. 3. Окно выбора степени офускации

Далее необходимо выбрать нужный файл расширением .c, содержащий код на языке программирования C.

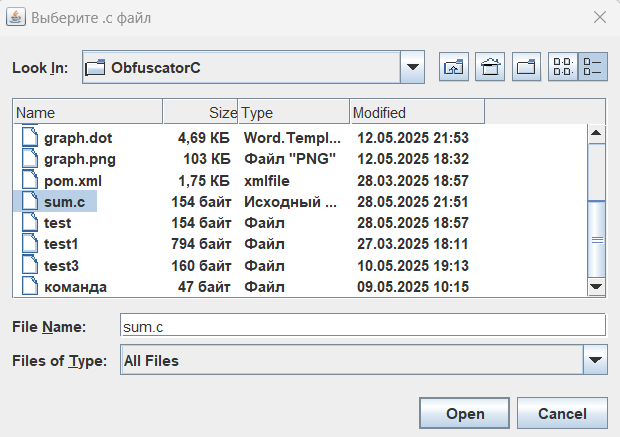


Рис. 4. Выбор исходного файла

В конце нужно выбрать место для сохранения файла с обфусцированным кодом.

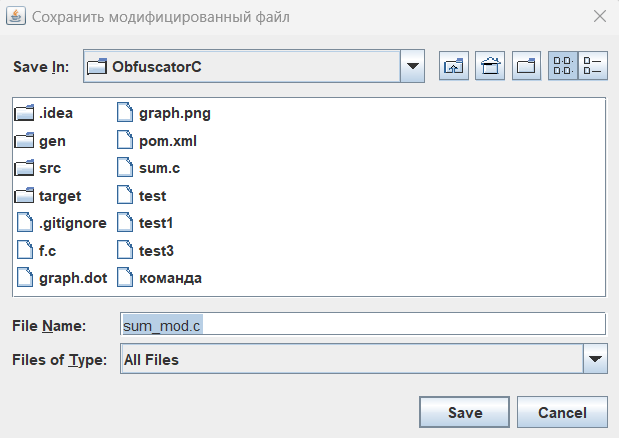


Рис. 5. Сохранение полученного файла в результате обфускации

Глава 3. Результат обфускации

Чтобы убедится насколько изменился поток управления программы необходимо провести испытание. В качестве тестового исходного кода взят код ниже:

#include <stdio.h>  
  
int main() {  
 int a = 1;  
 int sum = 0;  
 int b = 4;  
 sum = a + b \* 6;  
 printf("%d\n", sum);  
 return sum;  
}

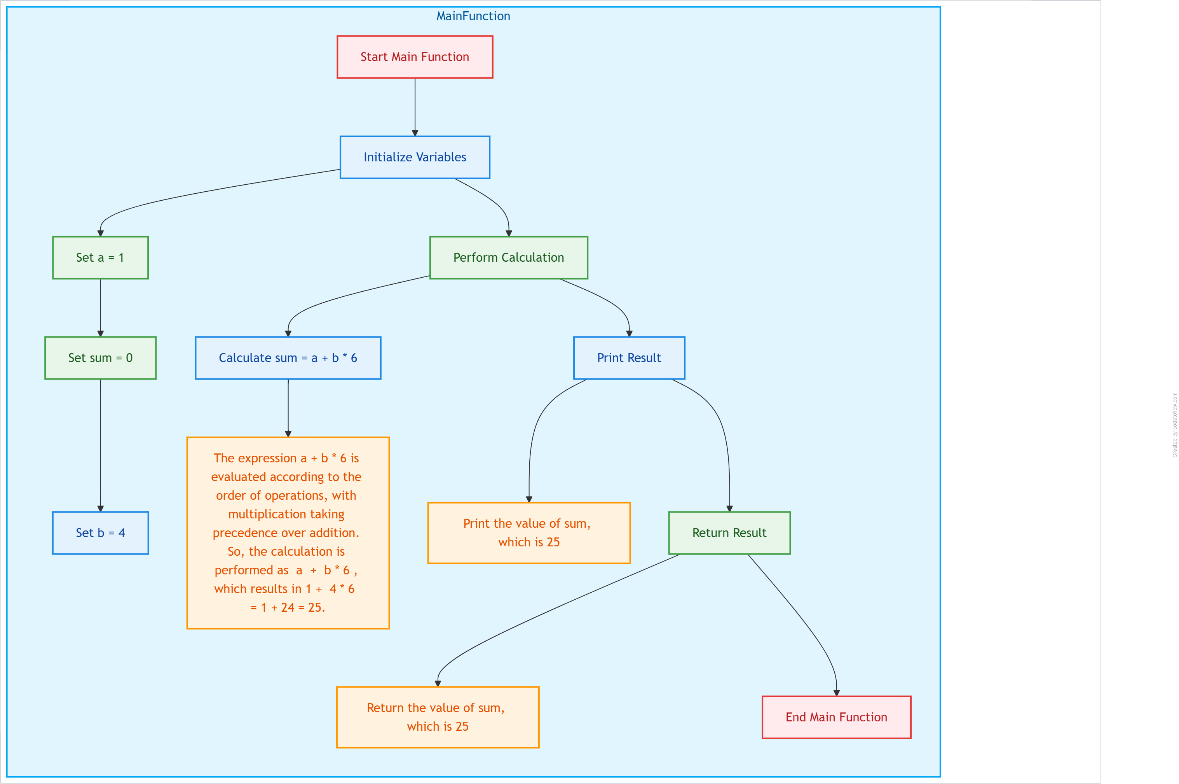


Рис.6. Поток управления исходного кода

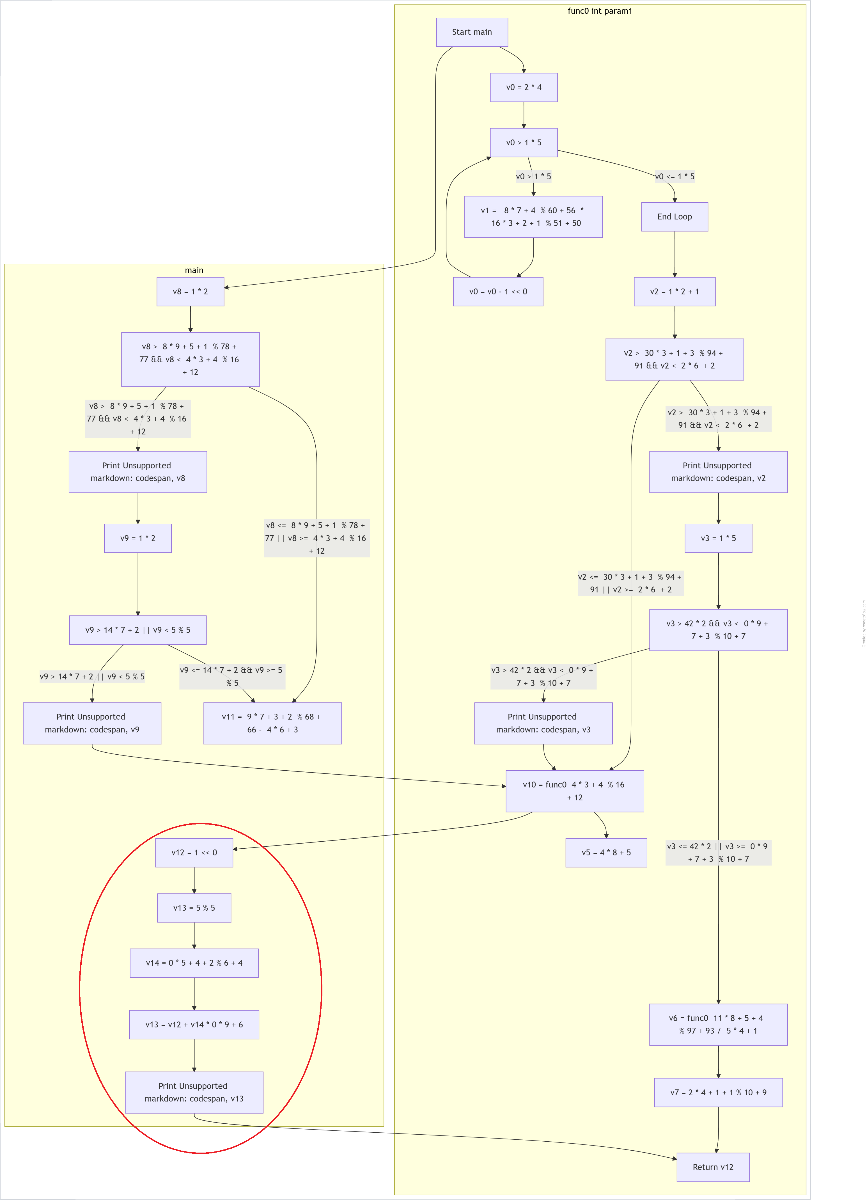


Рис. 7. Поток управления преобразованного исходного кода

Можно заметить, что вместо одного исходного метода main у нас добавился метод func0, содержащий цикл и условия. Также main стал содержать новые переменные, среди которых содержится вызов новой функции. За счет этого происходит маскировка изначальной его логики (выделена красным на рис.7).

Исходя из выше сказанного, можно сказать, что данные преобразования точно приведут к затруднению обратного инжиниринга модифицированного кода, но не повлияют на ожидаемый результат работы программы.

Список использованной литературы

1) А.В. Чернов. Анализ запутывающих преобразований программ. Труды Института системного программирования РАН, том 3, 2002, стр. 7-38.

2) Курмангалеев Ш. Ф., Корчагин В. П., Матевосян Р. А. Описание подхода к разработке обфусцирующего компилятора. Труды Института системного программирования РАН, 2012, стр. 67 - 76.

3) Christian Collberg; JasvirNagra Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection Addison-Wesley Professional Pub. Date: July 24, 2009 Print ISBN-10: 0-321-54925-2, 748 стр.

4) Obfuscator-LLVM– Software Protection for the Masses / Pascal Junod, Julien Rinaldini, Johan Wehrli, Julie Michielin // Proceedings of the IEEE/ACM 1st International Workshop on Software Protection, SPRO’15, Firenze, Italy, May 19th, 2015 / Ed. by Brecht Wyseur.– IEEE, 2015.–– P. 3–9.

5) Suresh Anjali J., Sankaran Sriram. A Framework for Evaluation of Software Obfuscation Tools for Embedded Devices // Applications and Techniques in Information Security / Ed. by Lejla Batina, Gang Li.– Singapore : Springer Singapore, 2020.–– P. 1–13.

6) UnThemida: Commercial obfuscation technique analysis with a fully obfuscated program / Jae Hyuk Suk, Jae Yung Lee, Hongjoo Jin et al. // Software- Practice and Experience.–– 2018.––.–– Vol. 48, no. 12.–– P. 2331–2349.–– Funding Information: This work was sup ported by Institute for Information Communications Technology Pro motion (IITP) grant funded by the Korea government (MSIT) (No. 2014-6-00910, Study on Security of Cryptographic Software). Pub lisher Copyright: © 2018 John Wiley Sons, Ltd.

7) Themida.–– URL: https://themida.com.

8) The tigress C obfuscator.–– URL: https://tigress.wtf.

9) Ахо, Альфред В., Лам, Моника С., Сети, Рави, Ульман, Джеффри Д. Компиляторы: принципы, технологии и инструментарий, 2-е изд., 2008.

10) Седжвик, Роберт, Уэйн, Кевин. С28 Алгоритмы на Java, 4-е изд., 2013.

Ссылка на исходный код обфускатора языка C

