ТЕХНИЧЕСКИЕ НАУКИ

УДК 004.056

Анализ методов защиты кода от ревёрс-инжиниринга: разработка концепции

Д.О. Антонов
Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация

Аннотация
Рассматривается проблема защиты программного обеспечения от ревёрс-инжиниринга, которая становится все более актуальной в условиях развития цифровых технологий. Анализируются основные угрозы ревёрс-инжиниринга. Обосновывается идея о том, что существующие методы защиты кода часто не обеспечивают достаточной стойкости к современным атакам. Целью статьи является разработка концепции эффективной защиты кода, основанной на комбинированном применении различных методов и адаптации к конкретным угрозам. Особое внимание уделено механизму выбора методов защиты на основе анализа угроз и компромиссов.

Ключевые слова: ревёрс-инжиниринг, защита кода, обфускация, шифрование, антиотладочные методы, анализ угроз, концепция защиты, многоуровневая защита

Для цитирования. Антонов Д.О. Анализ методов защиты кода от ревёрс-инжиниринга: разработка концепции. *Молодой исследователь Дона.* 2026;11(1):18–22.

Analysis of Code Protection Methods against Reverse Engineering: Development of a Concept

Daniil O. Antonov
Don State Technical University, Rostov-on-Don, Russian Federation

Abstract
The article studies the problem of protecting software from reverse engineering. The relevance of this problem increases due to continuous development of digital technologies. The main threats caused by reverse engineering were analysed. The problem of inability of existing code protection methods to provide sufficient resistance to the modern cyber attacks was substantiated. The article aims at developing an efficient code protection concept based on the combined use of various methods, and adaptation to the specific threats. Particular attention was paid to the procedure of selecting protection methods based on the analysis of threats and trade offs.

Keywords: reverse engineering, code protection, obfuscation, encryption, anti-debugging techniques, analysis of threats, protection concept, multi-layered protection

For Citation. Antonov DO. Analysis of Code Protection Methods against Reverse Engineering: Development of a Concept. *Young Researcher of Don*. 2026;11(1):18–22.

Введение. В условиях стремительного развития цифровых технологий и расширения спектра программных решений — от финансовой сферы до оборонной — защита программного обеспечения от несанкционированного анализа приобретает стратегическое значение. Одной из наиболее опасных и актуальных угроз в области информационной безопасности является ревёрс-инжиниринг — процесс исследования исполняемого файла с целью восстановления его логики, структуры и зачастую исходного кода. Данная методика позволяет злоумышленникам получить доступ к интеллектуальной собственности разработчика, выявить уязвимости, изменить поведение приложения или создавать нелегальные копии продукта.

Научная проблема, лежащая в основе настоящего исследования, состоит в отсутствии универсальной и устойчивой к разнообразным векторам атак системы защиты кода, способной эффективно противостоять усложняющимся методам ревёрс-инжиниринга. Несмотря на множество разработанных подходов — таких как обфускация, шифрование, внедрение антиотладочных механизмов и использование виртуальных машин — каждый из них в отдельности обладает ограниченной стойкостью. Отсутствие комплексной стратегии, учитывающей синергию методов и анализирующей их совокупную эффективность, создаёт широкие возможности для реализации атак.

Актуальность темы обусловлена необходимостью создания новых концепций, ориентированных на гибридное применение защитных методик и адаптацию к современным реалиям — включая рост числа автоматизированных инструментов ревёрс-инжиниринга, распространение вредоносного ПО и активность киберпреступных группировок. Связь проблемы с ключевыми задачами цифровой безопасности проявляется в обеспечении конфиденциальности алгоритмов, защите авторских прав, противодействии репаккингу (процессу модификации программного обеспечения для изменения функционала или обхода защиты) и предотвращении внедрения вредоносного кода.

Цель статьи заключается в разработке концепции защиты кода от ревёрс-инжиниринга, основанной на комбинированном использовании различных подходов, усиленных их взаимной интеграцией и адаптацией к конкретным угрозам.

Основная часть. Ревёрс-инжиниринг программного обеспечения (или обратная разработка) представляет собой процесс анализа исполняемой программы с целью восстановления её структуры, логики работы и, при возможности, исходного кода [1]. Этот процесс может осуществляться без доступа к исходным текстам, то есть исключительно на основе бинарных файлов. Бинарный файл (исполняемый файл) — это скомпилированный файл с машинным кодом, предназначенный для выполнения процессором. Он не предназначен для чтения человеком (поскольку после компиляции исходный код преобразуется в низкоуровневые инструкции, выглядящие как бессвязный поток байтов), содержит структуру с метаданными и может быть представлен в различных форматах в зависимости от операционной системы.

Обзор угроз. На современном этапе развития цифровых технологий ревёрс-инжиниринг стал мощным инструментом, активно используемым как в легитимных целях (анализ вредоносного ПО, тестирование на уязвимости, восстановление утерянного кода), так и в рамках злонамеренной деятельности — кражи интеллектуальной собственности, обхода механизмов лицензирования и защиты. С учётом масштабов применения подобных техник важно понимать, какие конкретно угрозы они несут для безопасности программных решений.

Обзор основных угроз, связанных с ревёрс-инжинирингом, представлен в таблице 1 [2].

Таблица 1

Обзор основных угроз ревёрс-инжиниринга

Угроза	Описание	Возможные последствия
Декомпиляция и дизассемблирование кода	Анализ исполняемого кода для восстановления псевдокода, близкого к исходному.	Раскрытие бизнес-логики, алгоритмов защиты, механизмов лицензирования и других критически важных компонентов. Упрощение дальнейшего анализа и поиска уязвимостей.
Модификация исполняемых файлов (патчинг)	Внесение изменений в исполняемые файлы, включая удаление защиты, обход проверок лицензий и внедрение вредоносного кода.	Несанкционированное использование программного обеспечения, нарушение работы программы, внедрение вредоносного функционала, обход систем безопасности.
Анализ поведения программы во время выполнения	Исследование работы программы в процессе её выполнения с использованием отладчиков, трассировщиков, эмуляторов и инструментов перехвата (например, IDA Pro, Ghidra, OllyDbg) для отслеживания критических участков и обхода защиты.	Выявление уязвимостей, обход механизмов защиты и лицензирования, понимание логики работы программы на уровне выполнения, что может способствовать дальнейшей модификации или эксплуатации.

Ревёрс-инжиниринг представляет собой не только техническую, но и экономическую угрозу: утечка интеллектуальной собственности способна нанести серьёзный ущерб разработчику, особенно в условиях конкурентного рынка.

Обзор методов защиты

Современные подходы к защите программного кода от ревёрс-инжиниринга можно классифицировать по задачам, связанным с предотвращением несанкционированного анализа, модификации и копирования приложений.

Обфускация

Данный метод направлен на усложнение структуры кода при сохранении его функциональности. Распространённые приёмы включают переименование идентификаторов, вставку ложного или недостижимого кода, а также виртуализацию логики программы. Эти подходы существенно затрудняют статический анализ (изучение кода без его выполнения), не меняя архитектуру приложения. Обфускация эффективна для защиты программ на языках высокого уровня, например Java. К недостаткам относят возможное снижение производительности и наличие инструментов, частично автоматизирующих деобфускацию [3]. На сегодняшний день она остаётся базовым, но уже недостаточным средством защиты — полезным как первый рубеж, однако неспособным самостоятельно противостоять квалифицированному анализу, особенно с использованием специализированного ПО.

Антиотладочные методы

Этот класс техник нацелен на обнаружение и противодействие инструментам динамического анализа, таким как отладчики или эмуляторы. Эффективные приёмы включают самопроверку на наличие отладчика, анализ временных задержек, перемещение кода в память с динамическим размещением, а также внедрение нестандартных инструкций, не обрабатываемых типичными средствами анализа. Подобные методы затрудняют установку точек останова и могут дезориентировать исследователя. Однако их реализация требует осторожности, чтобы не нарушить легальную работу программы [4]. Антиотладочные техники крайне полезны в условиях активной угрозы динамического анализа, особенно для защиты ценных алгоритмов. Тем не менее их избыточное применение способно привести к нестабильности приложения.

Шифрование

Позволяет защитить код от статического анализа благодаря хранению в зашифрованном виде с последующей расшифровкой во время выполнения. На практике применяются алгоритмы, такие как AES или RSA, а также дополнительные модули, например аппаратные ключи, защищающие криптографические материалы. Шифрование эффективно для исполняемых файлов и конфиденциальных алгоритмов, однако требует обеспечения безопасной расшифровки и управления ключами, что может повлиять на производительность [5]. Данный метод особенно ценен для критичных компонентов, но его использование оправдано только при наличии чётко выстроенной инфраструктуры управления ключами — в противном случае риски компрометации возрастают.

Виртуализация

Один из наиболее устойчивых к анализу подходов предполагает выполнение критичных частей программы в рамках виртуальной машины с индивидуальной архитектурой. Это подразумевает создание собственной системы команд и интерпретатора, что делает анализ кода чрезвычайно трудоёмким. Подобная защита особенно актуальна для распределённых систем или облачных сервисов, где обеспечение безопасности логики может быть децентрализовано. Недостатком выступает высокая сложность реализации и потребность в значительных вычислительных ресурсах [6]. Виртуализация представляет собой перспективное направление, однако из-за сложности внедрения её применение оправдано не всегда — например, только для компонентов с наивысшей степенью важности, поскольку требует как высокой квалификации, так и существенных затрат на поддержку.

Комбинирование методов

Использование различных способов защиты в сочетании друг с другом позволяет значительно повысить уровень безопасности. Например, обфускация может эффективно комбинироваться с антиотладочными мерами и шифрованием, что усложняет процесс атаки и требует больше времени и ресурсов, чем применение одного метода. Комбинированные решения также обеспечивают гибкость адаптации защиты в зависимости от специфики угроз и архитектуры приложения [3].

Разработка механизма выбора методов защиты на основе анализа угроз

Концепция создания надёжной системы защиты кода от ревёрс-инжиниринга включает следующие ключевые этапы:

1. Идентификация критически важных компонентов: необходимо определить части программного обеспечения, представляющие наибольшую ценность для злоумышленников — такие как алгоритмы лицензирования, ключевая бизнес-логика и механизмы шифрования.

2. Оценка вероятности и потенциального ущерба от каждой угрозы: следует проанализировать, какие угрозы наиболее актуальны для данного типа ПО, а также возможные последствия их реализации. Например, для широко распространяемого коммерческого продукта кража лицензионных механизмов способна нанести значительный финансовый ущерб.

3. Выбор методов защиты, наиболее эффективных против выявленных угроз: процесс предполагает детальное сопоставление угроз с доступными мерами защиты, что позволяет создать целенаправленную систему безопасности. Соответствие выявленных рисков и применяемых мер защиты представлено в таблице 2.

Таблица 2

Сопоставление выявленных угроз и методов защиты программного кода

Угроза	Методы защиты	Пояснение
Декомпиляция, дизассемблирование	Обфускация, шифрование кода	Обфускация усложняет читаемость кода, делая его трудным для понимания. Шифрование делает код нечитаемым до момента его расшифровки во время выполнения.
Модификация исполняемых файлов (патчинг)	Шифрование, контроль целостности, виртуализация	Шифрование делает патчинг неэффективным без ключа. Контроль целостности позволяет обнаружить любые изменения кода. Виртуализация изолирует критичный код от внешнего вмешательства.
Динамический анализ	Антиотладочные методы	Антиотладка позволяет обнаружить отладчики и эмуляторы и изменить поведение программы, затрудняя изучение её логики.

Учет компромиссов: при выборе методов защиты важно учитывать их влияние на производительность приложения, сложность реализации и стоимость разработки. Основные компромиссы различных методов защиты от ревёрс-инжиниринга представлены в таблице 3.

Таблица 3

Основные компромиссы методов защиты от ревёрс-инжиниринга

Метод защиты	Основные компромиссы
Обфускация	Возможное снижение производительности, частичная автоматизация деобфускации, низкая эффективность против динамического анализа.
Шифрование	Влияние на производительность (дешифровка), сложность управления ключами, потенциальная уязвимость при компрометации ключей, необходимость защищенной расшифровки в памяти.
Контроль целостности кода	Может потребовать дополнительных ресурсов для проверки, потенциальные ложноположительные срабатывания, не предотвращает саму модификацию.
Антиотладочные методы	Риск нестабильной работы приложения, особенно в кроссплатформенных средах, возможность обхода опытными аналитиками, низкая эффективность против статического анализа.
Виртуализация	Значительное снижение производительности, очень высокая сложность и стоимость реализации и поддержки, высокие требования к вычислительным ресурсам.

Реализация многоуровневой защиты: наиболее эффективным подходом является комбинирование нескольких методов защиты. Например, обфускация может быть использована вместе с антиотладочными методами и шифрованием для создания более надежного барьера против ревёрс-инжиниринга.

На рис. 1 представлена структурная схема разработанного механизма выбора методов защиты от ревёрс-инжиниринга.

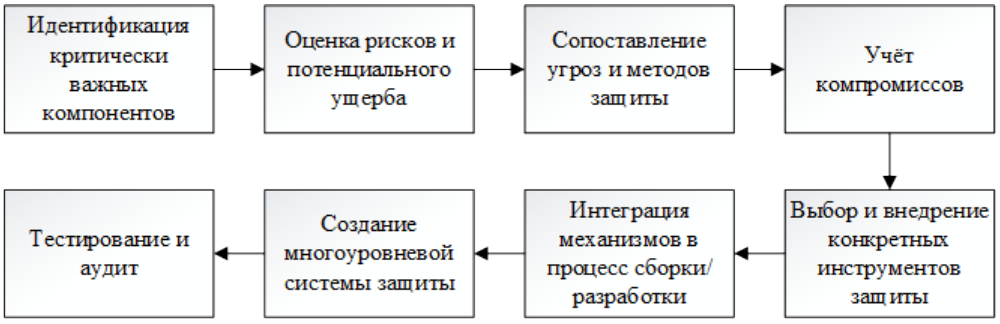

Рис. 1. Схема концепции защиты от ревёрс-инжиниринга

Заключение. Проведённый анализ подчёркивает критическую важность комплексного подхода к защите программного обеспечения от ревёрс-инжиниринга, учитывающего как статический, так и динамический анализ. В отличие от традиционных методов, ориентированных на отдельные средства защиты, предложенная концепция рассматривает взаимодействие различных механизмов, адаптируясь к специфике угроз, что способствует формированию более эффективной и гибкой стратегии. Это особенно актуально в условиях стремительного развития технологий ревёрс-инжиниринга и роста числа кибератак, где надёжная защита кода играет ключевую роль в обеспечении безопасности данных, конфиденциальности и сохранении интеллектуальной собственности.

Список литературы

1. *Реверс-инжиниринг: что это такое и для чего он нужен.* INNOPOL. URL: https://innopol.tech/tpost/b8o9tfv1x1-revers-inzhiniring-chto-eto-takoe-i-dlya (дата обращения: 10.11.2025).

2. *Как реверс-инжиниринг помогает обнаруживать уязвимости.* СБЕР. URL: https://www.sberbank.ru/ru/person/kibrary/articles/kak-revers-inzhiniring-pomogaet-obnaruzhivat-uyazvimosti (дата обращения: 10.11.2025).

3. Красов А.В., Зуев И.П., Карельский П.В., Радынская В.Е., Гераськина В.С. Алгоритмы и методы защиты программного кода на базе обфускации. *I-methods*. 2020;12(1):1–12.

4. Нечта И.В. Метод защиты программ от отладочных точек останова посредством исполнения фрагментов кода в общем буфере. *Вестник СибГУТИ*. 2022;(3(59):48–55. https://doi.org/10.55648/1998-6920-2022-16-3-48-55

5. Лебедев Р.С. Защита от копирования программного обеспечения. *Инновации. Наука. Образование.* 2021;(43):433–439.

6. Маркин Д.О., Макеев С.М. Система защиты терминальных программ от анализа на основе виртуализации исполняемого кода. *Вопросы кибербезопасности*. 2020;(1(35):29–41. https://doi.org/10.21681/2311-3456-2020-01-29-41

Об авторе:

Даниил Олегович Антонов, студент кафедры «Кибербезопасность информационных систем» Донского государственного технического университета (344003, Российская Федерация, г. Ростов-на-Дону, пл. Гагарина, 1), mr.dany2003@mail.ru

Конфликт интересов: **автор заявляет об отсутствии конфликта интересов.**

Автор прочитал и одобрил окончательный вариант рукописи.

About the Author:

Daniil O. Antonov, Student of the Department of Cybersecurity of Information Systems, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, Russian Federation), mr.dany2003@mail.ru

Conflict of Interest Statement: **the author declares no conflict of interest.**

The author has read and approved the final manuscript.