Научная статья
УДК 004.4; DOI: 10.61260/2218-13X-2025-1-109-119

ГЕНЕТИЧЕСКИЙ РЕВЕРС-ИНЖИНИРИНГ ПРОГРАММ ДЛЯ ПОИСКА УЯЗВИМОСТЕЙ

✉**Израилов Константин Евгеньевич.**
Санкт-Петербургский Федеральный исследовательский центр Российской академии наук, Санкт-Петербург, Россия
✉***konstantin.izrailov@mail.ru***

Аннотация. Работа представляет собой систематизацию основных научных результатов, полученных в процессе авторского исследования, посвященного вопросу реверс-инжиниринга программ в интересах поиска в них уязвимостей. Отличительной особенностью исследования является интеллектуализация процесса за счет применения генетических алгоритмов. Все результаты предназначены для разрешения соответствующих проблемных вопросов области безопасности программного обеспечения и состоят из следующих:

1) методология реверс-инжиниринга программного обеспечения для выявления уязвимостей;

2) модель жизненного цикла программы с разноуровневыми уязвимостями с позиции эволюции представлений;

3) концепция генетической деэволюции представлений программы для выявления уязвимостей;

4) научно-методический и алгоритмический инструментарий генетической декомпиляции представлений программы для выявления уязвимостей;

5) архитектура системы для проведения генетической деэволюции представлений программы с интеллектуальным функционалом по поиску разноуровневых уязвимостей.

Приводится методологическая схема результатов (в графическом виде), а также краткая характеристика каждого из них (новизна, теоретическая и практическая значимость).

Ключевые слова: безопасность программного обеспечения, уязвимости, реверс-инжиниринг, декомпиляция, искусственный интеллект, генетические алгоритмы

Для цитирования: Израилов К.Е. Генетический реверс-инжиниринг программ для поиска уязвимостей // Науч.-аналит. журн. «Вестник С.-Петерб. ун-та ГПС МЧС России». 2025. № 1. С. 109–119. DOI: 10.61260/2218-13X-2025-1-109-119.

Scientific article

GENETIC REVERSE-ENGINEERING OF A PROGRAMS TO FIND VULNERABILITIES

✉**Izrailov Konstantin E.**
Saint-Petersburg Federal research center of the Russian academy of sciences, Saint-Petersburg, Russia
✉***konstantin.izrailov@mail.ru***

Abstract. The work is the main scientific results systematization obtained in the course of the author's research devoted to the issue of programs reverse engineering in order to search for vulnerabilities in them; at the same time, a distinctive feature of the research is the intellectualization of the process through the genetic algorithms usage. All results are intended to resolve corresponding problematic issues in the field of software security and consist of the following:

1) methodology of software reverse engineering to identify vulnerabilities;

2) model of the life cycle of a program with multi-level vulnerabilities from the standpoint of the representations evolution;

3) concept of program representations genetic de-evolution to identify vulnerabilities;

4) scientific, methodological and algorithmic instrument for genetic decompilation of program representations to identify vulnerabilities;

5) architecture of the system for carrying out program representations genetic de-evolution with intelligent functionality for searching for multi-level vulnerabilities.

A methodological diagram of the results is given (in graphic form), as well as a brief description of each of them (novelty, theoretical and practical significance).

Keywords: software security, vulnerabilities, reverse engineering, decompilation, artificial intelligence, genetic algorithms

For citation: Izrailov K.E. Genetic reverse-engineering of a programs to find vulnerabilities // Scientific and analytical journal «Vestnik Saint-Petersburg university of State fire service of EMERCOM of Russia». 2025. № 1. P. 109–119. DOI: 10.61260/2218-13X-2025-1-109-119.

Введение

Безопасность области программного обеспечения (ПО), в том числе, определяется отсутствием уязвимостей в программном коде, что является актуальнейшей проблематикой сферы информационной безопасности. Для разрешения проблемных вопросов существуют различные подходы, основным из которых считается их непосредственное выявление в коде программ (имеющем как конечную выполняемую, так разрабатываемую промежуточную форму). При этом сам факт обнаружения уязвимости не сделает ПО более безопасным, поскольку требуется ее устранение, как правило, с помощью модификации исходного кода (ИК) и пересборки проекта. Тем не менее с применением данного подхода сопряжена научная проблема (Проблема), формулируемая в нотации «возможности vs потребности». С одной стороны, для анализа, как правило, имеется только низкоуровневый машинный код программы (МК) или аналогичный ему ассемблерный код (АК), поскольку ее ИК и другие высокоуровневые представления не указываются производителями ПО (например, из соображений коммерческой тайны). С другой стороны, поиск уязвимостей необходимо осуществлять не только в конечном представлении программы, но и во всех предыдущих (особенно, если именно в них внедрялись уязвимости). То есть суть Проблемы заключается в том, что имеющиеся представления программы слабо подходят для поиска в них уязвимостей и требуется получение более высокоуровневых и человекоориентированных. И хотя определенные решения по получению предыдущих представлений (то есть их деэволюция) и существуют, тем не менее все они имеют лишь частное приложение в ограниченных условиях. Разрешение же Проблемы гипотетически позволит качественно повысить эффективность поиска уязвимостей в различных представлениях программы, что и считается целью всего авторского исследования (Исследование).

Согласно современным трендам и успехам в сфере информационных технологий искусственный интеллект показал свою успешность в различных областях, исключением не стала и информационная безопасность. Как следствие, для разрешения Проблемы весь процесс последовательности деэволюций ее представлений, так называемый реверс-инжиниринг (РИ), был рассмотрен на предмет возможности применения искусственного интеллекта в части генетических алгоритмов (ГА); выбор последних обосновывается их особенностями в части решаемых задач – оптимизация популяции особей (то есть множества ИК), заданных набором генов (то есть конструкций языка программирования) для «выращивания» среди них наилучшего представителя (то есть того, который компилируется в заданный МК). Тема же исследования была сформулирована следующим образом: «Реинжиниринг программ в интересах поиска уязвимостей на базе генетических алгоритмов».

Далее представлена обобщающая схема полученных основных научных результатов (ОНР), за которой последует краткое описание сути и характеристик каждого из них; расширенное ознакомление с результатами может быть получено из авторских публикаций, ссылки на которые также приводятся.

Методологическая схема результатов

Методологическая взаимосвязь всех ОНР в рамках проведенного Исследования отображена в схематичном виде на рисунке.

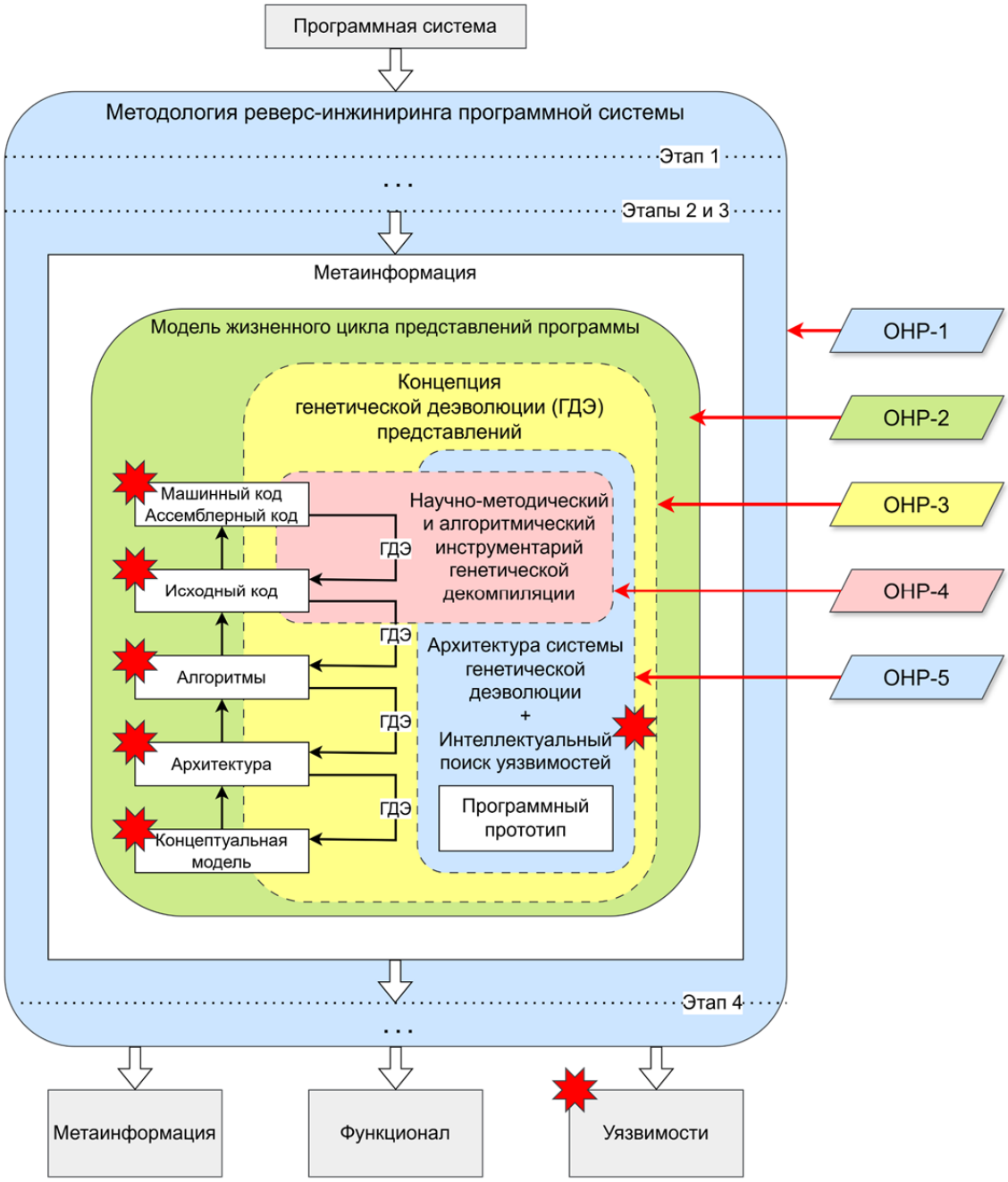

Рис. **Методологическая схема основных научных результатов**

Согласно данной схеме в Исследовании было получено пять результатов (постфиксно пронумерованных в хронологическом порядке получения). Изначально была создана методология реверс-инжиниринга программной системы (ОНР-1), выполнение наиболее

сложной группы шагов которой потребовало наличия модели жизненного цикла представлений программы (ОНР-2), позволившей сформировать концепцию генетической деэволюции представлений (ОНР-3). Для реализации концепции с использованием ГА был синтезирован соответствующий научно-методический и алгоритмический инструментарий (ОНР-4), реализуемый согласно архитектуре системы генетической деэволюции (ОНР-5); успешность экспериментов над программным прототипом архитектуры послужила обоснованием работоспособности и всех пяти полученных ОНР. С этой позиции тема Исследования в виде реинжиниринга программ является основополагающей (с позиции сложности проведения) частью более общей методологии реверс-инжиниринга их системы.

Описание и характеристики результатов

Опишем далее каждый из полученных ОНР в предложенной ранее форме и степени детализации, указывая в заголовках их полные названия (в отличие от методологической схемы, где для лучшей читаемости даны укороченные версии).

ОНР-1. Методология реверс-инжиниринга ПО для выявления уязвимостей

Поскольку практически любое современное ПО представляет собой сложную систему программ, выполняемых на подходящей аппаратной платформе (а не только как обособленные приложения в операционных системах), то требуется целый набор методов и средств, имеющих сложные связи (например, внешние интерфейсы программ позволят определить логику их информационного обмена) и применяемых в определенной нетривиальной последовательности. Это же, в свою очередь, приводит к необходимости наличия целой методологии реверс-инжиниринга такой системы (Методология) [1–3] с определением фактического функционала, метаинформации о ее программах и их уязвимостях, что является ОНР-1. Затем первые два полученных результата могут быть проанализированы экспертом или автоматизированными средствами, что позволит дополнить или уточнить третий. Программная система в общем случае может быть единым бинарным образом, выполняемым на некотором устройстве (например, являться монолитной прошивкой BIOS или содержать отдельную файловую систему с UEFI-драйверами). Метаинформация о программе представляет собой детализацию таких ее представлений, как МК и ИК, алгоритмы и архитектура. Сама Методология состоит из четырех этапов, содержащих 35 шагов (каждый из которых имеет свой набор средств автоматизации в той или иной степени).

Методология впервые охватывает весь накопленный опыт области, дополняя его отсутствующими методами и средствами их выполнения (в виде отдельных шагов).

Теоретическая значимость Методологии заключается в проведенном обобщении и систематизации основных шагов, составляющих процесс РИ, которые записаны как в виде схемы, так и аналитически.

Также Методология (методы и средства) позволяет проектировать и непосредственно проводить РИ программных систем, имеющих различную структуру, аппаратное размещение и особенности функционирования в интересах поиска уязвимостей. Для каждого шага Методологии указана его сложность и форма выполнения (ручная или автоматическая), что позволяет делать предварительную оценку трудоемкости всего процесса такого РИ.

Поиск уязвимостей наиболее эффективен в тех представлениях, в которых они были внедрены, то есть, как правило, в высокоуровневых; при этом сама программа находится в низкоуровневом машиноориентированном. Однако преобразование программы с повышением уровня абстракции представлений, и, следовательно, человекоориентированности (например, из АК и МК в ИК) является наиболее сложной группой шагов Методологии. Как следствие, существует проблемный вопрос такого преобразования между соседними представлениями, что инициировало получение следующего ОНР.

ОНР-2. Модель жизненного цикла программы с разноуровневыми уязвимостями с позиции эволюции представлений

Для «понимания» схемы преобразования представлений программы с учетом различных их вариаций (например, блок-схем алгоритмов, псевдокода, двухмерных структурных схем, байт-кода и пр.) и потенциально внедряемых уязвимостей была построена модель жизненного цикла программы (Модель) [4–6], представленная в графическом и аналитическом виде, – ОНР-2. Соответствующая ей схема описывает «эволюционный» процесс создания программы от идеи (максимально абстрактной, высокоуровневой) до выполняемого кода (минимально абстрактного, низкоуровневого). При этом на базе Модели была предложена новая классификация уязвимостей – по их структурному уровню: концептуальные, высокоуровневые, среднеуровневые, низкоуровневые и атомарные. Анализ Модели позволил обосновать возможность обратного («деэволюционного») преобразования представлений – противоположного к классическому процессу программного инжиниринга; например, получение из МК не только ИК программы, но и ее алгоритмов, архитектуры и т.д.

Предложенная Модель отличается от аналогов высоким уровнем абстракции, наличием единого набора признаков представлений и учетом различных сценариев процесса. Построенная же на ее базе классификация уязвимостей в принципе является новой, отражающей точку их встраивания и, соответственно, программный язык описания.

Модель устанавливает двунаправленную (де)эволюционную взаимосвязь между различными представлениями программы в соответствии с этапами и сценариями ее разработки, расширяя тем самым теоретические знания в предметной области.

С позиции практической значимости для составления «маршрута» деэволюции представлений (с отражением в них соответствующих классов уязвимостей и форм их поиска) данная Модель может послужить «картой», поскольку фактически является графом представлений и однонаправленных переходов между ними.

Уязвимости, изначально заложенные в ИК (или более высокоуровневые представления), будут иметь некоторое отражение и в МК, что позволит их выявить в случае обратного преобразования. Механизм же такой деэволюции является отдельным проблемным вопросом области, в интересах разрешения которого был получен следующий ОНР.

ОНР-3. Концепция генетической деэволюции представлений программы для выявления уязвимостей

Поскольку какие-либо общие подходы к деэволюции представлений программы практически отсутствуют, а имеются лишь частные решения для получения псевдо-ИК (не всегда корректного или компилируемого) из МК, то была разработана концепция такой деэволюции (Концепция) [7–11] – ОНР-3. Поскольку в основе Концепции лежит искусственный интеллект в части ГА, то деэволюция была естественно названа «генетической». Суть данного процесса заключается в итеративном решении оптимизационной задачи по подбору кода программы в предыдущем представлении, который бы в точности компилировался в заданный код в текущем представлении. При этом последовательное применение реализаций Концепции позволит преобразовать программу с уязвимостями в то представление, в котором они были внедрены и где их поиск наиболее эффективен, то есть осуществить генетический реверс-инжиниринг (ГРИ).

В отличие от аналогичных подходов, непосредственно преобразующих текущее представление в предыдущее (экспертно, алгоритмически или иным контекстно-зависимым способом), генетическая деэволюция (ГДЭ) имеет противоположную направленность, так как производит множественные преобразования программы из предыдущего представления в текущее. При этом механизм в основе подхода отличается инвариантностью к представлениям, поскольку не зависит от их синтаксиса (и, в том числе процессорной архитектуры).

Обзор релевантных научных работ, гипотетические и эмпирические эксперименты, а также теоретические изыскания позволили обосновать возможность и целесообразность проведения РИ программ с помощью ГА.

Предложенная Концепция позволяет синтезировать соответствующие практические методы РИ программ, а также их работоспособные реализации; ее же инвариантность дает возможность применять решения не только для классического получения ИК из МК или АК (достаточно близких друг к другу), но и для любых других пар соседних представлений.

Ряд теоретических исследований и гипотетических экспериментов обосновал возможность и целесообразность применения Концепции. Тем не менее для подтверждения ее реализуемости потребовалось создание соответствующего инструментария в части ГА, изложенного ниже.

ОНР-4. Научно-методический и алгоритмический инструментарий генетической декомпиляции представлений программы для выявления уязвимостей

В качестве проверки и подтверждения работоспособности Концепции были выбраны два представления программы: МК (а точнее, тождественный ему АК) и ИК, который необходимо из него восстанавливать, созданный для этого научно-методический инструментарий (Инструментарий) [12–18] и является ОНР-4. Выбор данной пары представлений обоснован тем, что на сегодняшний день большинство программ имеет вид именно МК, а множество существующих «успешных» средств поиска уязвимостей и экспертных «практик» предназначены для анализа ее ИК. Так, преобразование «МК → ИК» классически называется *декомпиляцией*, задача же дальнейшей деэволюции (в алгоритмы, архитектуру и т.д.), хоть пока и не рассматривается в полной мере, будет крайне актуальна уже в ближайшем будущем.

ГА в классическом (а точнее, существующем) виде слабо применимы для осуществления ГДЭ даже этих двух представлений, что потребовало как модификации самого ядра алгоритмов, так и создания используемых ими инструментов, состоящих из следующих (И_n):

И_1) метод генетической декомпиляции (ГДК) на базе модифицированного ГА;

И_2) модель представления ИК (особь в ГА);

И_3) алгоритм работы с моделью ИК (для операций скрещивания и мутации в ГА);

И_4) алгоритм генерации ИК (для инициализации первоначальной популяции в ГА);

И_5) алгоритм прогнозирования размера ИК по размеру его АК (для выбора длины хромосомы в ГА);

И_6) алгоритм предсказания константных значений в ИК (для снижения разновидности особей в ГА);

И_7) метрика сравнения двух АК (для функции приспособленности в ГА).

Инструментарий производит необходимые преобразования как «чистого» МК программы, так и находящихся в нем уязвимостей, что позволяет их восстановить в «терминах» человекоориентированного ИК. Ключевая особенность Инструментария (а, следовательно, и самой Концепции) заключается в его независимости от особенностей конкретных представлений, для чего, в частности, используется формальный синтаксис кода программы, передаваемый в качестве параметра. Сама же программа представляет собой путь по графу этих синтаксических правил (ГСП), а ее запись в рамках ГА (то есть как особи, определяемой последовательностью ген хромосомы) соответствует выбору альтернативных путей разбора правил по графу.

Инструменты, составляющие ОНР-4, обладают следующей новизной:

– впервые получен метод ГДК (И_1), который в отличие от аналогичных методов декомпиляции является полностью интеллектуальным и основан на работе ГА; также в самом ГА основные операции скрещивания и мутации модернизированы путем разделения их влияния на форму (внешний вид) и содержание (внутреннюю логику) программы;

– модель представления ИК (И_2) отличается от аналогичных (например, древовидных) записью программы в виде последовательности чисел, определяющих альтернативные ветки по ГСП;

– алгоритмы работы с моделью ИК (И_3) и его генерации (И_4) являются новыми, поскольку функционируют на базе авторского модельного представления программы как полного пути по ГСП;

– впервые получены алгоритмы прогнозирования размера ИК от размера его АК (И_5) и предсказания константных значений в ИК (И_6);

– метрика сравнения двух АК отличается от аналогичных чувствительностью оценки близости двух текстов и независимостью от форматов их представления, а также наличием коэффициентов учета влияния позиций строк и символов.

Теоретическая значимость полученного Инструментария заключается в получении в синтаксически-оптимальной модели ИК программы, заданной как путь по ГСП для представления (И_2). При этом была установлена статистическая зависимость между размерами программ в двух соседних представлениях – ИК и МК (И_5), а также вычислена статистическая вероятность появления константных значений в ИК программе (И_6). Полученная формула определения близости двух АК, представленных как списки символьных строк (И_7), расширяет класс метрик сравнения текстовых представлений данных, инвариантных к их структуре.

В рамках ГДК часть инструментов обладает следующей практической значимостью: собственно, метод ГДК необходим для непосредственной (то есть практической) реализации решений на базе ГА, осуществляющих получение ИК из МК и АК (И_1), алгоритм работы с моделью ИК позволяет осуществлять его «форм-содержательную» модификацию в рамках ГДЭ (И_3), а алгоритм генерации ИК предназначен для создания синтаксически-корректных экземпляров программы согласно заданным условиям (в том числе псевдослучайных заданной длины) в интересах получения начального эволюционного поколения для ГДЭ (И_4).

Практическая реализация как самой модифицированной версии ГА, так и используемого Инструментария, требует разработки соответствующей программной архитектуры (а также ее прототипирования), описанной ниже.

ОНР-5. Архитектура системы для проведения ГДЭ представлений программы с интеллектуальным функционалом по поиску разноуровневых уязвимостей

Реализация Концепции в интересах преобразования МК/АК в ИК с помощью созданного Инструментария потребовала проектирования соответствующего программного блока архитектуры (Архитектура) [19–22], что является ОНР-5. Термин «блок» применяется в том смысле, что полный процесс ГРИ через все множество представлений может быть построен по схеме «pipe-line» путем составления последовательности таких типовых Архитектур, то есть каждый блок принимает на вход результаты предыдущего и передает свои результаты следующему. Также Архитектура содержит авторский интеллектуальный модуль поиска уязвимостей в данном представлении, работа которого основана на анализе генетического представления программы и сравнении его с заданным шаблоном, что имеет ряд преимуществ по сравнению с аналогичными способами. Так, уязвимость в некотором представлении описывается с помощью языка программирования (даже если он графической или иной формы) и, следовательно, может быть задана через его формальный синтаксис – как путь по ГСП, что соответствует некоторой последовательности ген, то есть сигнатуре уязвимости (в данном представлении). Следовательно, данная последовательность может быть найдена в другой, соответствующей, программе, что и определяет общую идею поиска уязвимостей; при этом, модуль поиска является интеллектуальным в том смысле, что построен на объектах и принципах ГА.

Архитектура является результатом программного проектирования авторской концепции ГДЭ и необходимого для нее Инструментария и, следовательно, обладает новизной. При этом в нее введен модуль, использующий новый интеллектуальный способ сигнатурного поиска уязвимостей в программе на основе их авторского «генетического» представления. В отличие же от существующих контекстно-зависимых аналогов, генетическая сигнатура уязвимости не зависит от конкретного представления программы и его синтаксиса, а определяется выбранным ГСП для текущего.

Важным теоретическим вкладом интеллектуального поиска в область безопасного ПО является качественное развитие сигнатурных способов поиска уязвимостей (как и их хранения) в сторону генетического представления программ.

Предложенная Архитектура предназначена как для обособленного использования, так и для создания схемы из последовательности таких унифицированных блоков, позволяя тем самым проводить весь ГРИ на практике. В частности, реализация архитектурного блока «МК/АК → ИК» позволяет решать актуальнейшую задачу декомпиляции бинарных программ для поиска в них уязвимостей как автоматически в процессе, так и экспертно или автоматизировано по ИК. При этом ведение базы генетических сигнатур позволит идентифицировать уязвимости независимо от синтаксиса представлений.

Архитектура для получения ИК по МК программы с поиском в нем уязвимостей уже имеет реализацию в виде программного прототипа, который показал свою работоспособность и эффективность применения в ряде сценариев.

Заключение

В статье представлен краткий обзор пяти ОНР, полученных в процессе авторской исследовательской работы, направленной на развитие направления РИ программ. При этом целевым назначением РИ является поиск уязвимостей (что отражено в каждом из результатов), а его ключевой особенностью – применение ГА.

Результаты следующие:

1) методология РИ ПО, восстанавливающая из единой программной системы необходимую информацию о каждой из них и об их взаимодействии;
2) модель жизненного цикла программы с описанием эволюции представлений, связанной с частью шагов методологии;
3) концепция обратных преобразований между этими представлениями;
4) научно-методический и алгоритмический инструментарий для реализации концепции в случае получения ИК из МК/АК;
5) программная архитектура блока для создания работоспособного прототипа такого преобразования.

Помимо высокорейтинговых международных научных форумов, все ОНР были апробированы на международных и российских конференциях, где получили одобрение со стороны научной общественности.

Методологическая корректность проведенной работы, а также целостность, новизна и значимость (для теории и практики) ее ОНР позволяют говорить и о завершенности всего исследования (по крайне мере, на данном ключевом этапе). Работоспособный же прототип такого ГРИ обосновывает достоверность результатов с практической точки зрения.

Работа выполнена при частичной финансовой поддержке бюджетной темы FFZF-2025-0016.

Список источников

1. Израилов К.Е. Методология реверс-инжиниринга машинного кода. Часть 1. Подготовка объекта исследования // Труды учебных заведений связи. 2023. Т. 9. № 5. С. 79–90. DOI: 10.31854/1813-324X-2023-9-5-79-90.

2. Израилов К.Е. Методология реверс-инжиниринга машинного кода. Часть 2. Статическое исследование // Труды учебных заведений связи. 2023. Т. 9. № 6. С. 68–82. DOI: 10.31854/1813-324X-2023-9-6-68-82.

3. Израилов К.Е. Методология реверс-инжиниринга машинного кода. Часть 3. Динамическое исследование и документирование // Труды учебных заведений связи. 2024. Т. 10. № 1. С. 86–96. DOI: 10.31854/1813-324X-2024-10-1-86-96.

4. Израилов К.Е. Моделирование программы с уязвимостями с позиции эволюции ее представлений. Часть 1. Схема жизненного цикла // Труды учебных заведений связи. 2023. Т. 9. № 1. С. 75–93. DOI: 10.31854/1813-324X-2023-9-1-75-93.

5. Израилов К.Е. Моделирование программы с уязвимостями с позиции эволюции ее представлений. Часть 2. Аналитическая модель и эксперимент // Труды учебных заведений связи. 2023. Т. 9. № 2. С. 95–111. DOI: 10.31854/1813-324X-2023-9-2-95-111.

6. Израилов К.Е. Средство построения жизненного цикла программы с потенциальными уязвимостями: св-во о рег. программы для ЭВМ № 2023664967 от 11 июля 2023 г.

7. Израилов К.Е. Концепция генетической деэволюции представлений программы. Часть 1 // Вопросы кибербезопасности. 2024. № 1 (59). С. 61–66. DOI: 10.21681/2311-3456-2024-1-61-66.

8. Израилов К.Е. Концепция генетической деэволюции представлений программы. Часть 2 // Вопросы кибербезопасности. 2024. № 2 (60). С. 81–86. DOI: 10.21681/2311-3456-2024-2-81-86.

9. Израилов К.Е. Концепция генетической декомпиляции машинного кода телекоммуникационных устройств // Труды учебных заведений связи. 2021. Т. 7. № 4. С. 10–17. DOI:10.31854/1813-324X-2021-7-4-95-109.

10. Израилов К.Е. Применение генетических алгоритмов для декомпиляции машинного кода // Защита информации. Инсайд. 2020. № 3 (93). С. 24–30.

11. Израилов К.Е. Программное средство генерации экземпляров исходного кода программы согласно формальному синтаксису языка программирования: св-во о рег. программы для ЭВМ (в печати).

12. Израилов К.Е. Прогнозирование размера исходного кода бинарной программы в интересах ее интеллектуального реверс-инжиниринга // Вопросы кибербезопасности. 2024. № 4 (62). С. 13–25. DOI: 10.21681/2311-3456-2024-4-13-25.

13. Израилов К.Е. Исследование распределения константных значений в исходном коде программ на языке С // Труды учебных заведений связи. 2024. Т. 10. № 5. С. 119–129. DOI: 10.31854/1813-324X-2024-10-5-118-128.

14. Израилов К.Е. Методика оценки эффективности средств алгоритмизации, используемых для поиска уязвимостей // Информатизация и связь. 2014. № 3. С. 39–42.

15. Izrailov K. GREMC: Genetic Reverse-Engineering of Machine Code to Search Vulnerabilities in Software for Industry 4.0. Predicting the Size of the Decompiling Source Code // The proceedings of International Russian Smart Industry Conference. Sochi, 2024. P. 622–628. DOI: 10.1109/SmartIndustryCon61328.2024.10515515.

16. Израилов К.Е. Программное средство построения распределения константных значений в исходном коде программ на языке С: св-во о рег. программы для ЭВМ (в печати).

17. Израилов К.Е. Программное средство оценки размера исходного кода по его машинному коду: св-во о рег. программы для ЭВМ (в печати).

18. Израилов К.Е. Утилита восстановления алгоритмов работы машинного кода «AlgorithmRecover»: св-во о гос. рег. программы для ЭВМ № 2013618433 от 23 июля 2013 г.

19. Израилов К.Е. Архитектура системы для проведения генетической реинжиниринга программы с целью поиска разноуровневых уязвимостей // Вопросы кибербезопасности. 2025. № 1 (65). С. 108–116. DOI: 10.21681/2311-3456-2025-1-108-116.

20. Израилов К.Е. Проблемные вопросы генетической деэволюции представлений программы для поиска в них уязвимостей и рекомендации по их разрешению // Труды учебных заведений связи. 2025. Т. 11. № 1. С. 84–98. DOI: 10.31854/1813-324X-2025-11-1-84-98.

21. Израилов К.Е. Визуализация многопризнаковых уязвимостей программного кода с помощью метода главных компонент // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Сер. 1: Естественные и технические науки. 2020. № 1. С. 3–8.

22. Израилов К.Е. Система критериев оценки способов поиска уязвимостей и метрика понятности представления программного кода // Информатизация и связь. 2017. № 3. С. 111–118.

References

1. Izrailov K.E. Metodologiya revers-inzhiniringa mashinnogo koda. Chast' 1. Podgotovka ob"ekta issledovaniya // Trudy uchebnyh zavedenij svyazi. 2023. T. 9. № 5. S. 79–90. DOI: 10.31854/1813-324X-2023-9-5-79-90.

2. Izrailov K.E. Metodologiya revers-inzhiniringa mashinnogo koda. Chast' 2. Staticheskoe issledovanie // Trudy uchebnyh zavedenij svyazi. 2023. T. 9. № 6. S. 68–82. DOI: 10.31854/1813-324X-2023-9-6-68-82.

3. Izrailov K.E. Metodologiya revers-inzhiniringa mashinnogo koda. Chast' 3. Dinamicheskoe issledovanie i dokumentirovanie // Trudy uchebnyh zavedenij svyazi. 2024. T. 10. № 1. S. 86–96. DOI: 10.31854/1813-324X-2024-10-1-86-96.

4. Izrailov K.E. Modelirovanie programmy s uyazvimostyami s pozicii evolyucii ee predstavlenij. Chast' 1. Skhema zhiznennogo cikla // Trudy uchebnyh zavedenij svyazi. 2023. T. 9. № 1. S. 75–93. DOI: 10.31854/1813-324X-2023-9-1-75-93.

5. Izrailov K.E. Modelirovanie programmy s uyazvimostyami s pozicii evolyucii ee predstavlenij. Chast' 2. Analiticheskaya model' i eksperiment // Trudy uchebnyh zavedenij svyazi. 2023. T. 9. № 2. S. 95–111. DOI: 10.31854/1813-324X-2023-9-2-95-111.

6. Izrailov K.E. Sredstvo postroeniya zhiznennogo cikla programmy s potencial'nymi uyazvimostyami: sv-vo o reg. programmy dlya EVM № 2023664967 ot 11 iyulya 2023 g.

7. Izrailov K.E. Koncepciya geneticheskoj deevolyucii predstavlenij programmy. Chast' 1 // Voprosy kiberbezopasnosti. 2024. № 1 (59). S. 61–66. DOI: 10.21681/2311-3456-2024-1-61-66.

8. Izrailov K.E. Koncepciya geneticheskoj deevolyucii predstavlenij programmy. Chast' 2 // Voprosy kiberbezopasnosti. 2024. № 2 (60). S. 81–86. DOI: 10.21681/2311-3456-2024-2-81-86.

9. Izrailov K.E. Koncepciya geneticheskoj dekompilyacii mashinnogo koda telekommunikacionnyh ustrojstv // Trudy uchebnyh zavedenij svyazi. 2021. T. 7. № 4. S. 10–17. DOI:10.31854/1813-324X-2021-7-4-95-109.

10. Izrailov K.E. Primenenie geneticheskih algoritmov dlya dekompilyacii mashinnogo koda // Zashchita informacii. Insajd. 2020. № 3 (93). S. 24–30.

11. Izrailov K.E. Programmnoe sredstvo generacii ekzemplyarov iskhodnogo koda programmy soglasno formal'nomu sintaksisu yazyka programmirovaniya: sv-vo o reg. programmy dlya EVM (v pechati).

12. Izrailov K.E. Prognozirovanie razmera iskhodnogo koda binarnoj programmy v interesah ee intellektual'nogo revers-inzhiniringa // Voprosy kiberbezopasnosti. 2024. № 4 (62). S. 13–25. DOI: 10.21681/2311-3456-2024-4-13-25.

13. Izrailov K.E. Issledovanie raspredeleniya konstantnyh znachenij v iskhodnom kode programm na yazyke C // Trudy uchebnyh zavedenij svyazi. 2024. T. 10. № 5. C. 119–129. DOI: 10.31854/1813-324X-2024-10-5-118-128.

14. Izrailov K.E. Metodika ocenki effektivnosti sredstv algoritmizacii, ispol'zuemyh dlya poiska uyazvimostej // Informatizaciya i svyaz'. 2014. № 3. S. 39–42.

15. Izrailov K. GREMC: Genetic Reverse-Engineering of Machine Code to Search Vulnerabilities in Software for Industry 4.0. Predicting the Size of the Decompiling Source Code // The proceedings of International Russian Smart Industry Conference. Sochi, 2024. P. 622–628. DOI: 10.1109/SmartIndustryCon61328.2024.10515515.

16. Izrailov K.E. Programmnoe sredstvo postroeniya raspredeleniya konstantnyh znachenij v iskhodnom kode programm na yazyke C: sv-vo o reg. programmy dlya EVM (v pechati).

17. Izrailov K.E. Programmnoe sredstvo ocenki razmera iskhodnogo koda po ego mashinnomu kodu: sv-vo o reg. programmy dlya EVM (v pechati).

18. Izrailov K.E. Utilita vosstanovleniya algoritmov raboty mashinnogo koda «AlgorithmRecover»: sv-vo o gos. reg. programmy dlya EVM № 2013618433 ot 23 iyulya 2013 g.

19. Izrailov K.E. Arhitektura sistemy dlya provedeniya geneticheskoj reinzhiniringa programmy s cel'yu poiska raznourovnevyh uyazvimostej // Voprosy kiberbezopasnosti. 2025. № 1 (65). S. 108–116. DOI: 10.21681/2311-3456-2025-1-108-116.

20. Izrailov K.E. Problemnye voprosy geneticheskoj deevolyucii predstavlenij programmy dlya poiska v nih uyazvimostej i rekomendacii po ih razresheniyu // Trudy uchebnyh zavedenij svyazi. 2025. T. 11. № 1. C. 84–98. DOI: 10.31854/1813-324X-2025-11-1-84-98.

21. Izrailov K.E. Vizualizaciya mnogopriznakovyh uyazvimostej programmnogo koda s pomoshch'yu metoda glavnyh komponent // Vestnik Sankt-Peterburgskogo gosudarstvennogo universiteta tekhnologii i dizajna. Ser. 1: Estestvennye i tekhnicheskie nauki. 2020. № 1. S. 3–8.

22. Izrailov K.E. Sistema kriteriev ocenki sposobov poiska uyazvimostej i metrika ponyatnosti predstavleniya programmnogo koda // Informatizaciya i svyaz'. 2017. № 3. S. 111–118.

Информация о статье:
Статья поступила в редакцию: 20.01.2025; одобрена после рецензирования: 04.03.2025;
принята к публикации: 06.03.2025
Information about the article:
The article was submitted to the editorial office: 20.01.2025; approved after review: 04.03.2025;
accepted for publication: 06.03.2025

Сведения об авторах:
Израилов Константин Евгеньевич, старший научный сотрудник лаборатории проблем компьютерной безопасности Санкт-Петербургского Федерального исследовательского центра Российской академии наук (199178, Санкт-Петербург, В.О., 14-я линия, д. 39), кандидат технических наук, доцент, e-mail: konstantin.izrailov@mail.ru, https://orcid.org/0000-0002-9412-5693, SPIN-код: 5109-3499

Information about the authors:
Izrailov Konstantin E., senior researcher at the laboratory of computer security problems of the Saint-Petersburg Federal research center of the Russian academy of sciences (199178, Saint-Petersburg, V.O., 14th Line, 39), candidate of technical sciences, associate professor, e-mail: konstantin.izrailov@mail.ru, https://orcid.org/0000-0002-9412-5693, SPIN: 5109-3499