Алгоритм «E» превентивного управления профессиональной надежностью при индивидуальной и коллективной деятельности персонала БАК в условиях низкого уровня функциональной надежности персонала (от 1 до 37 баллов) и высокой сложности условий деятельности (от 64 до 100 баллов).

В рамках алгоритмов «D» и «E» предусмотрены мероприятия по мониторингу, прогнозированию и оптимизации надежности деятельности персонала как функциональной надежности персонала прогнозной динамики и превентивного управления профессиональной надежностью персонала БАК на основе анализа и оценки внешних и внутренних факторов.

Предложенные алгоритмы могут быть использованы для превентивного управления безопасностью сложных транспортных и производственных систем и комплексов.

DOI: 10.25728/iccss.2023.49.69.057

Савин Л.А.

Применение инструментов обратной разработки для выявления недекларированных возможностей программного обеспечения систем управления

Аннотация: В данной работе обсуждается одна из основных проблем реверс-инжиниринга – получение исходного кода программы из собранного решения. Определена важность решения данной задачи для выявления недекларированных возможностей программного обеспечения систем управления, проанализированы методы получения исходного кода и его конвертации на различные языки программирования.

Ключевые слова: компьютерная безопасность, обратная разработка, исходный код, угроза, системы управления, язык программирования

В наше время обратная разработка, или реверс-инжиниринг, имеет огромное значение в решении задач компьютерной безопасности, начиная со стандартной отладки программного обеспечения (ПО), и заканчивая выявлением серьезных угроз в

области информационных технологий. Как показывают исследования *Positive Technologies* [1], за год прирост по числу новых уязвимостей составил свыше 20%. Согласно этому же исследованию, злоумышленники чаще стали использовать вредоносное ПО для реализации атак. Доля успешно проведенных атак на 27% превышает тот же показатель в прошлом году [2]. В таких реалиях возможность быстро и точно оценить назначение программы и установить список ее возможностей, в том числе недекларированных, является одной из ключевых задач обратной разработки. Информационная безопасность сложных систем должна поддерживаться на высоком уровне, чтобы сохранять отказоустойчивость системы в целом. С одной стороны, не существует универсальных решений, способных охватить весь спектр этих задач. С другой стороны, в свободном доступе находится множество программных средств, с помощью которых специалист способен качественно провести анализ программного обеспечения и обеспечить бесперебойную работу автоматизированных систем управления.

Реверс-инжиниринг является универсальным методом для распознавания и анализа недекларированных возможностей ПО. Так как часто приходится работать с уже собранными в исполняемые файлы программами, без доступа к их исходному коду, инструменты обратной разработки представляют большую ценность. Они позволяют читать код приложения на языке ассемблера либо его интерпретацию на языке высокого уровня. Разумеется, это требует глубоких знаний в области языков программирования и, несомненно, самого языка ассемблера. Так как перевод дизассемблированного кода на код высокого уровня возможен не всегда, мы сталкиваемся с проблемой затратного с точки зрения продолжительности проведения анализа программ. Как следствие, ввиду возрастающего числа вредоносного ПО, эффективность работы по анализу кратно снижается. Стоит также отметить, что анализ программы на недокументированные возможности зачастую сложнее, чем просто анализ. Это связано со сложностью понимания логики программы сторонним человеком. Если в программу встраивается вредоносный код, его создатель, зачастую, старается «спрятать» его от чужих глаз. Обнаружить

ошибку, допущенную другим работником во время проектирования системы – также не самая простая задача.

В случае с ПО систем управления мы имеем дело с критически важной инфраструктурой. Это может быть как небольшое частное предприятие, так и промышленный комплекс. Скорость и качество анализа ПО являются главными приоритетами в предотвращении потенциальных для системы угроз. Недекларированные возможности в ПО возникают как по внешним причинам (таргетированная атака, несанкционированный доступ), так и по внутренним, связанным с человеческим фактором: халатность сотрудников, низкоквалифицированные кадры, «обиженные» сотрудники, баги в программном обеспечении, недостаточно качественное построение логики и т.п. [3]. Независимо от причины возникновения, все они могут нанести вред системе как в виде незначительных изменений, затрагивающих частных лиц, так и серьезных последствий, вплоть до массовых утечек информации и неработоспособности целых организаций [4]. Подобные угрозы необходимо выявлять заранее, чтобы не допустить глобальных проблем.

Для создания безопасного ПО, в первую очередь, требуется безопасный язык программирования. Для тестирования выберем три языка: Python, C#, C++. Выполненный в ходе исследования процесс формализован на схеме на рисунке 1. Вначале на выбранных языках программирования были реализованы программы с одинаковым функционалом. Для получившихся проектов была произведена сборка в исполняемые файлы. После этого была предпринята попытка получить доступ к исходному коду файлов, используя различный инструментарий реверс-инжиниринга. Была проведена оценка применимости конкретного инструмента к конкретному файлу с точки зрения читаемости дизассемблированного кода.

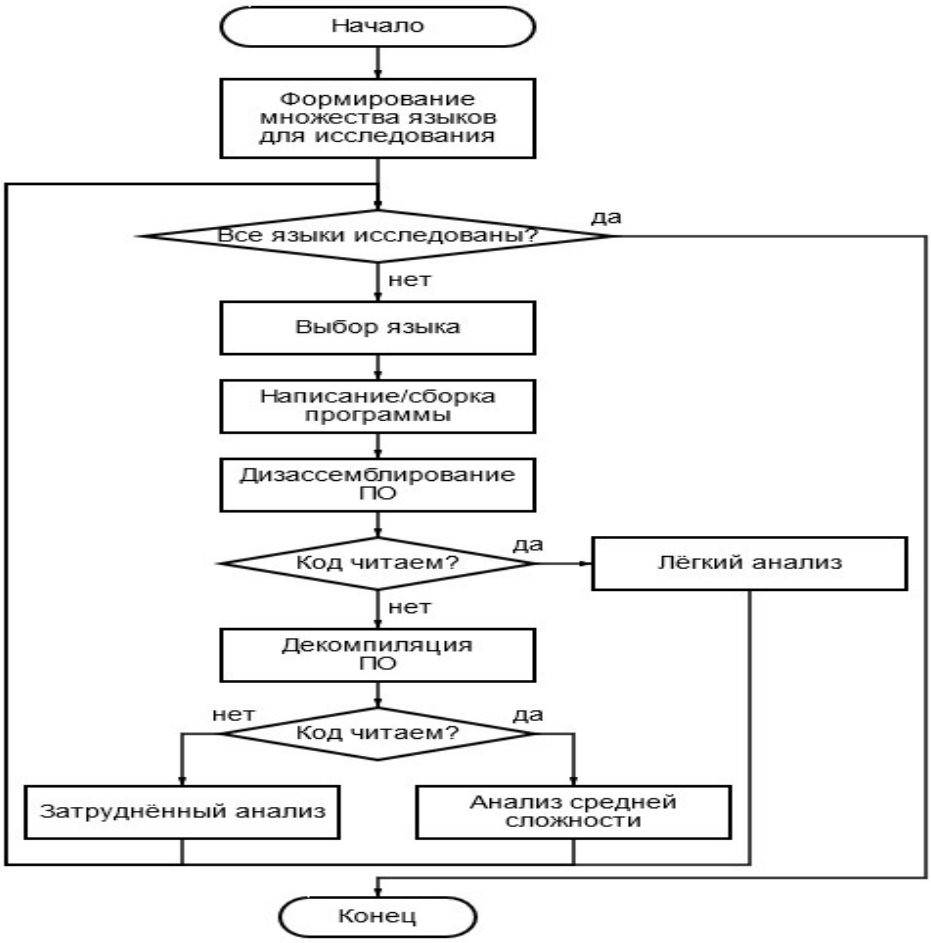

Рисунок 1 – Схема алгоритма выбора языка

Работа по обратной разработке была проведена с помощью следующих прикладных программ:

– для кода на Python использовались PyInstaller Extractor и модуль uncompyle6;

– для кода на C# – программы .NET Reflector и ILSpy;

– для кода на C++ – IDA Pro и Radare2.

В результате работы были проверены некоторые утверждения. Например, так как код на языке Python интерпретируемый, а не компилируемый, сборка кода в защищенный исполняемый файл не предусмотрена стандартными средствами. В данном случае, процесс обратной разработки позволит получить весь исходный код программы.

При работе с программой, написанной на языке C#, значительных особенностей, кроме выбора соответствующего инструмента дизассемблирования и получения результата в виде

псевдокода, замечено не было. Следует отметить, что по синтаксису языка псевдокод очень похож на язык Си, что делает его удобным для чтения и анализа.

Работа с приложением, реализованным на языке C++, наиболее сложна. Это связано с механизмом компиляции языка. Код программ на языке C++ напрямую транслируется в ассемблерный код, тогда как компилятор языка C#, например, сначала переводит свой код в промежуточный код, называемый IL кодом. Именно его и пытаются воспроизвести многие декомпиляторы. В программе, написанной на языке C++, при использовании даже стандартных библиотек код становится запутанным, появляются неочевидные на первый взгляд пути исполнения программы и переходы. С другой стороны, с достаточным уровнем знаний возможно обойти любую защиту подобного рода и выше, что показано авторами статьи [5].

При решении задач реверс-инжиниринга следует учитывать возможное наличие обфускации исходного кода. Программа с обфускацией должна быть предварительно деобфусцирована с помощью специального ПО. Существует множество таких программ [6].

Следует отдельно упомянуть конвертацию кода с одного языка программирования на другой. Написанное программное обеспечение с течением времени устаревает, в нем обнаруживаются изъяны логики, которые могут быть использованы с целью обхода защиты. В этом случае конвертация кода может заметно помочь обезопасить программу. С одной стороны, современные методы трансляции программного кода на другой язык подразумевают использование нейронных сетей, что позволяет исключить человеческий фактор. С другой – нейросети в текущем виде «слабы», зачастую дают неверный код, который может даже не выполниться на устройстве. Однако развитие технологии в этой сфере даст преимущество в виде написания кода без участия человека. Это позволит сфокусироваться на реверс-инжиниринге программ с целью выявления недокументированных возможностей, а не исправления ошибок логики программ.

По результатам тестов было выявлено (таблица 1), что из тройки языков (Python, C#, C++) предпочтителен для написания безопасных программ последний. Однако использование этого языка, равно, как и других, не дает гарантии на защищенность

программы от профессионала в области обратной разработки. Это означает, что любая программа, теоретически, может быть детально проанализирована на недекларированные возможности, основные ветвления логики. При этом следует учитывать, что написание длинного кода с плохо выстроенной архитектурой часто помогает защитить код лучше, чем многие программные средства, созданные для этих целей. Это обусловлено человеческим фактором, так как именно человек проводит обратную разработку и анализирует логику программного обеспечения.

Таблица 1 – Сравнительная характеристика языков для целей обратной разработки

Критерии сравнения	**Язык исходного кода программы**		
	Python	**C#**	**C++**
Набор ПО для обратной разработки	PyInstaller Extractor, uncompyle6	.NET Reflector, ILSpy	IDA Pro, Radare2
Сложность анализа	Легкий	Средней сложности	Затрудненный
Возможность получить исходный код	Да	Частично	Нет
Уровень необходимых навыков для обратной разработки	Знание языка Python, базовые умения работы с командной строкой	Понимание логики промежуточного кода, умение работы с разнообразным ПО для обратной разработки. Навыки определения обфускации программного кода	Глубокие знания устройства языка ассемблера, логики языка C++, навыки работы с hex редакторами, разнообразным ПО для обратной разработки
Пригодность языка для использования в ПО систем управления	Нет	Пригоден в некоторых случаях	Рекомендуется к применению

Подводя итог сказанному, следует отметить, что развитие инструментов обратной разработки и прикладного ПО может способствовать повышению надежности и отказоустойчивости ПО систем управления. С другой стороны, наличие качественного ПО не исключает человека из данной сферы, что в свою очередь означает зависимость безопасности системы от навыков конкретного человека или группы людей. Основная задача в этом случае – предоставить качественные инструменты для решения задач обратной разработки, и, как следствие, повысить безопасность системы в целом.

Представленный в статье материал основан на анализе результатов, полученных обучающимися РУТ (МИИТ) в ходе выполнения лабораторных работ по дисциплине «Технологии реверс-инжиниринга» и представляет собой продолжение цикла публикаций, посвященных опыту подготовки специалистов по компьютерной безопасности в РУТ (МИИТ) [7].

Литература:

1. Блог компании Positive Technologies. Актуальные киберугрозы: I квартал 2023. – URL: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2023-q1 (дата обращения 23.10.23).

2. *Савина А.* Киберпреступления в России: тенденции 2023 года. – URL: https://cloudnetworks.ru/analitika/kiberprestupleniya-v-rossii-tendentsii-2023-goda (дата обращения 23.10.23).

3. *Кравчук А.Ю., Котова Н.А., Аничкин И.И.* Современные подходы к обеспечению информационной безопасности автоматизированных систем управления технологическими процессами // Научно-аналитический журнал «Инновации и Инвестиции». – 2022. – №3. – С. 191-196. – URL: http://www.innovazia.ru/upload/iblock/ed4/mzn7vhrf4336ffv4gk76nd3aw9xar3it/%E2%84%963%202022.pdf (дата обращения 23.10.23).

4. Блог компании Positive Technologies. Актуальные киберугрозы: II квартал 2023 года. – URL: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2023-q2 (дата обращения 24.10.23)

5. *Аникин И.В., Исяндавлетова Я.М.* Реверсивный анализ вредоносного программного обеспечения Racoon Stealer // Научный журнал «Инженерный вестник Дона». Сетевое издание. – 2023. – №4. – URL: http://ivdon.ru/ru/magazine/archive/n4y2023/8346 (дата обращения 25.10.23).

6. Веб-сервис для хостинга IT-проектов GitHub. – URL: https://github.com/NotPrab/.NET-Deobfuscator (дата обращения 24.10.23).

7. *Логинова Л.Н., Сидоренко В.Г.* Направления совершенствования подготовки специалистов в области компьютерной безопасности // Информатизация образования и науки. – 2023. – №2(58). – С. 64-71.

DOI: 10.25728/iccss.2023.50.79.058

Хабибулин Р.Ш.

Задача управления пожарными рисками на объектах топливно-энергетического комплекса на основе методов оптимизации и искусственного интеллекта

Аннотация: Сформулирована общая постановка научной задачи разработки методов, моделей, алгоритмов и специализированного программного и информационно-аналитического обеспечения поддержки принятия решений при управлении пожарными рисками. Показаны некоторые результаты исследований по созданию практико-ориентированных технологий управления в рассматриваемой предметной области.

Ключевые слова: пожарные риски, управление пожарными рисками, системы поддержки принятия решений, топливно-энергетический комплекс

Введение

В целом, проблематика управления пожарной безопасностью производственных объектов с использованием инструментария пожарных рисков и специализированного программного обеспечения рассмотрена во многих работах, например [1-3]. В работе [4] автором сформулирован общий комплексный подход к