

## СОДЕРЖАНИЕ

<b>Введение.....</b>	<b>6</b>
<b>Расчётно-пояснительная записка.....</b>	<b>7</b>
<b>1. Техническое задание.....</b>	<b>7</b>
1.1 Общие сведения.....	7
1.1.1 Полное наименование системы и её условное обозначение .....	7
1.1.2 Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты.....	7
1.1.3 Перечень документов, на основании которых создаётся система, кем и когда утверждены эти документы .....	7
1.1.4 Плановые сроки начала и окончания работы по созданию системы.....	7
1.2 Назначение и цели создания системы.....	7
1.2.1 Назначение системы .....	7
1.2.2 Цели создания системы .....	8
1.3 Характеристика объекта автоматизации.....	8
1.4 Требования к системе.....	8
1.4.1 Требования к системе в целом.....	8
1.4.1.1 Требования к структуре и функционированию системы.....	8
1.4.1.2 Требования к численности и квалификации персонала и режимам его работы .....	9
1.4.1.3 Показатели назначения.....	9
1.4.1.4 Требования к надёжности.....	10
1.4.1.5 Требования к безопасности.....	11
1.4.1.6 Требования к эргономике и технической эстетике.....	11
1.4.1.7 Требования к транспортабельности подвижных АС.....	12
1.4.1.8 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы.....	12

1.4.1.9 Требования к защите информации от несанкционированного доступа.....	12
1.4.1.10 Требования по сохранности информации при авариях.....	13
1.4.1.11. Требования к средствам защиты от влияния внешних воздействий.....	13
1.4.1.12 Требования к патентной чистоте.....	13
1.4.1.13 Требования к стандартизации и унификации.....	13
1.4.1.14 Дополнительные требования.....	14
1.4.2 Требования к функциям (задачам), выполняемым системой.....	14
1.4.2.1 Требования к подсистеме. Перечень функций, задач или их комплексов.....	14
1.4.2.2 Требования к качеству реализации каждой функции (задачи или комплекса задач), к форме предоставления выходной информации, характеристики необходимой точности и времени выполнения требований одновременности выполнения группы функций, достоверности выдачи результатов.....	15
1.4.3 Требования к видам обеспечения.....	15
1.4.3.1 Требования к математическому обеспечению.....	15
1.4.3.2 Требования к информационному обеспечению.....	15
1.4.3.3 Требования к лингвистическому обеспечению.....	16
1.4.3.4 Требования к программному обеспечению.....	16
1.4.3.5 Требования к техническому обеспечению.....	16
1.4.3.6 Требования к метрологическому обеспечению.....	17
1.4.3.7 Требования к организационному обеспечению.....	17
1.4.3.8 Требования к методическому обеспечению.....	17
1.4.3.9 Требования к другим видам обеспечения системы.....	17
1.5 Состав и содержание работ по созданию (развитию) системы.....	17
1.6 Порядок контроля и приёмки системы.....	17

1.6.1 Виды, состав, объем и методы испытаний системы и составных частей.....	17
1.6.2 Общие требования к приёмке работ по стадиям.....	18
1.6.3 Статус приёмочной комиссии.....	18
1.7 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие.....	18
1.8 Требования к документированию.....	18
<b>2. Научно-исследовательская часть.....</b>	<b>19</b>
2.1. Постановка задачи проектирования.....	19
2.2. Описание предметной области.....	19
2.3. Анализ аналогов и прототипов.....	26
2.4. Перечень задач, подлежащих решению в процессе разработки.....	27
2.5. Обоснование выбора инструмента и платформы для разработки.....	28
<b>3. Проектно-конструкторская часть.....</b>	<b>28</b>
3.1. Разработка структуры приложения.....	28
3.2. Разработка алгоритмов обработки информации.....	29
3.3. Разработка архитектуры приложения.....	46
3.4. Реализация готового приложения.....	47
3.5. Разработка интерфейса взаимодействия пользователя с системой.....	48
<b>4. Проектно-технологическая часть.....</b>	<b>49</b>
4.1 Тестирование и отладка макета рабочей программы.....	49
4.2 Разработка руководства пользователя и руководства программиста.....	52
4.3 Экспериментальные данные тестирования.....	55
<b>Заключение.....</b>	<b>57</b>
<b>Список используемых источников.....</b>	<b>58</b>
<b>Приложения.....</b>	<b>60</b>
Перечень принятых сокращений.....	60

## **Введение**

Тема, касающаяся внедрения кода в сторонние процессы, актуальна в современном мире. Со времён появления первых компьютерных программ люди, обладающие навыками в области программирования, всячески старались улучшить существующие программы путём добавления новых графических элементов, цветовых тем, функционала в эти программы. Однако почти всегда они сталкивались с проблемой отсутствия исходного кода целевой программы. По этой причине народные умельцы научились внедрять свой код в стороннее приложение с целью изменение его работы. Появились самые разнообразные техники внедрения кода, начиная от простого написания кода на одном из популярных языков программирования, заканчивая сложным написанием машинных инструкций и их дальнейшим запуском в памяти другой программы. Существующие техники внедрения кода используются сейчас по сей день и применяются в различных областях.

Во всяком случае программистам необходимо прибегать к использованию функций, предоставляемых операционной системой, для реализации намеченной цели. Такие функции называются API-функциями и служат для предоставления пользователю удобного интерфейса взаимодействия с операционной системой.

Цель курсовой работы: Изучить функции взаимодействия с операционной системой на примере реализации приложения для внедрения кода в сторонние процессы, используя различные техники внедрения кода.

Задачи:

1. Разработать техническое задание для приложения.
2. Провести исследования предметной области.
3. Разработать приложение реализующее внедрение кода в сторонние процессы, используя API-функции операционной системы.
4. На основе созданного приложения провести исследования различных техник внедрения кода в сторонние процессы и сравнить их между собой.

## **1 Техническое задание**

### **1.1 Общие сведения**

#### **1.1.1 Полное наименование системы и её условное обозначение**

Автоматизированная система внедрения исполняемого кода в сторонние процессы АС «Pig».

#### **1.1.2 Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты**

Заказчик: Калужский филиал Московского государственного технического университета им. Н. Э. Баумана (КФ МГТУ им. Баумана).

Исполнитель (разработчик): Студент группы ИУК5-42Б КФ МГТУ им. Н. Э. Баумана Хохлов В. М.

#### **1.1.3 Перечень документов, на основании которых создаётся система, кем и когда утверждены эти документы**

АС создаётся на основании данного технического задания. Иных документов, являющихся основанием разработки АС не предусмотрено.

#### **1.1.4 Плановые сроки начала и окончания работы по созданию системы**

Плановый срок начала работы — 1 февраля 2022

Плановый срок окончания работы — 25 мая 2022

### **1.2 Назначение и цели создания системы**

#### **1.2.1 Назначение системы**

АС «Pig» предназначена для автоматизации внедрения кода в сторонний процесс, в частности исполнения следующих процессов:

1. Производство внедрения кода в сторонний процесс;
2. Информирование пользователя о результатах работы.

### **1.2.2 Цели создания системы**

Основными целями создания АС «Pig» являются:

1. Получение навыков работы с API функционалом, предоставляемым операционной системой Windows;
2. Получение знаний, касающихся принципов работы алгоритмов внедрения кода в сторонние процессы, используя различные техники внедрения.
3. Сравнение техник внедрения кода в сторонние процессы с целью формирования отчёта в письменном и графическом виде.

### **1.3 Характеристика объекта автоматизации**

Объект автоматизации — алгоритмы внедрения кода в сторонние процессы. Внедрение кода представляет собой запись специальных данных в процессы, запущенные в операционной системе. Внедрение может реализовываться путём использования различных техник внедрения. Подробнее каждая техника должна быть освещена в научно-исследовательской части курсовой работы.

Объект автоматизации должен эксплуатироваться в нормальных условиях эксплуатации.

### **1.4 Требования к системе**

#### **1.4.1 Требования к системе в целом**

##### **1.4.1.1 Требования к структуре и функционированию системы**

Автоматизированная система должна состоять из следующих подсистем:

- Подсистема GUI
- Подсистемы ВКСР

Подсистема GUI предоставляет пользователю графический интерфейс, необходимый для выбора режимов работы АС, выбора внедряемых файлов. Подсистема графического интерфейса пользователя предоставляет собой окно,

содержащее в себе элементы управления необходимые для реализации нормального функционирования АС.

Подсистемы ВКСП предназначена для предоставления возможности пользователю внедрять код в сторонние процессы, запущенные в операционной системе.

Система функционирует в нормальном режиме, являющимся единственным режимом работы АС. Иных режимов для функционирования системы не предусмотрено. При нормальном режиме работы в АС не возникает ошибок различного рода, выполнение алгоритмов внедрения кода в сторонний процесс выполняется успешно.

Система создаётся преимущественно с целью изучения API функций операционной системы. Перспективы дальнейшего развития АС и её модернизации отсутствуют.

#### **1.4.1.2 Требования к численности и квалификации персонала системы и режиму его работы**

АС создаётся преимущественно для одного пользователя. Специальных требований к численности персонала (пользователей) не предъявляется.

К квалификации персонала, эксплуатирующего АС предъявляются следующее требование: умение работать с компьютером на уровне среднего пользователя. Минимальное понимание предметной области.

Требований к режиму работы персонала не предъявляется.

#### **1.4.1.3 Показатели назначения**

АС может эксплуатироваться при любых изменениях процессов и методов управления. При отклонении параметров объекта управления система должна работать в штатном режиме.

АС предназначена преимущественно для исследования работы API функций операционной системы и не предусматривает дальнейшего развития и модернизации.

АС должна сохранять своё целевое назначение на протяжении всего цикла работы, в независимости от действия пользователя.

#### **1.4.1.4 Требования к надёжности**

Система должна отвечать следующим требованиям надёжности:

1. Отказоустойчивость. Система должна оставаться работоспособной в случае возникновения ошибки в процессе работы одного из модулей, предназначенного для внедрения кода.

2. Защита от некорректного пользовательского ввода. Система не должна потерять работоспособность в случае получения некорректных данных со стороны пользователя.

3. Достаточный для нормального функционирования АС уровень надёжности должен достигаться путём проверок пользовательского ввода на корректность, соблюдения правил эксплуатации АС и предварительного ознакомления пользователя с работой АС.

К надёжности АС предъявляются следующие требования:

1. В качестве аппаратной платформы (операционной системы) должны использоваться средства с повышенной надёжностью;

2. При возникновении сбоев в работе модулей внедрения кода в сторонний процесс требуется проинформировать пользователя о соответствующей ошибке.

3. Надёжность программного обеспечения подсистем должна обеспечиваться за счёт следующих мероприятий:

1) Ознакомление пользователя с работой АС;

2) Соблюдение рекомендаций к эксплуатации АС;

3) Проведение комплекса мероприятий отладки, поиска и исправления ошибок на этапе тестирования и отладки АС.

Требований к методам оценки и контроля показателей надёжности не предъявляется.



#### **1.4.1.5 Требования к безопасности**

АС не работает с внешними техническими средствами, требующие соблюдения специальной техники безопасности при работе, установке, наладке, техническому осмотру и ремонту. Требования безопасности к АС не предъявляется.

#### **1.4.1.6 Требования к эргономике и технической эстетике**

Взаимодействие пользователей с прикладным программным обеспечением, входящим в состав системы, должно осуществляться посредством визуального графического интерфейса (GUI). Интерфейс системы должен быть понятным и удобным, не должен быть перегружен графическими элементами и должен обеспечивать быстрое отображение экранных форм.

Интерфейс должен быть рассчитан на преимущественное использование манипулятора типа «мышь», клавиатурный режим ввода должен использоваться главным образом при заполнении и/или редактировании текстовых и числовых полей экранных форм.

Система должна обеспечивать корректную обработку аварийных ситуаций, вызванных неверными действиями пользователей, неверным форматом или недопустимыми значениями входных данных. В указанных случаях Система должна выдавать пользователю соответствующие сообщения, после чего возвращаться в рабочее состояние, предшествовавшее неверной (недопустимой) команде или некорректному вводу данных.

Экранные формы должны разрабатываться с учётом требований унификации: все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации; для обозначения сходных операций должны использоваться сходные графические значки, кнопки и другие управляющие (навигационные) элементы.

#### **1.4.1.7 Требования к транспортабельности для подвижных АС**

Специальных условий для транспортабельности АС не предъявляется: система может транспортироваться на малогабаритных носителях информации (флешка, жёсткий диск) или передаваться по сети Интернет.

#### **1.4.1.8 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы**

АС не обладает дополнительными техническими средствами, требований к использованию дополнительных технических средств АС не предъявляются.

Допустимая площадь для размещения персонала — площадь, необходимая для размещения одного человека и персонального компьютера (ноутбука). Для корректной работы АС необходимо бесперебойное снабжение электроэнергией устройства, хранящего и выполняющего алгоритмы АС.

АС может управлять один человек, требований к количеству и квалификации обслуживающего персонала, а также режиму его работы не предъявляются.

АС не снабжена комплексом запасных изделий и приборов. Требования к составу, размещению и условиям хранения не предъявляются.

Требования к регламенту обслуживания АС не предъявляются, однако для правильного функционирования АС рекомендуется следовать инструкции, прилагаемой к АС описанной в конструкторской части курсовой работы.

#### **1.4.1.9 Требования к защите информации от несанкционированного доступа**

АС не хранит данные, составляющие коммерческую или государственную тайну, персональные данные пользователей и иные данные, доступ к которым должен быть ограничен или запрещён лицам, не имеющим прав доступа к этим данным. Требования к защите информации от несанкционированного доступа к данным АС не предъявляются.

#### **1.4.1.10 Требования по сохранности информации при авариях**

АС во время выполнения не будет хранить в оперативной памяти важные сведения, требующиеся для корректного функционирования АС в будущем. Требования к сохранности информации при авариях не предъявляются.

#### **1.4.1.11 Требования к средствам защиты от влияния внешних воздействий**

АС не должна обладать средствами, механизмами и иными предметами, требующими радиоэлектронной защиты. Требования радиоэлектронной защите не предъявляются.

АС предназначена для работы на стационарных компьютерах или ноутбуках, находящихся в условиях отсутствия внешних физических воздействий. Специальных требований к защите информации от влияния внешних воздействий и среде применения не предъявляется.

#### **1.4.1.12 Требования к патентной чистоте**

Установка системы в целом, как и установка отдельных частей системы не должна предъявлять требований к покупке лицензий на программное обеспечение сторонних производителей.

#### **1.4.1.13 Требования по стандартизации и унификации**

Единообразный подход к решению однотипных задач должен достигаться: единым программно-техническим способом реализации одинаковых функций системы, унификацией компонентов математического, информационного, лингвистического и программного обеспечения, унификацией компонентов технического обеспечения. В графических модулях необходимо использовать единообразные элементы управления и цветовую схему.

#### **1.4.1.14 Дополнительные требования**

Дополнительные требования к АС могут быть предъявлены на этапе программной реализации АС. На этапе «формирования требований к АС» дополнительные требования не предъявляются.

#### **1.4.2 Требования к функциям (задачам), выполняемым системой**

##### **1.4.2.1 Требования к подсистеме. Перечень функций, задач или их комплексов.**

АС должна содержать следующие модули:

1. Модули ВКСП
2. Модуль GUI

Задача модулей ВКСП — внедрить код, хранящийся на жёстком диске или ином носителе, в указанный пользователем процесс. Модули ВКСП должны реализовывать следующие функции:

1. Открывать процесс по полученному идентификатору;
2. Открывать файлы, хранящиеся на жёстком диске;
3. Внедрять код в сторонние процессы;
4. Производить запуск внедрённого кода на исполнение.

Задача модуля GUI — предоставление пользователю понятного интерфейса для реализации всего функционала АС. Модуль графического интерфейса пользователя должен предоставлять следующий функционал:

1. Предоставление интерфейса диалоговых окон для выбора файлов;
2. Считывание данных полей ввода данных;
3. Обработка нажатия кнопок;
4. Вывод предупреждений, ошибок работы модулей ВКСП;
5. Формирование команд запуска модулей ВКСП;
6. Запуск модулей ВКСП с указанием параметров;
7. Предоставление отчётов о результатах внедрения.

#### **1.4.2.2 Требования к качеству реализации каждой функции (задачи или комплекса задач), к форме представления выходной информации, характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций, достоверности выдачи результатов**

Функции должны полностью выполнять поставленные задачи. После выполнения каждая функция должна вернуть результат своей работы. Функции не должны аварийно завершать работу системы АС. В случае возникновения ошибки, программа должна корректно завершиться с указанием причины завершения.

#### **1.4.3 Требования к видам обеспечения**

##### **1.4.3.1 Требования к математическому обеспечению**

Требования не предъявляются к математическому обеспечению.

##### **1.4.3.2 Требования к информационному обеспечению**

АС должна состоять из модулей, размещаемых на жёстком диске компьютера. АС не предусматривает хранение данных в базах данных или на файловом сервере.

Файл с результатами работы модулей внедрения кода в сторонний процесс может находиться в любом месте на жёстком диске. Специальные файлы, содержащие внедряемый код также должны находиться на диске.

Подсистема GUI должна использовать кодировку Unicode.

Сбор данных в АС должны происходить путём взаимодействия пользователя с подсистемой GUI. Пользователь последовательно вводит данные, необходимые для дальнейшего использования АС. По нажатии специальной кнопки, данные структурируются, проверяется их корректность. Данные, введённые пользователем, выступают в роли параметров для модулей внедрения кода в сторонние процессы.

Передача данных из модуля GUI в модули внедрения кода представляет собой запуск модулей внедрения кода с параметрами, предоставленными пользователем в модуле GUI. По завершение работы модуль ВКСР произведёт запись данных в файл. Модуль GUI, получив данные из файла, должен вывести данные на экран для информирования пользователя о результатах внедрения.

Информационных обмен должен происходить между компонентами внедрения кода и GUI посредством взаимодействия данных компонентов с общим файлом. Данные, записанные в файл модулями внедрения кода используются для отображения их модулем GUI.

#### **1.4.3.3 Требования к лингвистическому обеспечению**

Для реализации АС должен применяться язык С. Для написания специальных файлов, предназначенных для внедрения в сторонний процесс, допускается использование языка ассемблера.

#### **1.4.3.4 Требования к программному обеспечению**

Система должна быть снабжена всеми необходимыми библиотеками, предоставляющим доступ ко всем функция взаимодействия с операционной системой. Базовой программной платформой должна служить операционная система MS Windows. При проектировании системы необходимо максимально эффективным образом использовать все предоставляемые операционной системой функции.

#### **1.4.3.5 Требования к техническому обеспечению**

Основным техническим средством, носителем АС и средством запуска АС должен являться персональный компьютер (ноутбук). Для корректной работы АС техническое средство должно работать на базе операционной системы Windows. Требований к наличию иных комплектующих изделий для использования совместно с АС не предъявляется.

#### **1.4.3.6 Требования к метрологическому обеспечению**

Требования к метрологическому обеспечению не предъявляются.

#### **1.4.3.7 Требования к организационному обеспечению**

Требования к организационному обеспечению не предъявляются.

#### **1.4.3.8 Требования к методическому обеспечению**

АС должна быть снабжена необходимым руководством к использованию системы.

#### **1.4.3.9 Требования к другим видам обеспечения системы**

Требования не предъявляются.

### **1.5 Состав и содержание работ по созданию (развитию) системы**

1. Реализация модулей внедрения кода. Экспертиза: согласование правильности работоспособности модулей с заказчиком. (Срок 4 неделя)

2. Разработка прототипа интерфейса. Экспертиза: согласование прототипа с заказчиком. (Срок 6 неделя)

3. Разработка физической и логической схемы программы. Экспертиза: демонстрация проделанной работы заказчику. (Срок 8 неделя).

4. Разработка макета программы. Экспертиза: демонстрация проделанной работы заказчику. (Срок 10 неделя).

5. Отладка и устранение ошибок программы. Экспертиза: демонстрация готовой к вводу в эксплуатации АС. (Срок 14 неделя)

### **1.6 Порядок контроля и приёмки системы**

#### **1.6.1 Виды, состав, объем и методы испытаний системы и её составных частей**

Составные части АС должны быть протестированы по окончании стадии разработки программ. Тестирование системы должно включать обнаружение ошибок в ходе выполнения программы, ошибок в результате некорректных

действий пользователя, неточностей в работе модулей GUI. При добавлении нового функционала предыдущие тесты должны сохранить работоспособность.

### **1.6.2 Общие требования к приёмке работ по стадиям**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой. После демонстрации работоспособности АС на каждой стадии разработки происходит согласование текущего функционала, после чего разработка переходит на следующую стадию.

### **1.6.3 Статус приёмочной комиссии**

Приёмочная комиссия, организованная КФ МГТУ им. Баумана, осуществляет приёмку работы.

## **1.7 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие**

Специальных мероприятий для подготовки объекта автоматизации к вводу системы в действие не предусматривается. Для запуска и использования АС её необходимо установить на компьютер с установленной операционной системой Windows.

## **1.8 Требования к документированию**

Требуется предоставить:

1. Техническое задание в соответствии с ГОСТ 34.602-89
2. Расчётно-пояснительную записку, включающую исследовательскую часть, проектно-конструкторскую часть и проектно-технологическую часть. Расчётно-пояснительная записка выполняется с учётом требований, предусмотренных ГОСТ 7.32-2001 и 2.105-95.



## ТЕХНИЧЕСКОЕ ЗАДАНИЕ СОСТАВИЛИ

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата

## ТЕХНИЧЕСКОЕ ЗАДАНИЕ СОГЛАСОВАНО

Наименование организации, предприятия	Должность исполнителя	Фамилия, имя, отчество	Подпись	Дата

## 2 Научно-исследовательская часть

### 2.1 Постановка задачи проектирования

Для успешного проектирования автоматизированной системы необходимо выполнить ряд задач:

1. Реализовать графический интерфейс программы.
2. Реализовать модули для внедрения кода в сторонние процессы.
3. Разработать модели взаимодействия пользователя с программой.
4. Разработать модель взаимодействия компонентов программы между собой в процессе выполнения.

### 2.2 Описание предметной области.

**Характеристика внедрения кода в операционных системах.**

Операционная система (ОС) является системным программным

обеспечением, благодаря которому приводится в действие технические средства компьютера. Это программное обеспечение координирующее работу ЭВМ и производящее управление другими программными модулями посредством скоординированной последовательности операций.

Важным элементом успешного решения поставленных задач, наряду с обязательным использованием функциональных комплектующих персональных компьютерных устройств, является применение высокопроизводительной операционной системы. Наиболее полно удовлетворяет разнообразные потребности пользователей популярнейший продукт от корпорации «Microsoft» под названием «Операционная система Windows». Например, система имеет высокий уровень общей производительности, обладает способностью выполнять одновременно множество высоко затратных действия без задержек и сбоев, способна функционировать на устройствах различных платформ и конфигураций, располагает низким процентом системных отказов, выполнена в удачном дружественном интерфейсе, интуитивно понятном для пользователей и поддерживает установку множества дополнительных приложений от сторонних разработчиков. Дополнительно, «Windows» позволяет приложениям внедрять собственный код в системные настройки и процессы операционной системы, наделяя их расширенными правами для достижения максимального удобства пользователей.

Наиболее распространенным является процесс внедрения части собственного кода приложений в работающий процесс с целью изменить его поведение. Такая техника может использоваться как для расширения добросовестных возможностей приложений, так и быть направленной на причинение скрытого вреда. Но в любом случае, каковы бы ни были первоначальные причины, вызвавшие внедрение кода, такой вид воздействия на закрытые элементы управления системы может привести к нежелательным или губительным последствиям.

При запуске любой программы в системе создаётся один или несколько процессов запущенной программы, при этом исполняемый файл загружается в оперативную память компьютера определенным образом, формируется внутренняя структура организации памяти процесса, запускается выполнение первой инструкции программы.

В общем случае все исполняемые файлы имеют общую структуру организации памяти, в состав которой входят следующие регионы: сигнатуры DOS и PE-заголовков, секции данных, секции кода, стеки, управляемые кучи, таблицы перемещений, таблицы импорта/экспорта dll, незанятое пространство.

Каждый регион имеет свой определённый набор правил доступности. Существует несколько специальных правил доступности: регион памяти может быть доступен для записи, чтения и исполнения. Данные правила доступности могут сочетаться между собой.

В операционной системе предусмотрены функции выделения памяти под регион. Функция может работать как с процессом, в котором она вызывается, так и с любым другим сторонним процессом.

Существуют несколько способов внедрения кода в адресное пространство стороннего процесса, популярными являются: Reflective Dll Injection, Dll Injection, Code Injection.

Внедрение кода используется приложениями для расширения собственных функциональных возможностей за счёт использования программных модулей операционной системы «Windows». Но, к сожалению, не только добросовестные приложения осуществляют внедрение кода. Большинство видов вредоносного программного обеспечения также используют такой метод как один из вариантов заражения системы и нанесения вреда устройству или данным пользователей. Вот лишь некоторые примеры внедрения стороннего кода различными приложениями:

1. Антивирусное программное обеспечение часто прописывает собственный код элементов безопасности в веб-браузеры, установленные на

устройствах пользователей. Благодаря внедрению кода, антивирус производит потоковый мониторинг сетевых соединений и диагностику шлюзовых подключений, проверяет входящие и исходящие пакеты данных, а также блокирует содержимое опасных и ненадёжных веб-сайтов, существенно снижая риск возможного заражения.

2. Вредоносные приложения могут добавлять зловерный программный код в веб-браузер для сбора различной информации о деятельности пользователей. Например, вести учет посещённых веб-сайтов, отслеживать и фиксировать предпочтения пользователей, похищать защищённую информацию, такую как пароли, номера кредитных карт и банковские коды безопасности, изменять настройки вашего веб-браузера, скрытно производить загрузку зловерных программ и т.д.

3. Сторонние приложения, позволяющие устанавливать пользовательские темы и визуальные стили в операционной системе «Windows» (например, «Stardock WindowBlinds»), вводят собственный код для изменения способа отображения системных графических элементов. Такой вариант даёт возможность создавать собственные или использовать набор готовых тем рабочего стола, и применять выбранные стили ко всем установленным компонентам.

4. Внешние приложения организации пространства (например, «Stardock's Fences») вносят изменения в системный код, что позволяет им очищать или структурировать по тематическим блокам ярлыки файлов и программ для уменьшения засорённости рабочего стола «Windows», добавлять новые функции и создавать автоматические правила для последующих файлов.

5. Программная платформа сценариев, такая как «AutoHotkey», с помощью добавления небольшого кода, предоставляет пользователям возможность создавать сценарии, которые запускаются в фоновом режиме с помощью комбинаций, предварительно заданных, горячих клавиш. Например, вставка отдельных элементов или текстов целиком, запуск приложений при

нажатию определённого пользовательского сочетания символов, переназначение стандартных сочетаний клавиш (таких как «Ctrl + V», «Ctrl + C» и т.д.) или кнопок мыши на горячие клавиши по вашему выбору, создание диалоговых окон или полноценных программ.

6. Программное обеспечение управления графическим оборудованием (например, драйвера продуктов компании «NVIDIA») могут осуществлять внедрение собственных библиотек динамической компоновки «DLL» для выполнения различных графических задач на высоком уровне производительности.

7. Отдельные виды стороннего программного обеспечения внедряют динамически подключаемые библиотеки «DLL» для расширения списка дополнительных функций меню приложений.

8. Нередко, отдельные участники компьютерных игр особо популярных видов, применяют инструменты мошенничества и внедряют программные коды динамической компоновки для изменения модели поведения игры с целью получения несправедливого преимущества над другими игроками.

Рассмотрим технику внедрения Reflective DLL Injection. Обычно для загрузки DLL-библиотеки в Windows вызывается функция LoadLibrary, которая принимает в качестве аргумента путь к библиотеке. Для этого нужно, чтобы библиотека была на диске, с которого и производится загрузка. Тем не менее техника внедрения кода Reflective DLL Injection позволяет не использовать функцию для загрузки библиотеки в адресное пространство стороннего процесса, внедрение производится с использованием функций записи байтов библиотеки в адресное пространство другого процесса.

### **Техника внедрения кода Reflective DLL Injection.**

Техника Reflective DLL Injection позволяет внедрить код DLL-библиотеки в процесс из памяти. Основное преимущество такого подхода заключается в том, что библиотека не регистрируется в системе. В результате ее

практически невозможно обнаружить ни на уровне системы, ни на уровне процесса, в который она была загружена.

Техника реализуется путём выполнения следующих шагов:

1. Некоторый исполняемый файл считывает DLL-библиотеку с диска в адресное пространство своего процесса и передаёт управление на её экспортируемую функцию `ReflectiveLoader`.

2. Поскольку теперь библиотека существует в произвольном месте в памяти, `ReflectiveLoader` вычисляет текущее местоположение самой DLL-библиотеки в памяти. Для этого `ReflectiveLoader` получает адрес текущей инструкции и, двигаясь в обратном направлении, ищет байты 4D5A, соответствующие MZ-сигнатуре. Это нужно, чтобы дать библиотеке возможность анализировать свои собственные заголовки для дальнейшего запуска.

3. `ReflectiveLoader` определяет адрес библиотеки `kernel32.dll` в текущем процессе, после чего анализирует таблицу ее экспорта и находит функции `LoadLibrary`, `GetProcAddress` и `VirtualAlloc`, необходимые для дальнейшей загрузки.

4. Теперь `ReflectiveLoader` выделяет непрерывный участок памяти (`VirtualAlloc`), где и размещает код DLL-библиотеки (заголовки и секции) в соответствии с виртуальными адресами. Затем `ReflectiveLoader` обрабатывает свою вновь загруженную таблицу импорта: загружает необходимые библиотеки (`LoadLibrary`) и импортируемые из них функции (`GetProcAddress`).

5. Наконец, `ReflectiveLoader` вызывает `DllEntryPoint` внедряемой библиотеки. С этих пор внедрение кода считается успешно произведенным. Запускаются алгоритмы, описанные в теле функции

### **Техника внедрения кода Code Injection.**

Внедрение кода распространено в Windows. Приложения «вставляют» части своего собственного кода в другой запущенный процесс, чтобы изменить

его поведение. Этот метод можно использовать во благо или во зло, но в любом случае он может вызвать проблемы.

Внедрение кода также обычно называется внедрением DLL, потому что введённый код часто имеет форму DLL. Однако приложения также могут внедрять в процесс другие типы кода, не являющиеся библиотеками DLL. Данный метод обладает преимуществами при внедрении. Файл с машинными инструкциями в большинстве случаев не распознает антивирус, что даёт возможность внедрять код, не беспокоясь о сохранности его содержимого.

Внедрение кода используется для выполнения всевозможных трюков и функций в Windows. Хотя его используют легитимные программы, он также используется вредоносными программами. Например:

Антивирусные программы часто внедряют код в веб-браузеры. Они могут использовать его, например, для отслеживания сетевого трафика и блокировки опасного веб-контента.

Вредоносные программы могут добавлять код в ваш веб-браузер, чтобы лучше отслеживать ваши действия в Интернете, красть защищённую информацию, такую как пароли и номера кредитных карт, а также изменять настройки браузера.

WindowBlinds Stardock, который задаёт темы для рабочего стола, внедряет код в изменить способ отображения окон .

Stardock's Fences внедряет код в изменить способ работы рабочего стола Windows .

AutoHotkey, который позволяет создавать сценарии и назначать им общесистемные горячие клавиши , внедряет код для этого.

Графический драйвер подобен встроенным DLL от NVIDIA для выполнения множества задач, связанных с графикой.

### **Техника внедрения кода DLL Injection.**

DLL инъекция даёт возможность выполнять свой код в адресном пространстве уже запущенного процесса. Многие используют инфицирования

для написания читов для игр, выполнения вредоносных действий для системы и т.п. Но данный прим не обязательно применять для реализации коварных планов, а например, для обновления своего приложения.

Алгоритм работы очень просто, нам нужно создать поток в стороннем процессе и внедрить в него выполнения нашего кода. Для этого в стороннем процессе выделяется память необходимая для хранения пути до внедряемой библиотеки. Далее с помощью создания нового потока в стороннем процессе загружается библиотека, путь до которой указали ранее.

### **2.3 Анализ аналогов и прототипов.**

В сети Интернет существуют сотни программ, способных внедрять код в сторонние процессы. Большинство найденных программ являются консольными программами, предусматривающие наличие у пользователя минимальных знаний работы с консолью. Помимо консольных программ в Сети также есть графические программы. У всех этих программ имеется недостаток — отсутствие исходного кода, что значительно затрудняет анализ алгоритма работы программы.

#### **Reflective DLL Injection.**

Самой известной программой для внедрения кода, используя технику Reflective DLL Injection, является работа Стивена Фьювера, размещённая на GitHub. Проект представляет собой программу для внедрения кода и библиотеку, предназначенную для внедрения в сторонний процесс. Программа является консольной, что вызывает затруднения у некоторых пользователей при работе с консолью. Преимуществами данного проекта являются работоспособность программы, возможность изменения исходного кода как программы-инжектора, так и внедряемой библиотеки; открытый исходный код.

Все преимущества программы удовлетворяют требованиям технического задания, программу можно использовать как готовый модуль для работы АС внедрения кода, внося незначительные изменения.



## **DLL Ijection.**

В качестве программы для внедрения кода в сторонний процесс, используя технику DLL Injection, была рассмотрена работа автора DarthTon, размещенная на платформе Github.

Преимуществами данной программы являются: поддержка как x64, так и x86 разрядных процессов и модулей, возможность внедрения в режиме ядра (применяется для внедрения драйверов), внедрение в нативные процессы (те, у которых загружен только ntdll.dll), вызов пользовательской процедуры инициализации после инъекции, открытый исходный код.

Недостатками являются: громоздкий исходный код, консольный вид программы.

Для реализации модуля для внедрения кода необходимо разработать собственное решение.

## **Code Injection.**

Программа для внедрения кода посредством использования техники Code Injection является простой в реализации, что является основанием для разработки собственного программного модуля для внедрения кода, использующего данную технику внедрения.

## **2.4 Перечень задач, подлежащих решению в процессе разработки**

1. Разработать работающую автоматизированную систему для внедрения кода, используя различные техники внедрения кода.
2. Разработать программные модули реализующие каждую из техник внедрения кода в сторонние процессы.
3. Разработать алгоритм считывания данных, введенных пользователем, из соответствующих полей ввода.
4. Разработать алгоритм вызова модулей внедрения кода в сторонний процесс с передачей параметров.

5. Разработать алгоритмы внедрения кода в сторонние процессы, использующие различные техники внедрения.

## **2.5 Обоснование выбора инструмента и платформы для разработки**

Для реализации модулей внедрения кода в сторонние процессы необходимо использовать язык С. Язык С не содержит промежуточных библиотек для обращения к модулям операционной системы, что упрощает поиск описания всех API-функций предоставляемых операционной системой. Код, написанный на С является более быстрым по сравнению с кодом, написанным на других высокоуровневых языках программирования.

Платформой разработки может служить любой текстовый редактор и компилятор языка С. Для разработки программ может использоваться среда разработки от компании Microsoft Visual Studio 2022. Основанием для её использования служит быстрое написание кода благодаря встроенной в интерфейс программы подсветки синтаксиса и системы автоматического заполнения кода Intellesence. Встроенный компилятор избавляет разработчика от необходимости компилировать код программы, используя консольный компилятор. Встроенный отладчик обеспечивает быстрое решение возникших проблем в ходе выполнения программы.

## **3 Проектно-конструкторская часть**

### **3.1 Разработка структуры приложения**

Автоматизированная система внедрения кода предназначена для упрощения использования консольных программ внедрения кода в сторонние процессы за счёт построения графической оболочки и автоматизации процесса внедрения кода в сторонний процесс. Система предназначена для использования пользователями, имеющими базовые навыки работы с компьютером.

Основными потребителями автоматизированной системы будут служить пользователи, работа которых заключается в изучении принципов работы программ, внедряющих код в сторонние процессы, а также программисты, целью которых является написание внедряемых файлов и производство их внедрение в сторонние процессы используя разрабатываемую автоматизируемую систему.

Разработка компонентов АС должна быть произведена в следующей последовательности: в первую очередь необходимо разработать модули внедрения кода в сторонний процесс, затем разработать графическую оболочку для запуска всех разработанных модулей с параметрами, полученными от пользователя в текстовых полях ввода графической формы.

Автоматизированная система разрабатывается для платформы Windows x86-64, что даёт возможность работать как на 32-битных операционных системах, так и на 64-битных.

### **3.2 Разработка алгоритмов обработки информации**

Автоматизированная система является комплексной системой, в которой каждый отдельный модуль выполняет определённые действия. В каждом отдельном модуле внедрения кода в сторонний процесс присутствует алгоритм внедрения, представляющий собой последовательность действий, приводящих к тому или иному результату. Для более содержательного отражения работы всех алгоритмов были разработаны блок-схемы, показывающие как работает тот или иной алгоритм отдельно взятых модулей. Все блок-схемы размещены в графической части курсовой работы.

Алгоритмы внедрения кода в сторонние процессы имеют схожие принципы работы, однако алгоритм каждого из модулей отличается от алгоритмов других модулей внедрения кода. Подробнее работа каждого алгоритма отображена на блок-схемах ниже.

## Блок-схемы алгоритмов

### Блок-схема алгоритма CodeInjection

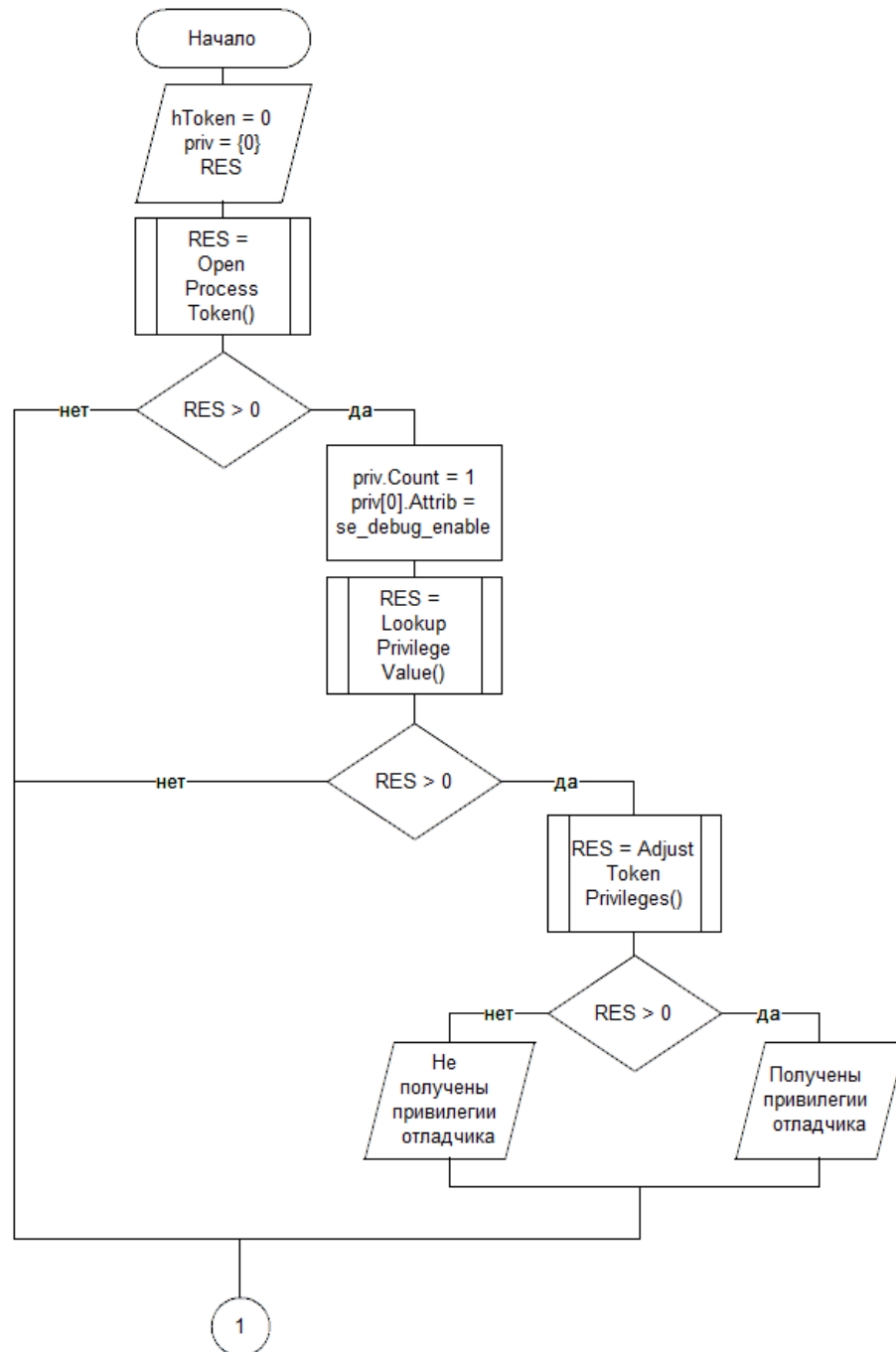


Рисунок 1. Блок-схема алгоритма Code injection

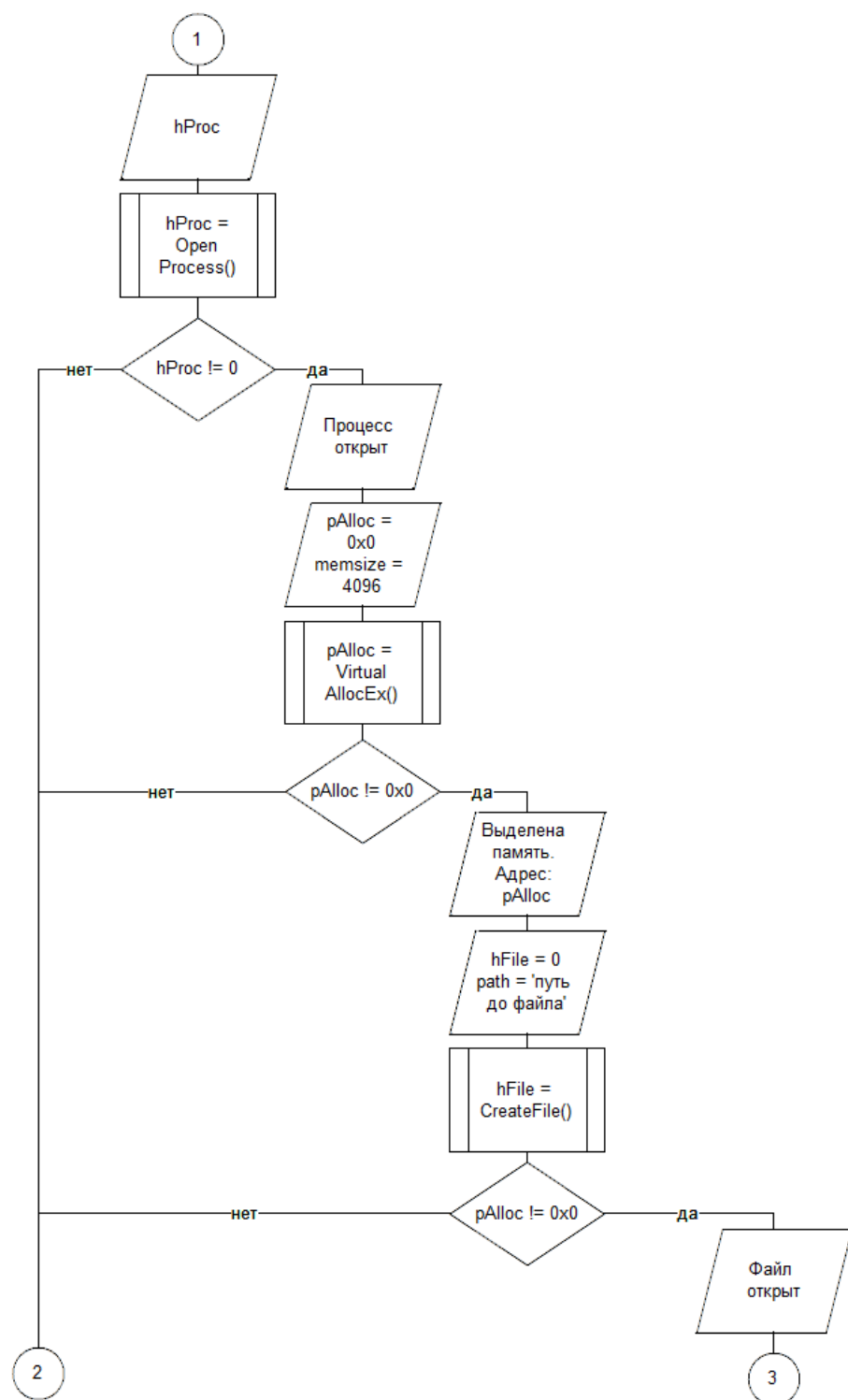


Рисунок 2. Блок-схема алгоритма Code injection

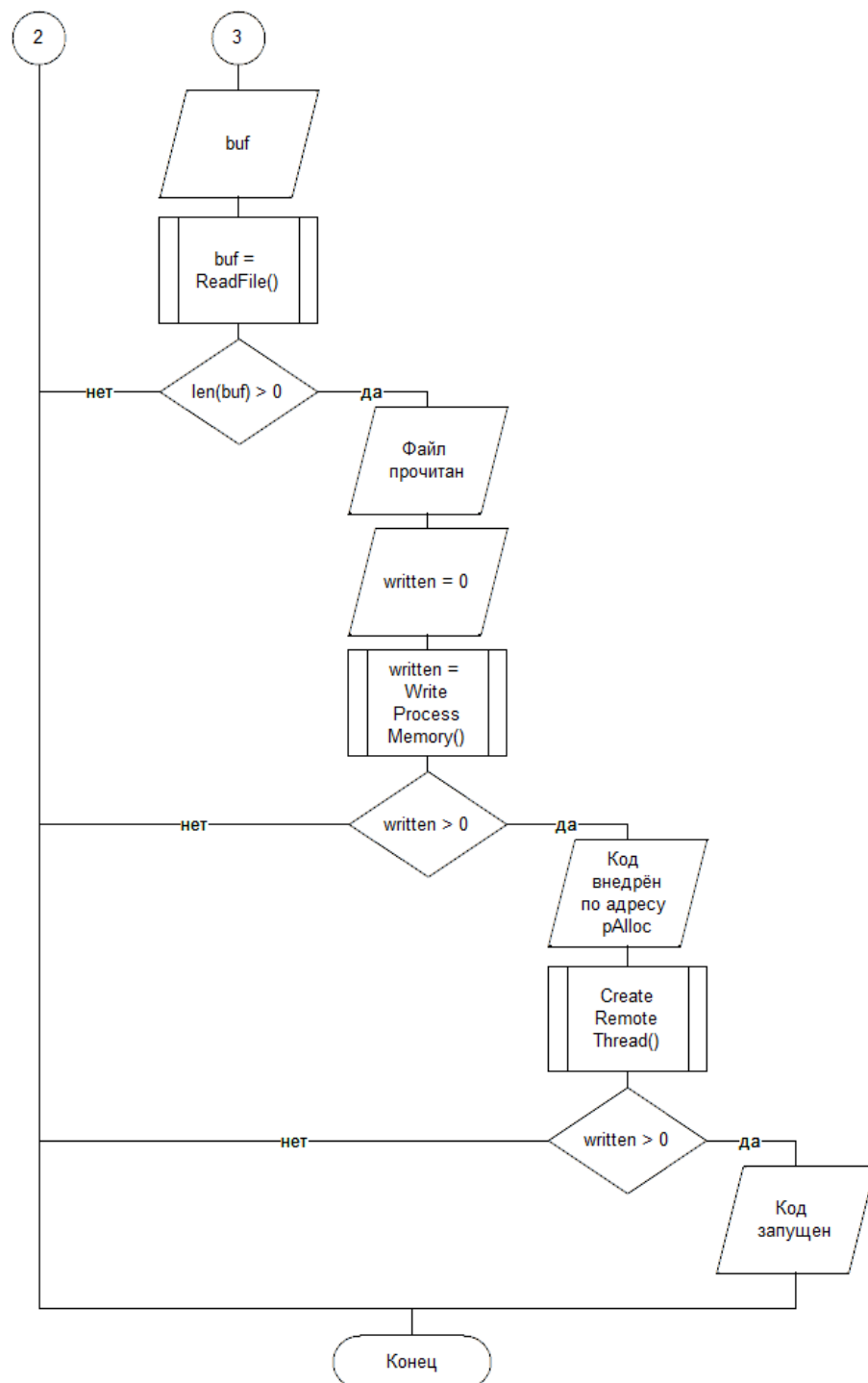


Рисунок 3. Блок-схема алгоритма Code injection

### Блок-схема алгоритма Dll Injection

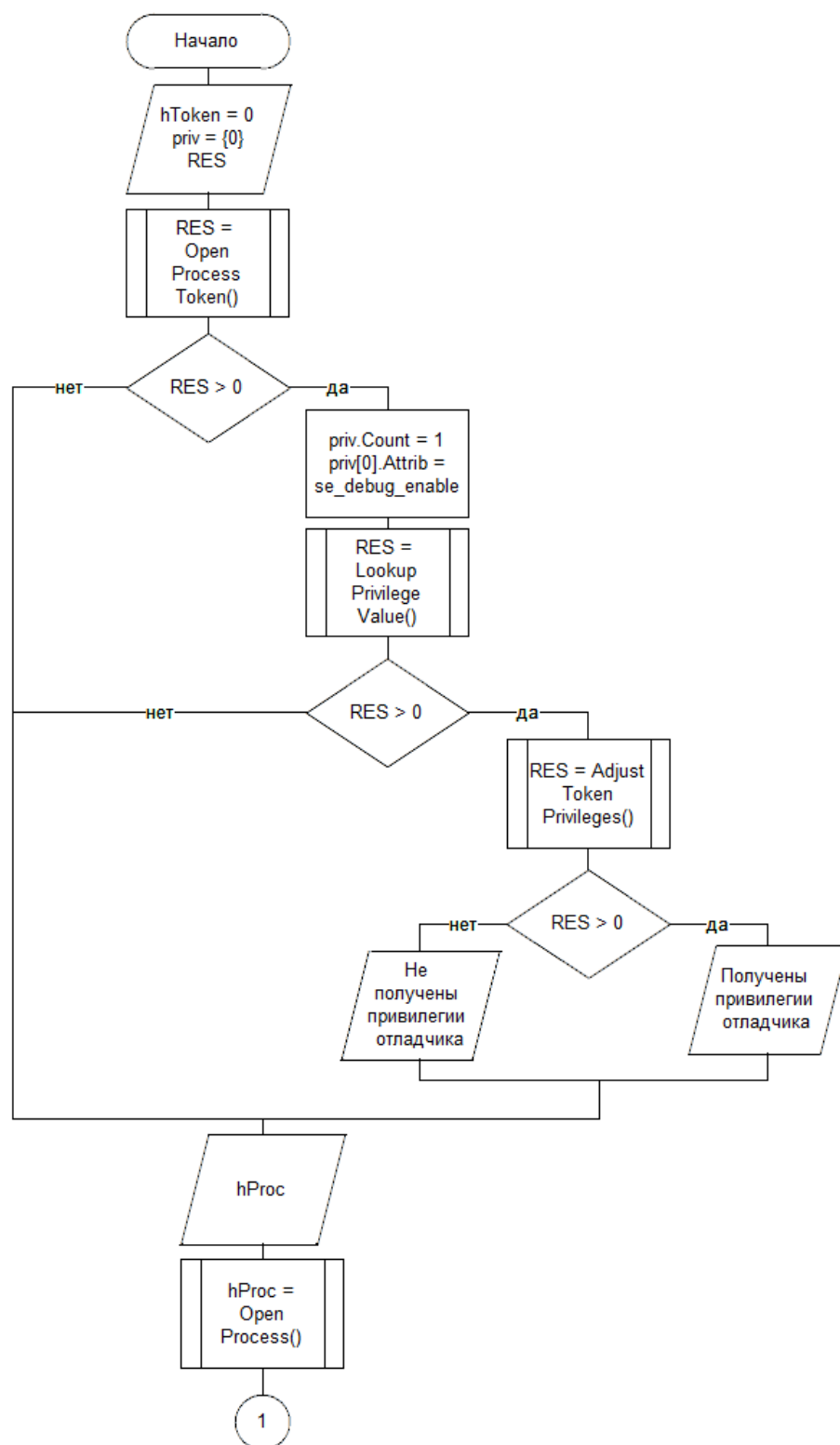


Рисунок 4. Блок-схема алгоритма Dll Injection

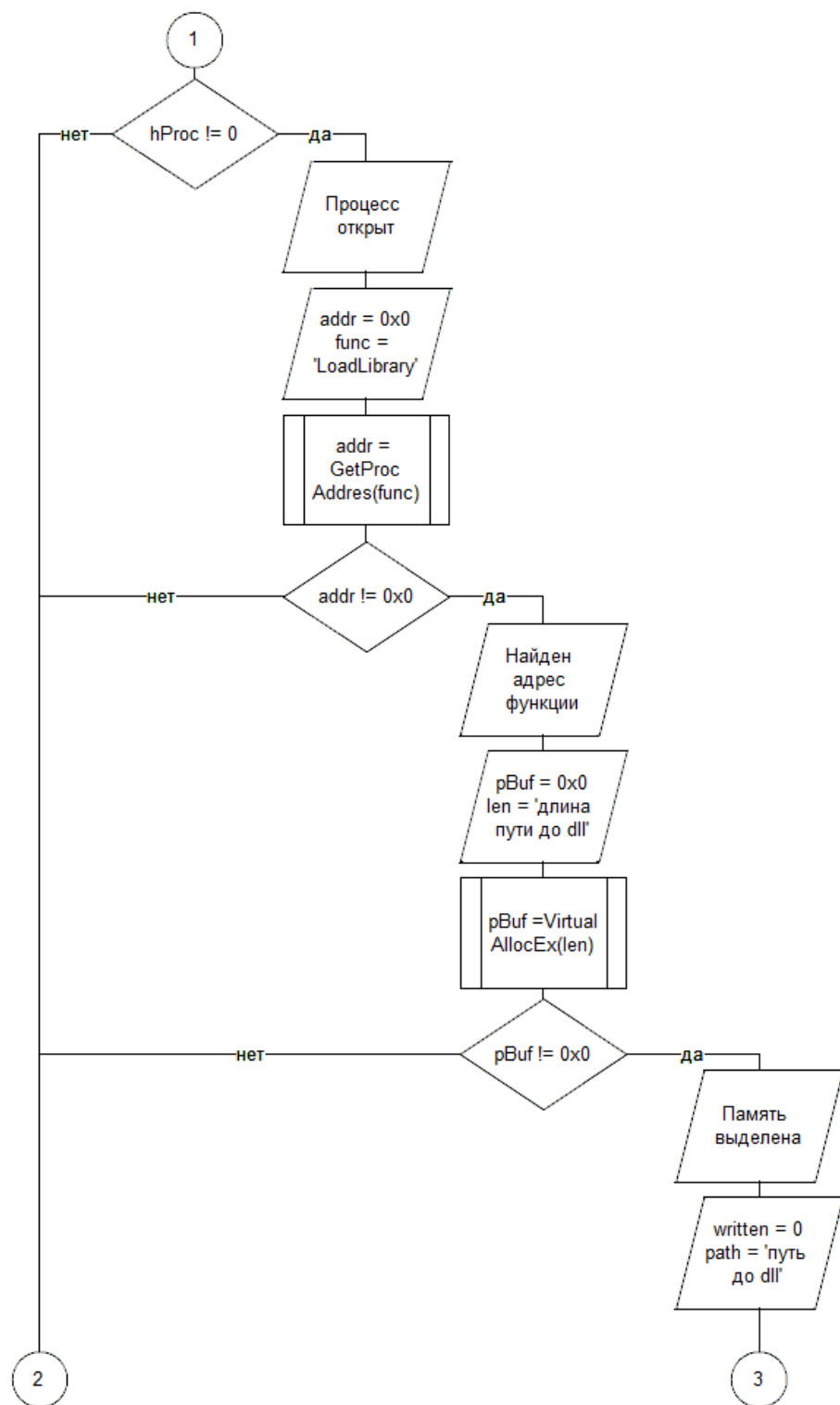


Рисунок 5. Блок-схема алгоритма Dll Injection



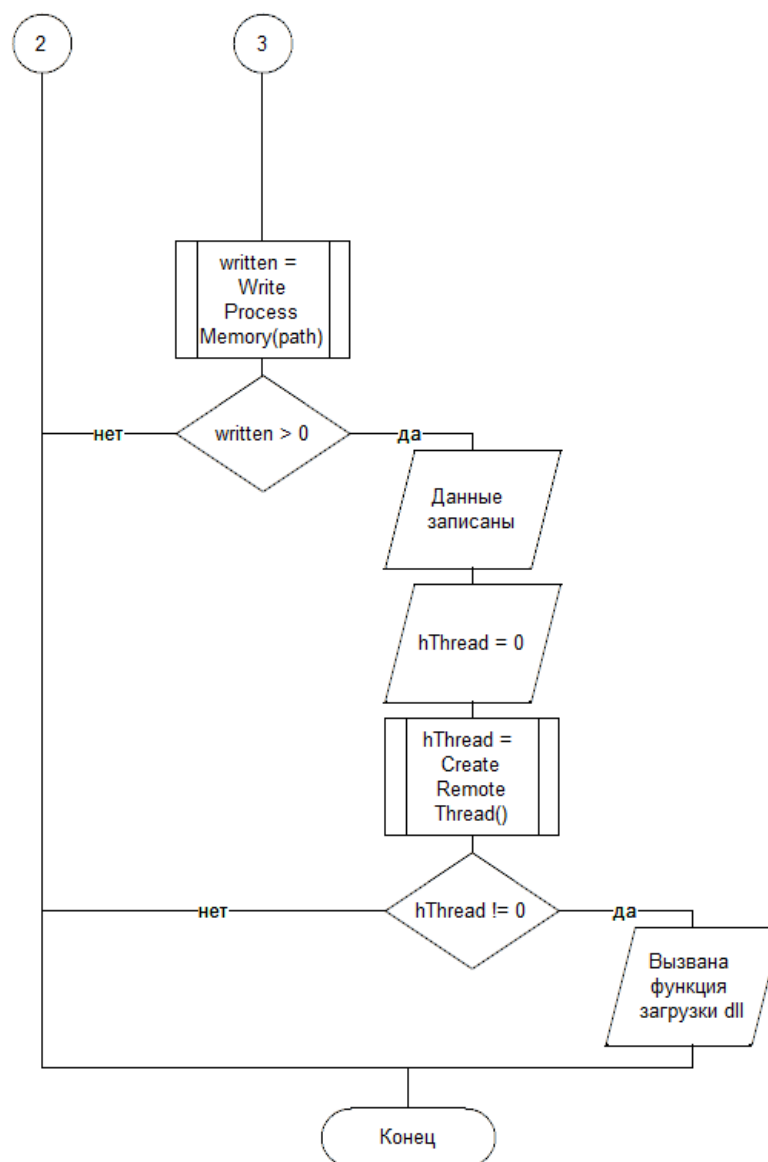


Рисунок 6. Блок-схема алгоритма Dll Injection

## Блок-схема алгоритма Reflective Dll Injection

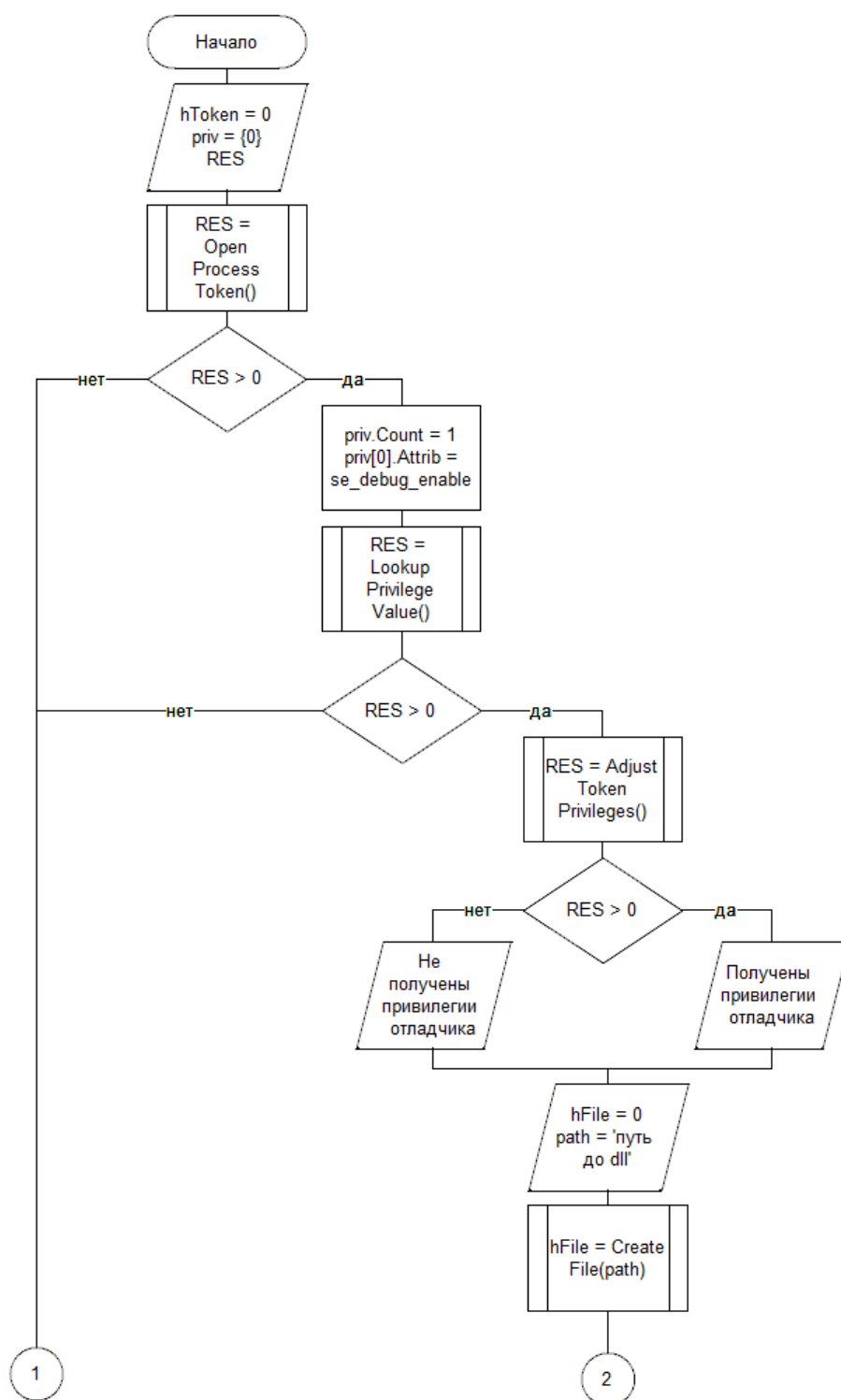


Рисунок 7. Блок-схема алгоритма Reflective Dll Injection

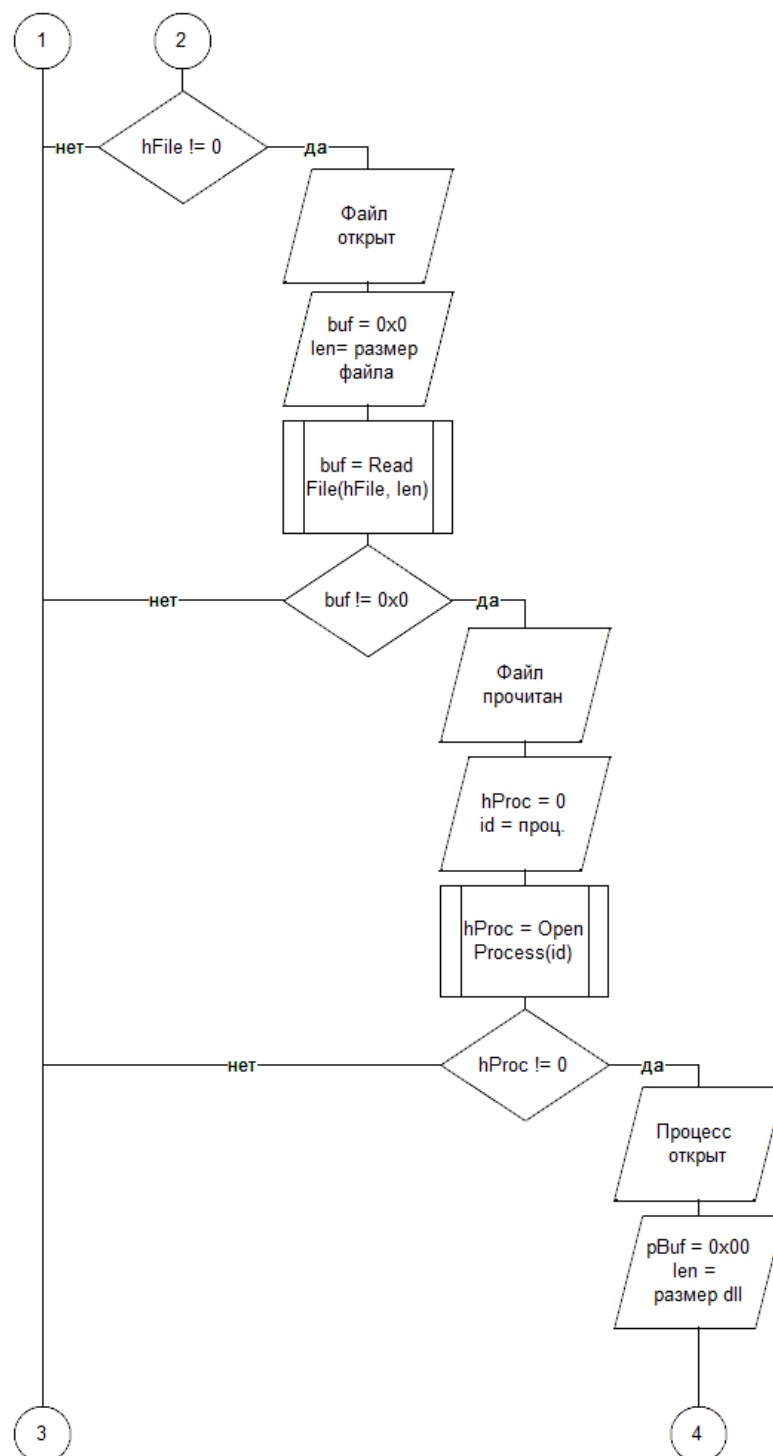


Рисунок 8. Блок-схема алгоритма Reflective Dll Injection

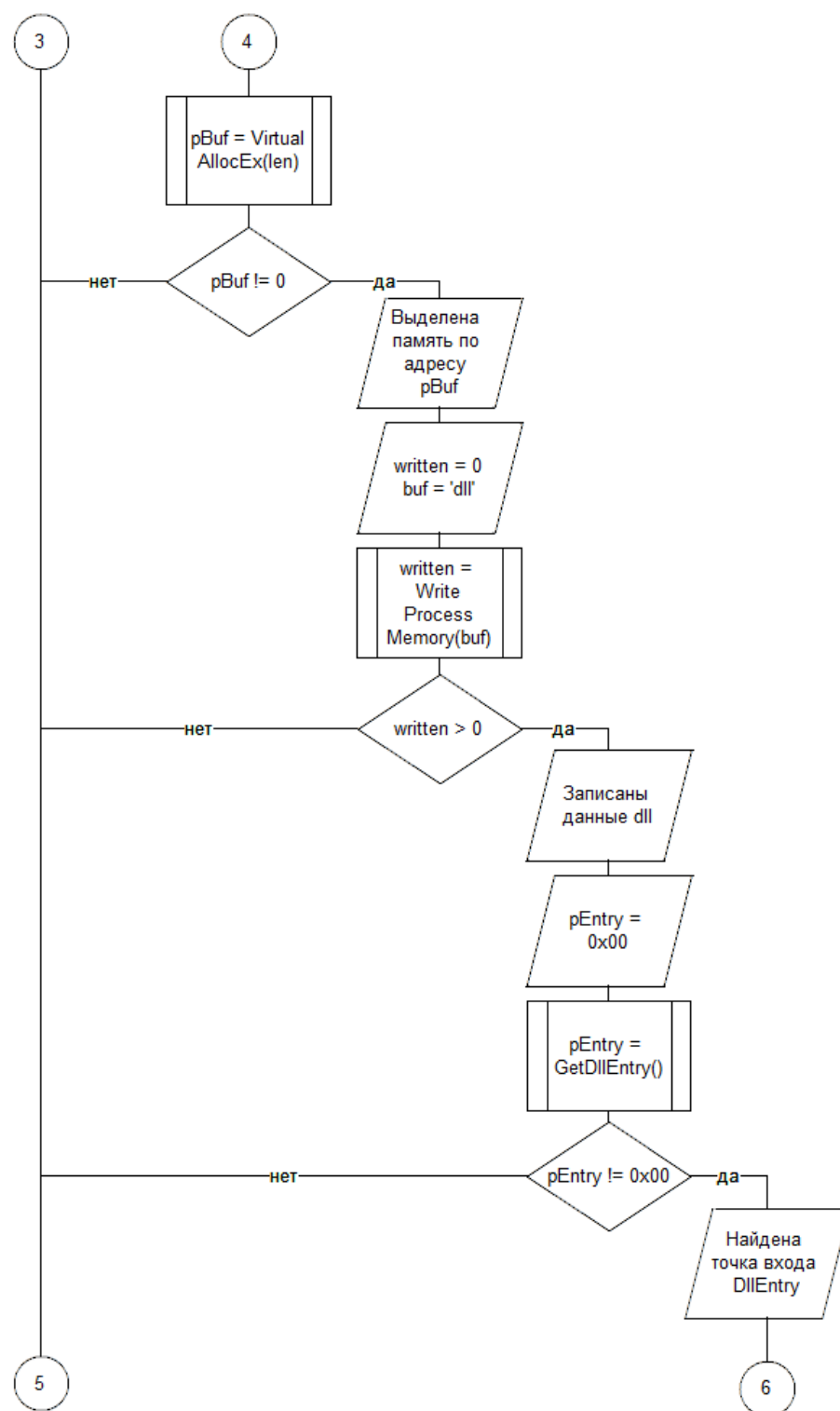


Рисунок 9. Блок-схема алгоритма *Reflective Dll Injection*

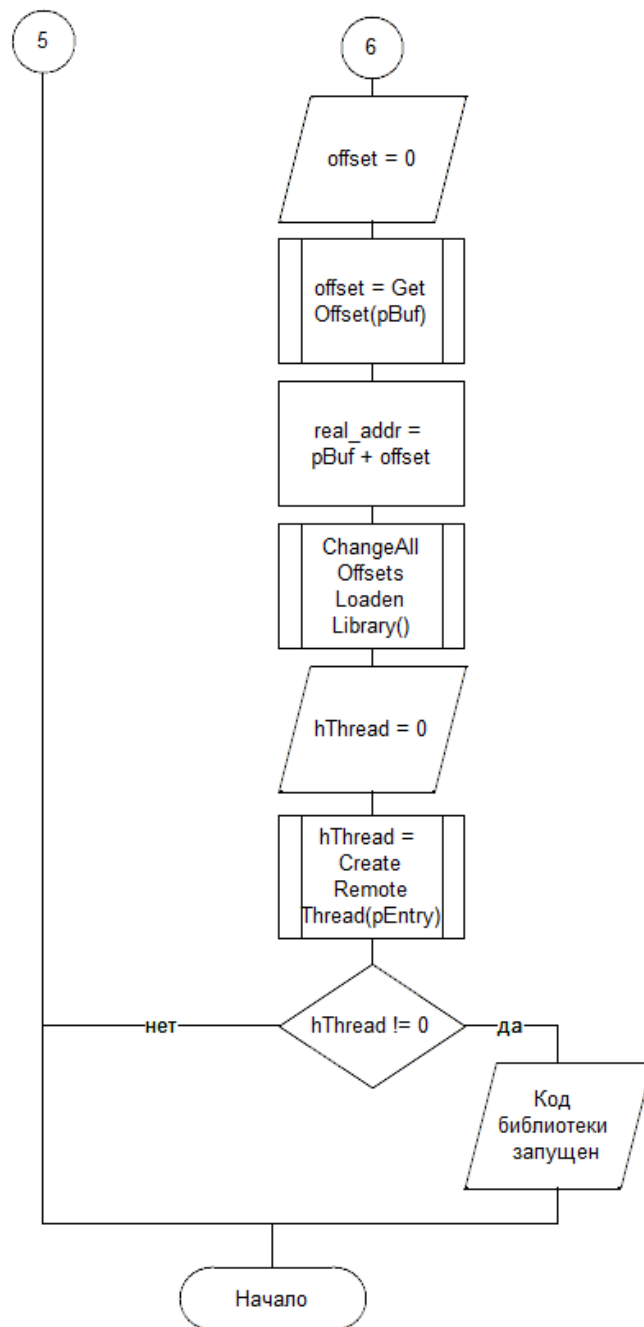


Рисунок 10. Блок-схема алгоритма Reflective Dll Injection

## **Описание алгоритмов**

### *Алгоритм работы модуля внедрения кода Code Injection:*

1. Получить привилегии отладчика для текущего процесса
2. Получить дескриптор стороннего процесса, используя идентификатор
3. Выделить память в стороннем процессе.
4. Открыть файл с внедряемым кодом.
5. Прочитать файл в память текущего процесса.
6. Записать содержимое файла в память стороннего процесса.
7. Запустить внедренный код путем создания нового потока.

### *Алгоритм работы модуля внедрения кода Dll Injection:*

1. Получить привилегии отладчика для текущего процесса.
2. Получить дескриптор стороннего процесса.
3. Найти адрес функции LoadLibrary
4. Выделить память в стороннем процессе.
5. По адресу выделенной памяти записать путь до внедряемой dll.
6. Вызвать функцию в отдельном потоке, передав в качестве параметра путь до внедряемой dll.

### *Алгоритм работы модуля внедрения кода Reflective Dll Injection:*

1. Получить привилегии отладчика для текущего процесса.
2. Получить дескриптор внедряемого dll-файла.
3. Прочитать файл в память текущего процесса.
4. Получить дескриптор стороннего процесса.
5. Выделить память в стороннем процессе.

6. Внедрить dll-файл по адресу выделенной памяти.
7. Найти функцию ReflectiveLoader
8. Вызвать функцию в отдельном потоке.

**Требуемые библиотеки и функции для реализации алгоритмов в полном объёме.**

В процессе проектирования алгоритмов были определены требуемые библиотеки для работы автоматизированной системы, а также функции, предоставляемые данными библиотеками.

*Функции библиотеки kernel32.dll*

1. GetCurrentDirectory — извлекает текущий каталог для текущего процесса. Принимает параметры: размер буфера; указатель на буфер, в который будет записан путь текущей директории. Возвращает 0 в случае ошибки; в случае успеха — длину текущей директории в байтах.

2. OpenProcess — открывает существующий объект локального процесса. Принимаемые параметры: параметр, определяющий права доступа при открытии процесса; параметр, определяющий наследование дескриптора созданными процессами; идентификатор процесса. Возвращаемое значение — дескриптор открытого процесса в случае успеха.

3. VirtualAllocEx — резервирует, фиксирует или изменяет состояние области памяти в виртуальном адресном пространстве указанного процесса. Функция инициализирует память, которую она выделяет, равным нулю. Принимаемые параметры: дескриптор процесса; требуемый адрес для выделения области памяти; размер выделяемой памяти; тип выделения памяти; параметры защиты памяти. Возвращаемое значение: базовый адрес выделенной памяти, в случае успеха; в случае неудачного выделения памяти функция вернёт нулевое значение.

4. VirtualQueryEx — извлекает информацию о диапазоне страниц в виртуальном адресном пространстве указанного процесса. Принимаемые параметры: дескриптор процесса; базовый адрес региона памяти стороннего процесса; указатель на структуру MEMORY\_BASIC\_INFORMATION, содержащую всю информацию о регионе памяти; размер структуры в байтах. Возвращаемое значение: в случае успеха функция вернёт количество байтов записанных в структуру MEMORY\_BASIC\_INFORMATION; в случае ошибки функция вернёт 0.

5. CreateFile — создаёт, открывает устройство ввода-вывода. Принимаемые параметры: указатель на строку с именем файла; режим доступа при открытии файла; режим совместного использования файла; указатель на структуру SECURITY\_ATTRIBUTES; флаги открытия файла; флаги атрибутов; дескриптор временного файла. Возвращаемое значение: в случае успеха функция вернёт дескриптор открытого файла; в случае ошибки функция вернёт -1 в качестве результата своей работы.

6. GetFileSize — извлекает размер файла. Принимаемые параметры: дескриптор файла; указатель на переменную, в которую возвращается двойное слово старшего порядка размера файла. Возвращаемое значение: в случае успеха функция вернёт двойное слово младшего порядка размера файла (его размер в байтах).

7. HeapAlloc - выделяет блок памяти из кучи. Выделенная память не может быть перемещена. Принимаемые параметры: дескриптор кучи процесса; флаги выделения памяти; количество байтов для выделения памяти. Возвращаемое значение: в случае успеха функция вернёт указатель на выделенную память или 0 в случае ошибки.

8. ReadFile — считывает данные из указанного файла или устройства ввода-вывода. Считывание происходит в позиции, указанной указателем файла, если это поддерживается устройством. Принимаемые параметры: дескриптор файла; указатель на буфер, куда будет записано содержимое файла; количество



байтов для прочтения; указатель на переменную, хранящую количество прочитанных байтов; указатель на структуру OVERLAPPED. Возвращаемое значение: в случае успеха функция вернёт значение отличное от нуля.

9. WriteProcessMemory — записывает данные в область памяти в указанном процессе. Вся область, в которую должна быть записана, должна быть доступна, иначе операция завершится неудачей. Принимаемые параметры: дескриптор процесса; базовый адрес для записи; указатель на буфер с данными для записи; размер буфера для записи; указатель на переменную в которой будет храниться число байт, записанных в сторонний процесс. Возвращаемое значение: в случае успеха функция вернёт значение, отличное от 0.

10. CreateRemoteThread — создает поток, который выполняется в виртуальном адресном пространстве другого процесса. Принимаемые параметры: дескриптор процесса, указатель на структуру SECURITY\_ATTRIBUTES, размер создаваемого стека; начальный адрес выполнения потока; параметры для выполнения функции в потоке; флаги создания потока; указатель на переменную, содержащую идентификатор потока. Возвращаемое значение: дескриптор нового потока в случае успешного создания потока; в случае ошибки функция вернёт 0.

11. GetProcAddress — извлекает адрес экспортируемой функции или переменной из указанной библиотеки динамических ссылок (DLL). Принимаемые параметры: дескриптор модуля, содержащего функции или переменные; указатель на строку с именем функции. Возвращаемое значение: адрес функции в библиотеке в случае успеха, 0 — в случае ошибки.

12. CloseHandle — закрывает открытый дескриптор объекта. Принимаемые параметры: дескриптор закрываемого объекта. Возвращаемое значение: в случае успеха функция вернёт отличное от 0 значение.

11. OpenProcessToken — открывает маркер доступа, связанный с процессом. Принимает дескриптор процесса, специальную маску доступа и

указатель на дескриптор открытого маркера доступа. Возвращаемое значение: ненулевое в случае успеха.

#### *Функции библиотеки user32.dll*

1. `MessageBox` — отображает модальное диалоговое окно, содержащее значок системы, набор кнопок и краткое сообщение для конкретного приложения, например информацию о состоянии или ошибке. Окно сообщения возвращает целое значение, указывающее, на какую кнопку нажал пользователь. Принимаемые параметры: дескриптор окна-владельца; информационное сообщение; название диалогового окна; тип диалогового окна. Возвращаемое значение может отличаться при использовании разных типов окна `MessageBox`.

2. `SendMessage` — отправляет указанное сообщение в окно или окна. Функция `SendMessage` вызывает оконную процедуру для указанного окна и не возвращается до тех пор, пока оконная процедура не обработает сообщение. Принимает параметры: дескриптор окна, оконная процедура которого получит сообщение; сообщение, которое будет опрарвлено; дополнительное сообщение со специальной информацией; дополнительно сообщение со специальной информацией. Возвращаемое значение зависит от полученных параметров.

3. `SetWinowText` — изменяет текст строки заголовка указанного окна. Принимаемые параметры: дескриптор окна, указатель на строку с новым текстом. Возвращаемое значение не равно нулю если функция завершилась успешно.

4. `CreateWindow` — создаёт перекрывающееся, всплывающее или дочернее окно. В качестве параметров принимает: имя класса окна, имя окна в строке заголовка, уровень расположения окна, стандартные позиции *x* и *y*; размера окна, дескриптор владельца окна; используемый класс меню; дескриптор экземпляра приложения. Возвращаемое значение — дескриптор вновь созданного окна.

5. ShowWindow — устанавливает специальное состояние окна. Принимаемые параметры: дескриптор окна, специальная опция для окна. Возвращаемое значение: если окно ранее было видимым, то вернётся значение отличное от нуля, если ранее окно не было видимым, то вернётся значение равное нулю.

6. RegisterClassA — регистрирует класс window для последующего использования в вызовах функции CreateWindow или CreateWindowEx. Принимаемые параметры: указатель на структуру WNDCLASSA, необходимую для регистрации класса окна. Возвращает уникальный идентификатор зарегистрированного класса окна. В случае ошибки возвращает значение, равное нулю.

7. PeekMessage — отправляет входящие сообщения, не поставленные в очередь, проверяет очередь сообщений потока на наличие отправленного сообщения и извлекает сообщение (если таковое существует). Принимаемые параметры: указатель на структуру MSG, принимающую сообщение; дескриптор окна из которого должно быть извлечено сообщение; значение первого сообщения в диапазоне сообщений, подлежащих проверке; значение последнего сообщения в диапазоне сообщений, подлежащих проверке; параметр, определяющий режим обработки сообщения. Возвращаемое значение: если сообщение доступно, вернётся ненулевое значение.

8. DispatchMessage — отправляет сообщение оконной процедуре. Принимает структуру MSG в качестве параметра. Возвращаемое значение — значение, возвращаемое оконной процедурой.

9. TranslateMessage — преобразует сообщения с виртуальным ключом в символьное сообщение, символьные сообщения отправляются в очередь сообщений вызывающего потока для чтения при следующем вызове потоком функции GetMessage или PeekMessage. Принимаемые параметры: указатель на структуру MSG. Возвращаемое значение: ненулевое значение в случае успеха преобразования.

### *Функции библиотеки comdlg.dll*

1. `GetOpenFileName` — создаёт диалоговое окно открытия, которое позволяет пользователю указать диск, каталог и имя файла или набора файлов, которые будут открыты. Принимаемые параметры: указатель на структуру `OPENFILENAME`, содержащую параметры для выбора файла. Возвращаемое значение отличное от 0, если пользователь выбрал файл.

## **3.3 Разработка архитектуры приложения**

Варианты воплощения программного продукта.

Вариант №1, предусматривающий наличие одного графического компонента и дополнительных компонентов, предназначенных для внедрения кода в сторонние процессы. При этом графический компонент выполнен с применением функций отрисовки изображений для лучшего взаимодействия с пользователем, а модули внедрения кода представлены в виде библиотек динамической компоновки. Каждая из библиотек загружается в память вызывающего процесса (графического модуля), после чего начинается выполнение, алгоритмов, прописанных в теле функции `DllMain` каждой из библиотек.

Вариант №2, предусматривающий наличие модулей внедрения кода в виде готовых приложений `.exe` формата и графического модуля такого же формата. При вызове одного из модулей, происходит обращение к командной строке текущего процесса, в командную строку записывается команда, представляющая собой инструкцию вызова того или иного модуля, в зависимости от выбора пользователя, с передачей параметров в запускаемый модуль.

Вариант №3, не предусматривающий наличие разделения на модули. Представляет собой готовый проект формата `.exe`, содержащий реализацию всех модулей внедрения кода в сторонний процесс, а так же графическую оболочку.

### **Предполагаемое качество функционирования системы и её компонентов на моделях различного уровня.**

Вариант 1 является надёжным вариантом проектирования автоматизированной АС. Надёжность достигается за счёт разделения программных модулей на отдельные компоненты, что исключает выход из строя всей автоматизированной системы в случае выхода из строя отдельного модуля. Недостатком данной системы является невозможность в ручном режиме запустить один из модулей.

Вариант 2 является аналогом варианта 1 по функционированию, однако в представленном варианте модели автоматизированной системы модули внедрения кода представляют собой консольные программы, которые можно запускать из консоли.

Вариант 3 является небезопасным вариантом работы информационной системы из-за отсутствия разделения на модули. При выходе из строя одного компонента, из строя выходит вся автоматизированная система.

#### **Выбор наилучшего варианта реализации.**





Наиболее подходящим вариантом является вариант под номером 2. Основанием для выбора данного варианта является возможность запуска модулей как в автоматическом режиме, используя графическую оболочку, так и в ручном режиме, вводя команду запуска в командную строку. Преимуществом данного варианта является также наличие разделения модулей на отдельные составляющие, что исключает выход из строя всей автоматизированной системы в случае выхода из строя одного из компонентов системы.

### **3.4 Реализация готового приложения**

Реализация приложения была проведена согласно пунктам 3.1-3.3, определяющих структуру, архитектуру и основные алгоритмы приложения. Приложение было разработано с учётом пунктов 1.4, в которых описаны основные требования к функционированию автоматизированной системы, её

внешнему виду, а также видам обеспечения, учтены дополнительные требования к системе, предъявленные в пункте 1.4.1.14 технического задания.

Результатом реализации готового приложения служит информационная система, требующая проведения тестирования работоспособности модулей для внедрения кода и графического интерфейса пользователя перед вводом в эксплуатацию. Готовая информационная система представляет собой набор из четырёх исполняемых файлов в формате .exe. Основным приложением для запуска всей системы является исполняемый файл с наименованием «Pig», запустив который пользователь может работать со всей системой.

Имя	Размер	Тип
 module_code_injector.exe	11 КБ	Приложение
 module_dll_injector.exe	13 КБ	Приложение
 module_reflective_dll_injector.exe	10 КБ	Приложение
 Pig.exe	14 КБ	Приложение

*Рисунок 11. Содержание информационной системы*

### **3.5 Разработка интерфейса взаимодействия пользователя с системой**

При разработке интерфейса были выполнены следующие требования:

- Интерфейс пользователя реализован на русском языке.
- Интерфейс не переполнен графическими примитивами: созданы только нужные для работы элементы интерфейса.
- Все элементы интерфейса выполнены в едином стиле и цветовой схеме.
- При использовании только API-функций операционной системы для реализации интерфейса.
- Отчёт о работе модулей внедрения кода должен быть выведен пользователю в виде открытого текстового документа.
- Реализована защита от некорректного ввода пользователя.
- Установлено ограничение на количество вводимых символов.
- Установлено ограничение на расширение пользовательского окна.

### **Описание внешнего вида интерфейса.**

Графический интерфейс представляет собой окно 650x300 пикселей. На окне размещены поля: ввода идентификатора процесса, поле ввода пути до файла с внедряемым кодом, поле ввода текстового файла, необходимого для отображения результатов работы модулей внедрения кода. Напротив каждого поля для ввода путей до файлов размещается кнопка, по нажатию на которую вызывается диалоговое окно для выбора файла. Ниже всех перечисленных элементов интерфейса размещён выпадающий список, предназначенный для выбора типа инъекции кода пользователем. Ниже выпадающего списка размещена кнопка, нажав которую считываются поля формы, на основе полученных данных происходит вызов соответствующего модуля с параметрами, полученными от пользователя.

## **4 Проектно-технологическая часть**

### **4.1 Тестирование и отладка макета рабочей программы.**

Разработанная автоматизированная система внедрения кода в сторонний процесс была протестирована на наличие ошибок в программном коде, способствующих выходу программного обеспечения из строя. В ходе тестирования были проведены следующие мероприятия:

1. Проведены тестовые запуски модулей внедрения кода в сторонний процесс с указанием неправильных параметров. Ожидаемым результатом должно служить завершение модуля и запись причины завершения в файл, указанным пользователем.

- 1.1. Ввод правильных данных идентификатора процесса, неправильных данных — путей до файлов внедряемого кода и файлов для записи отчёта о выполненной модулем работы. Ожидаемый результат — завершение процесса внедрения кода, запись причины остановки в командную строку. Реальный результат соответствует ожидаемому.

1.2. Ввод правильных данных идентификатора процесса, пути до файла, содержащего внедряемый код и неправильных данных выходного файла, служащего для отображения отчёта работы программы. Ожидаемый результат — вывод всех сообщений, являющихся результатом работы программы в командную строку. Реальный результат совпадает с ожидаемым: при неверно указанном пути до выходного файла, модуль внедрения кода выведет все информационные сообщения в командную строку.

1.3. Ввод правильных данных идентификатора процесса и пути до выходного файла и неправильных путей до файла, содержащего внедряемый код. Ожидаемый результат — выход из программы и соответствующая запись в файл о невозможности внедрения.

1.4. Ввод правильных путей до файлов и правильного идентификатора процесса. Ожидаемый результат — успешная запись внедряемого кода в сторонний процесс. Реальный результат — успешная запись в сторонний процесс происходит в случае успешного открытия стороннего процесса на запись, чтение и исполнение и получения его дескриптора. В противном случае сообщение об ошибке будет записано в выходной файл. Для получения прав на запись в системные процессы необходим другой подход к реализации программы — реализация драйвера ядра, способного открывать процессы со всеми правами доступа.

2. Проведены тестовые запуски модулей внедрения кода в сторонние процессы с указанием существующих файлов и правильных дескрипторов.

2.1. Запуск модуля Code Injection с указанием пути до файла, являющимся dll библиотекой. Ожидаемый результат — успешная отработка алгоритмов программы, запись в файл об успешности проведённого внедрения, отсутствие запуска внедрённого кода. Реальный результат соответствует ожидаемому.

2.2. Запуск модуля Reflective DLL Injection с неправильным расширением файла, содержащего внедряемый код. Ожидаемый результат —



успешное внедрение кода в сторонний процесс, запись об успешности внедрения в выходной файл, отсутствие запуска внедрённого кода. Реальный результат соответствует ожидаемому: при запуске модуля внедрения dll библиотеки происходит успешное внедрение файла с произвольным расширением в память стороннего процесса, однако код, внедрённый подобным образом не исполняется.

2.3. Запуск модуля DLL Injection с неправильным расширением. Ожидаемый результат — ошибка внедрения с указанием причины сбоя в выходном файле. Реальный результат — модуль внедрения кода аварийно завершает работу, без указания причины сбоя. В ходе проведения отладочных действий было выявлено несоответствие реального результата ожидаемому, в результате чего был переписан алгоритм обработки загрузки файла с расширением отличающимся от .dll. После проведения повторного тестирования ожидаемым результатом при загрузке файла с расширением, отличающегося от .dll в выходной файл была записана причина отказа в работе программы и программа завершена корректно.

3. Тестирование и отладка графического интерфейса подразумевает проведение мероприятий по выявлению ошибок при взаимодействии пользователя с графическим модулем, представляющим собой систему автоматизации внедрения кода. Тестирование заключалось в проведении ряда следующих мероприятий:

3.1. Нажатие на кнопку внедрения без заполнения необходимых полей ввода путей и идентификатора процесса. Ожидаемый результат — уведомление об ошибке в виде диалогового окна. Реальный результат соответствует ожидаемому.

3.2. Нажатие на кнопку обзора файлов в файловой системе. Ожидаемый результат — открывается диалоговое окно выбора файла из файловой системы. Реальный результат в точности повторяет ожидаемый.

3.3 Закрывать диалоговое окно выбора файла. Ожидаемым результатом является уведомление о том, что путь до того или иного файла выбран не был. Реальный результат — появляется диалоговое окно с предупреждением «Файл не выбран».

## 4.2 Разработка руководства пользователя и руководства программиста.

Система внедрения кода в сторонний процесс не требует особых условий эксплуатации. Ниже приведены основные рекомендации для работы с информационной системой, включающие в себя описание основных элементов графического интерфейса и их предназначение.

Элементы интерфейса

Основные графические элементы представлены на рисунке.

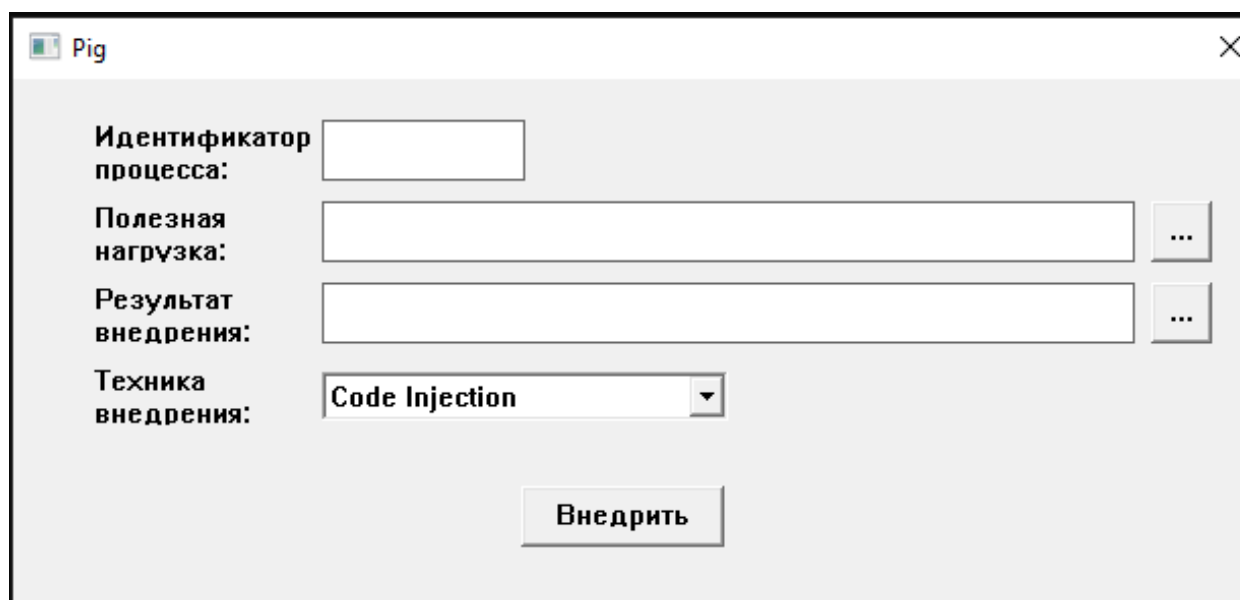
The image shows a screenshot of a software window titled "Pig". Inside the window, there are four labeled input fields arranged vertically on the left side. The first field is labeled "Идентификатор процесса:" and is an empty text box. The second field is labeled "Полезная нагрузка:" and is a text box with a browse button (three dots) to its right. The third field is labeled "Результат внедрения:" and is a text box with a browse button (three dots) to its right. The fourth field is labeled "Техника внедрения:" and is a dropdown menu currently showing "Code Injection". At the bottom center of the window is a button labeled "Внедрить". The window has a standard Windows-style title bar with a close button (X) in the top right corner.

Рисунок 12. Шаблон рабочего окна программы

«Идентификатор процесса» представляет собой поле ввода PID — числового значения, уникального для каждого процесса. Уникальный идентификатор процесса можно узнать в диспетчере задач.

«Полезная нагрузка» — поле ввода пути, содержащего файл, подлежащий внедрению в указанный в поле PID процесс выбранным способом.

«Результат внедрения» — поле ввода пути, содержащего файл, в который будет записан ход выполнения алгоритмов внедрения кода и их успешность.

Выпадающий список «Техника внедрения» представляет собой меню выбора одной из трёх представленных техник внедрения.

Кнопка «Внедрить» предназначена для запуска одного из модулей внедрения кода.

Кнопки « ... » предназначены для удобного выбора пути до файла с помощью диалогового окна.

### **Руководство пользователя.**

При запуске программы необходимо наличие файлов, содержащий внедряемый код, dll библиотек, содержащих исполняемый код.

Для проведения успешного внедрения кода необходимо придерживаться следующей инструкции: Записать в поле ввода PID число, представляющее собой идентификатор процесса, для проведения внедрения в данный процесс. В поле ввода «Полезная нагрузка» ввести путь до файла, который необходимо внедрить в процесс. В поле ввода «Результат» ввести путь до файла, после выполнения внедрения в данный файл будет помещена информация о проведённом внедрении. По завершении заполнения данных, необходимо нажать кнопку «Внедрить», дождаться выполнения алгоритмов внедрения. По завершении внедрения будет открыт текстовый файл, представляющий отчёт о произведённом внедрении.

### **Руководство для программиста.**

Данное руководство предназначено для пользователей, занимающихся разработкой специальных файлов для внедрения их в сторонний процесс, используя данную автоматизированную систему.

Требования к содержанию файла, предназначенного для внедрения в сторонний процесс с использованием техники внедрения Code Injection: файл должен быть заполнен машинными инструкциями, сгенерированными компилятором языка ассемблер, реализующими тот или иной алгоритм. Файл



#### 4.3 Экспериментальные данные тестирования.

В ходе выполнения тестирования автоматизированной системы были проведены тестовые запуски модулей изучена их работа по следующим критериям:

1. Скрытность — степень обнаружения следов программы при использовании различных техник внедрения.
2. Размер — сравнения размеров внедрённого кода различными модулями внедрения кода в сторонние процессы.
3. Живучесть — степень работоспособности модулей программы в условиях работы антивирусной программы.
4. Сложность в реализации.

На основе проведённых тестов была составлена характеристика каждой техники внедрения по данным критериям. Все сопутствующие результаты исследования отображены в виде скриншотов в приложениях.

##### Code Injection.

Техника внедрения по уровню скрытности имеет самый высокий уровень. Внедрённый код занимает незначительное количество памяти, что делает его менее заметным для отладчиков программ. Размер внедряемого кода достигает не более 1.5 килобайт, поэтому такой код более скрытный, чем аналогичный, с применением других техник внедрения. Живучесть модуля остаётся на высоком уровне, антивирусная программа не обнаруживает модуль внедрения кода в сторонний процесс как вредоносный файл, также антивирус не обнаруживает вредоносного кода в файле, предназначенного для внедрения кода, однако при загрузке кода в память стороннего процесса, антивирусная программа обнаруживает исполнение вредоносного кода в случае, если код содержит сигнатуры известных вирусов. Программа, в которую было произведено внедрение, останавливается и удаляется с жёсткого диска. В случае отсутствия во внедрённом коде сигнатур вирусов, код работает без сбоев. Сложность в реализации остаётся на высоком уровне. Причиной такой

сложности является необходимость написания кода на языке ассемблера, что сильно замедляет процесс разработки подобных файлов для внедрения. Процесс можно упростить путём использования готовых кодов или сгенерированных автоматически специальными программами.

#### Dll Injection.

Техника внедрения со стороны скрытности показывает высокий уровень обнаружения в программе, содержащей данный код. Кроме того загруженная библиотека регистрируется в системе, по этой причине вероятность обнаружения кода высокая. Размер внедрённого кода варьируется от 50 килобайт до нескольких мегабайт, что делает внедрённый код легко обнаруживаемым анализаторами исполняемых файлов и отладчиками. Живучесть внедряемых исполняемых файлов минимальная, при обнаружении dll с кодом на жёстком диске антивирусная программа удалит его. Модуль, внедряющий код в сторонний процесс при сканировании антивирусом не удаляется, но помечается как потенциально опасный объект. Для исполняемых файлов dll сложность в реализации минимальная, достаточно написать несколько строк для полноценного функционирования кода при внедрении в сторонний процесс.

#### Reflective DLL Injection.

Техника внедрения имеет особенность загрузки в сторонний процесс библиотек. По критерии скрытности данная техника уступает всем известным техникам внедрения за счёт огромного количества кода, внедряемого в сторонний процесс. Преимуществом данной техники служит отсутствие регистрации библиотеки в системе, из-за чего обнаружить библиотеку с использованием сторонних программ件 невозможно. Живучесть модуля в условиях работы антивирусной программы низкая. Живучесть внедряемых файлов такая же низкая. Простое сканирование антивирусом приводит к удалению компонентов. Сложность реализации высокая при отсутствии шаблона кода библиотеки для внедрения.

## **Заключение**

В ходе курсовой работы были реализованы модули, предназначенные для внедрения кода в сторонние процессы. Для автоматизации внедрения кода была реализована подсистема графического интерфейса.

В ходе реализации программы были использованы разнообразные API-функции работы с операционной системой. Получены знания их принципов работы, навыки использования их на практике разработки автоматизированной системы внедрения кода в сторонние процессы.

При проведении испытаний автоматизированной системы было проведено сравнение, в котором модуль Code Injector показал высокие показатели по скрытности, по сравнению с остальными модулями, а также минимальный размер кода, использующегося для внедрения с сторонний процесс. Однако написание подобного кода является сложной в реализации задачей, требующей знаний языка ассемблера и внутренней организации памяти процессов.

Модуль, реализующий технику DLL Injecіon показал хорошую работоспособность при условии выключенной антивирусной программы. При включении антивируса внедряемые файлы удаляются. Написание кода для внедрения не представляет трудностей.

Модуль, реализующий технику Reflective DLL Injection показал самую высокую степень скрытности в операционной системе. Техника написания файла с внедряемым кодом сложна.

### **Список используемых источников**

1. Алексеев П.П., Корш А.П., Прокди Р.Г., Антивирусы. – М.: Наука и техника, 2020, с. 80 - 100
2. Герберт Ш., С++ Шаг за шагом. - М.: ЭКОМ Паблишерз, 2017, с. 502
3. Девендрович В. Р., Справочник по функциям Win32 API, Горячая Линия. - Телеком, 2020, с. 132
4. Керниган Брайан, Ритчи Деннис, - 2-е. изд. Язык программирования С. - Вильяммс, 2017, с. 54
5. Крис Х. Паппас, Уильям Х. Мюррей III, Отладка в С++. Руководство для разработчиков. - Бином-Пресс, 2019, с.91
6. Литвиненко Н. А., Технология программирования на С++ Win32 API-приложения. - БХВ, 2019, с. 54
7. Леонтьев В. П., Новейшая энциклопедия компьютера. - Олма-Пресс, 2011
8. Лоспинозо Джош, С++ Для профи. - СПб.: Питер, 2021, с. 37
9. Макконнелл С., Совершенный код. - Санкт-Петербург: БХВ-Петербург, 2017, с. 78
10. Михайлов А. В., Компьютерные вирусы и борьба с ними. — Санкт-Петербург: Диалог-МИФИ, 2018, с. 87
11. Меджуи М., Митра Р., Непрерывное развитие API. Правильные решения в изменчивом технологическом ландшафте. - СПб.: Питер, 2020, с. 43
12. Побегайло А. П., Системное программирование в Windows. - СПб.: БХВ-Петербург 2019, с. 60 - 94
13. Прата Стивен, 6-е. изд. Язык программирования С++. Лекции и упражнения. - Саратов: Диалектика-Вильямс, 2018, с. 69
14. Рихтер Джеффри, Назар Кристоф, Windows via C/C++. Программирование на языке Visual C++. - Русская Редакция, Питер, 2018
15. Саймон Р., Microsoft Windows API. Справочник системного программиста. - М.: Диасофт, 2017, с. 56



- 16.Саак А.Э., Информационные технологии управления: Учебник для вузов - СПб.: Питер, 2010, с. 68
- 17.Страуструп Бьерн, Дизайн и эволюция C++. - СПб.: ДМК Пресс, Питер, 2014, с. 12
- 18.Стенли Б. Липпман, Жози Лажойе, Барбара Э. Му, Язык программирования C++. Базовый курс. - Диалектика-Вильямс, 2018
- 19.Ташков П. А., Защита компьютера на 100%: сбои, ошибки и вирусы. – СПб.: «Питер», 2010, с. 40 - 56
- 20.Цирлов В. Л., Основы информационной безопасности: краткий курс. - Ростов н/Д: Феникс, 2008, с. 70 - 102
- 21.Щупак Ю. А., Win32 API. Эффективная разработка приложений. - СПб.: Питер, 2018, с. 60
- 22.Эпплман Дан., Win32 API и Visual Basic. Для профессионалов. - СПб.: Питер, 2019. с. 99

## Приложения

### Перечень принятых сокращений

АС	Автоматизированная система
GUI	Графический интерфейс пользователя
СВКСП	Система внедрения кода в сторонний процесс
ВКСП	Внедрение кода в сторонний процесс