1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «Мониторинг событий в процессе»
2. по дисциплине «технологии разработки современного программного обеспечения»
3. Выполнил
4. студент гр. 5131001/20503 Черникова В. М.

<*подпись*>

1. Преподаватель
2. Завадский Е. В.

<*подпись*>

1. Санкт-Петербург
2. 2024

# ЦЕЛЬ РАБОТЫ

Целью работы – изучение и практическое освоение техники внедрения собственного кода в целевой процесс для перехвата вызовов функций и изменения его поведение.

# ЗАДАЧИ

В рамках выполнения лабораторной работы необходимо разработать программу и dll-библиотеку, которые обеспечивают два режима работы:

1. Отслеживание вызов указанных заранее функций. Каждый вызов функции в целевом процессе должен сопровождаться печатью сообщения на консоль разрабатываемой программы. Распечатанное сообщение должно содержать имя функции и метку времени.

2. Изменение поведение функции FindFirstFile, FindNextFile, CreateFile таким образом, что для целевого процесса пропадает указанный заранее файл. Для изменения поведения достаточно изменять значения, возвращаемые этими функциями в соответствии с определенными входными параметрами.

# Ход работы

Работа состояла из нескольких этапов:

1. Первый этап:

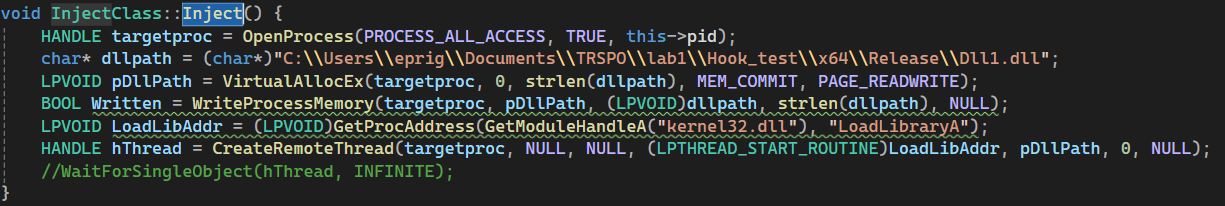
- написание тестовой программы, использующей API-функцию библиотеки kernel32.dll, например CloseHandle, и её сборка; выполнение отладки 64-битного процесса программы для определения трассы вызова выбранной API-функции; определение изменений, которые необходимо внести в ассемблерный код для осуществления перехвата вызова выбранной APIфункции.

1. Второй этап (занятие №2): разработка в рамках тестовой программы класса HookPatch, предоставляющего методы для внесения и отказа определённых ранее изменений, которые необходимо внести в ассемблерный код для осуществления перехвата вызова выбранной API-функции; разработка в рамках тестовой программы функции логирования, на которую будет передано управления при вызове выбранной API-функции; тестирование разработанного класса для перехвата выбранной API-функции тестовой программы.
2. Третий этап: собрать «пустую» dll-бибилиотеку; разработать программу, которая обеспечивает загрузку собранной ранее dll-библиотеки в целевой процесс. продумать протокол передачи режима функционирования и иных аргументов от программы к загруженной в целевой процесс dllбиблиотеке и сообщений мониторинга в обратную сторону. 4. Четвертый этап (занятие №4): реализация интерфейса командной строки программы; реализация ранее определённого протокола взаимодействия компонентов; интеграция разработанного ранее класс HookPatch в dllбиблиотеку; тестирование разработанных компонентов.

Итак, в результате прохождения этих шагов, были получены следующие результаты.

Для того, чтобы отслеживать вызовы функций из kernel32, в целевой процесс будет загружаться dll, которая будет патчить инструкции по адресу вызова функции.

Для загрузки своей dll в сторонний процесс была написана программа Injector, которая записывает в целевой процесс путь к dll, а затем создает отдельный поток, который подгрузит dll в процесс с помощью функции LoadLibraryA. Код можно видеть на рисунке 1.



*Рисунок 1 - Код инжектора*

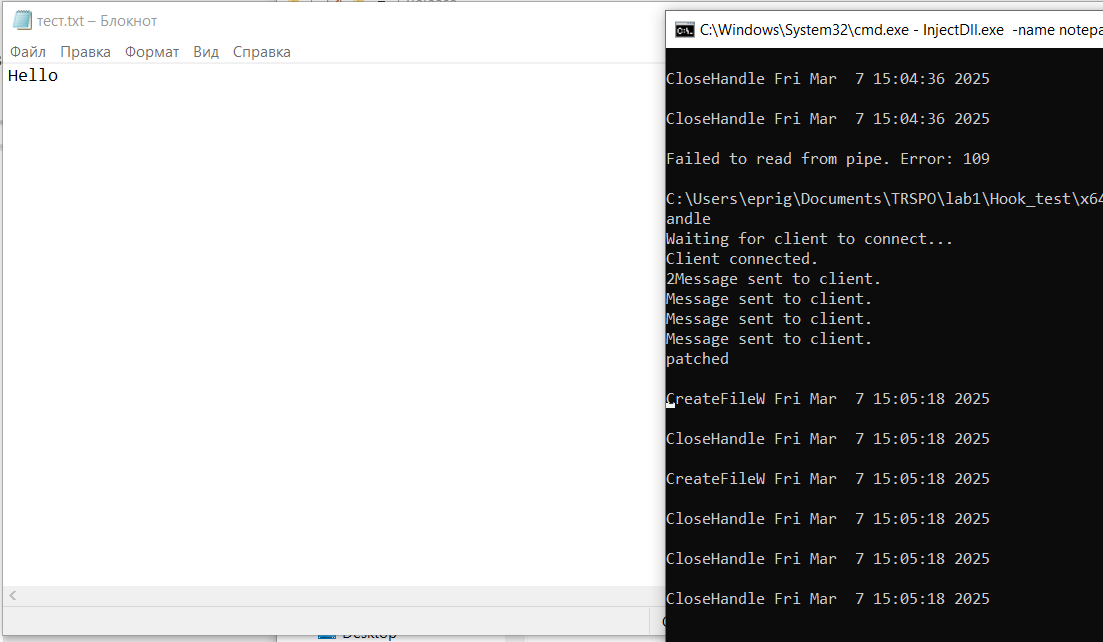
Для общения dll и программы-инжектора в них были прописаны классы PiperCl и Piper, которые создают именованный канал и пересылают по нему сообщения. С помощью него в dll передается информация, какие функции нужно отслеживать и какой файл необходимо скрыть.

Для патчинга был разработан класс HookPatch, он хранит функции, которые были пропатченны, а также их адреса. Функция принимает на вход адрес функции мониторинга и название функции из kernel32. Далее находит в kernel32 адрес функции и заменяет байты по этому адресу на переход на мою функцию, которая сохраняет все аргументы на стеке, затем переключается на функцию мониторинга, после чего достает аргументы и переносится на оригинальную функцию.

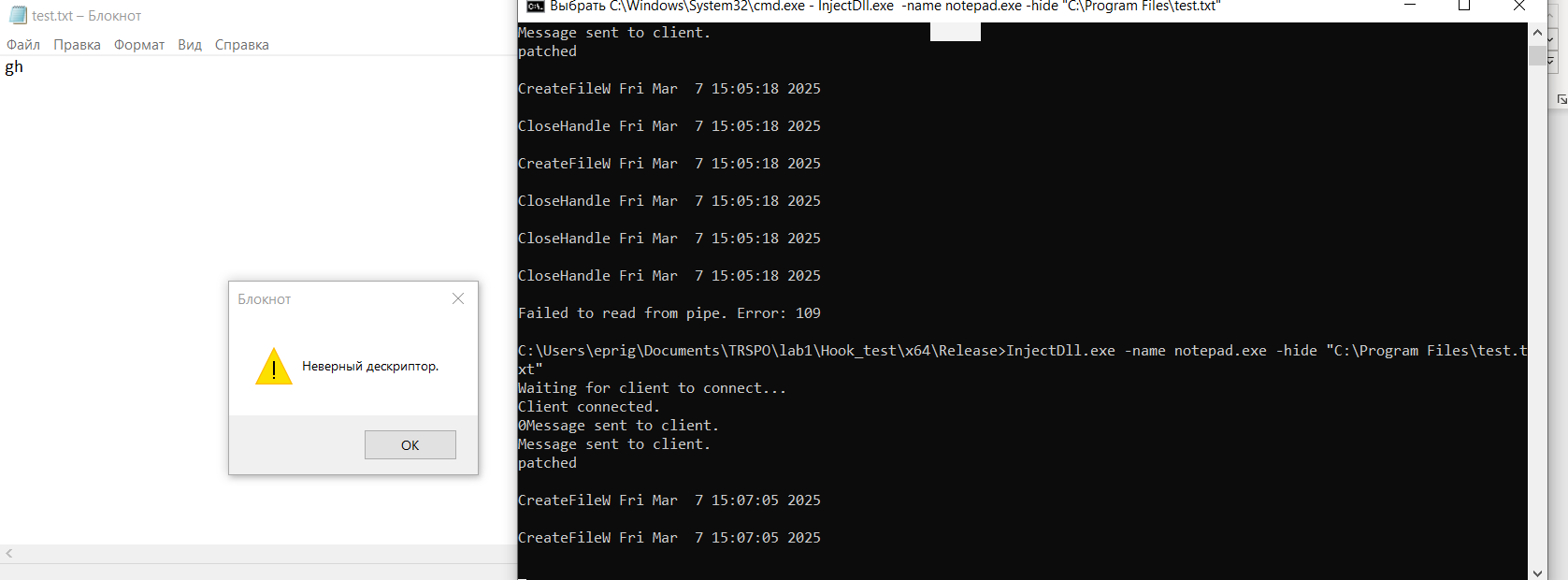
Также для фиксирования, какая именно функция была вызвана, я кладу на стек номер данной функции, который затем берется со стека и подается на вход в функцию мониторинга.

Для скрытия файла была написана отдельная функция, которая помимо мониторинга включает сравнение входной строки и в зависимости от результатов сравнения переключается или не переключается на оригинальную функцию.

Протестируем программу на notepad.exe. Тесты представлены на рисунках 2 и 3.



*Рисунок 2 - Мниторинг вызовов*



*Рисунок 3 - Сокрытие файла*

# Вывод

В результате лабораторной работы было изучено, как внедрять свою dll в чужой процесс. Также были получены знания о работе с именованными каналами и winapi.