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

**ЛАБОРАТОРНАЯ РАБОТА № 1**

«**Мониторинг событий в процессе**»

1. по дисциплине «Технологии разработки современного программного обеспечения»
2. Выполнил
3. студент гр. 4851003/10801 Л. А. Богданова

<*подпись*>

1. Преподаватель
   * 1. Г.С. Кубрин

<*подпись*>

1. Санкт-Петербург

2024

1. **ЦЕЛЬ РАБОТЫ**

Необходимо написать программу, которая осуществляет мониторинг вызова функций в процессе.

1. **ФОРМУЛИРОВКА ЗАДАНИЯ**

Программа должна использовать технику Hook&Inject, которая заключается в том, чтобы заставить целевой процесс осуществить загрузку указанной DLL, которая, в свою очередь выполняет изменение таблиц с функциями DLL, осуществляя их перехват. При использовании такой техники есть возможность осуществлять мониторинг вызова заданных функций, а также существует возможность подменять значения результата работы функции, меняя поведение целевого процесса.

В лабораторной требуется написать программу и dll библиотеку, которые позволяют:

1. Отслеживать 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. Изменять поведение функции FindFirstFile, FindNextFile, CreateFile таким образом, что для процесса пропадает указанный заранее файл. Для изменения поведения достаточно изменять значения, возвращаемые этими функциями в соответствии с определенными входными параметрами. Должно работать для cmd.exe в 32-битной версии.

При написании программы должны быть выполнены несколько условий:

1. Программа должна компилироваться Visual Studio.

2. Программа требует привилегий администратора для своей работы.

3. Программа должна обладать интерфейсом командной строки.

4. Программа должна предлагать следующие действия:

a. –pid < Target\_Process\_PID> – целевой процесс выбирается по значению PID.

b. –name <Target\_Process\_Name> – целевой процесс выбирается по имени.

c. –func <Function\_Name> – имя функции, вызов которой требуется отследить.

d. –hide <File\_Name> – имя файла, который надо скрыть от процесса.

5. Считается, что доступны следующие сценарии запуска программы:

a. Monitor.exe –pid 123 –func CreateFile

b. Monitor.exe –name explorer.exe –func CreateFile

c. Monitor.exe –name explorer.exe –hide “C:\hello.txt”

1. **ХОД РАБОТЫ**

**3.1 Теоретические сведения**

Термин Hooking (или же перехват) представляет собой фундаментальную технику получения контроля над выполнением определенного фрагмента кода, отслеживания в нем вызовов определенных функций, либо изменения поведения этих функций путем их подмены с помощью загрузки указанной Dll.

Внедрение (injection) – это технология, позволяющая выполнять другой код (полезную нагрузку) в адресном пространстве уже запущенного процесса. Dll-инъекции используются внешними программами, чтобы переопределить поведение программы-жертвы.

Программа, используемая для внедрения измененного кода в процесс, называется Dll-инжектором.

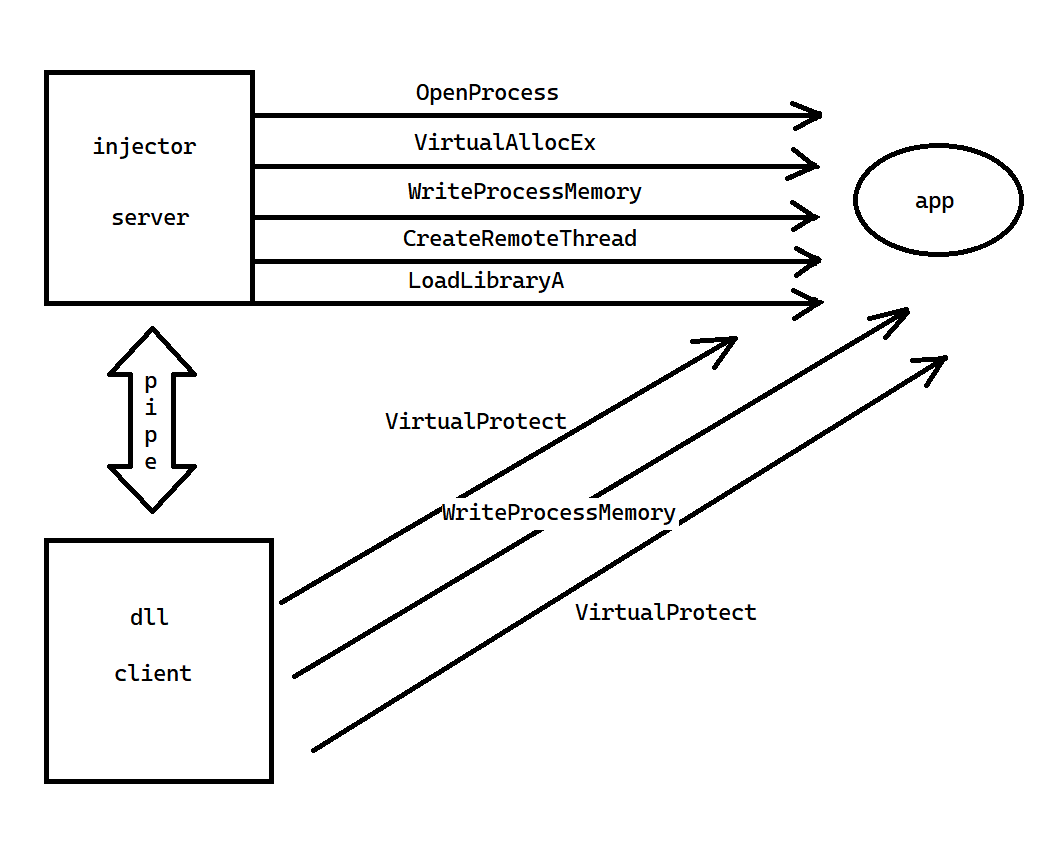
Таким образом, техника Hook&Inject заключается в том, чтобы осуществить загрузку указанной Dll в целевой процесс, которая, в свою очередь, выполняет изменение функций Dll, изменяя тем самым поведение целевого процесса. Схема работы Dll-инъекции приведена на рисунке 1.

Рисунок 1 – Схема работы dll-инъекции.

Для создания программы использующую технику Hook&Inject необходимо было разработать инжектор, а также саму dll.

Программа-инжектор необходима для считывания аргументов командной строки – определения целевого процесса, скрываемого от него файла и перехватываемой от него функции.

Перед началом работы инжектор выдает себе право на отладку других процессов, что необходимо для записи в адресное пространство другого процесса. Таким образом, инжектор требует прав администратора.

Сначала инжектор производит парсинг входных аргументов – выделяет имя/PID целевого процесса, в который нужно внедрить разработанную DLL.

Далее инжектор создает именованный канал. Общение между инжектором и dll осуществляется при помощи именованного канала - Pipe. Инжектор создает данный канал, а dll открывает (другими словами, подключается) канал.

После чего инжектор выполняет следующие действия:

* Функцией OpenProcess() получает дескриптор необходимого процесса.
* При помощи функции GetModuleHandleW(L"kernel32.dll") получает адрес стандартной библиотеки.
* Далее функцией GetProcAddress() получает адрес функции LoadLibraryA, которая отвечает за загрузку модулей в адресной пространство процесса.
* После чего при помощи функции VirtualAllocEx() выделяет память в процессе.
* Функцией WriteProcessMemory() записывает путь к разработанной dll в область памяти процесса, которая была выделена на прошлом шаге.
* Наконец при помощи функции CreateRemoteThread() создает поток, который запускается в виртуальном адресном пространстве процесса. В качестве аргументов данная функция принимает дескриптор процесса; функцию, код которой исполняется потоком и обозначает начальный адрес потока в удаленном процессе; указатель на аргумент с которым будет запускаться LoadLibraryA (данные аргумент находится в выделенной в адресном пространстве памяти процесса, это необходимо для того, чтобы создаваемый поток запустился в адресном пространстве необходимого процесса).

Для установки хуков использовалась библиотека библиотеки Microsoft Detours. Библиотека Detours перехватывает планируемые функции, переписывая их на уровне двоичного кода внутри процесса.

Разработанная DLL может реализовывать два алгоритма, в зависимости от аргументов, полученных через pipe от инжектора:

1. **Алгоритм перехвата всех вызовов указанной функции:**
   1. Поиск адреса указанной функции в стандартной библиотеке kerel32.dll (*DetourFindFunction())*
   2. Объявляем начало транзакции и выполняем захват (*DetourTransactionBegin()* и *DetourUpdateThread()*)
   3. Осуществляем подмену оригинальной функции на свою собственную (*DetourAttach()*)
2. **Алгоритм сокрытия указанного файла**
   1. Объявляем начало транзакции и выполняем захват (*DetourTransactionBegin()* и *DetourUpdateThread()*)
   2. Осуществляем подмену оригинальной функции на свои собственные (*DetourAttach()*)

Выполняем это для следующих функций: CreateFileA, CreateFileW, FindFirstFileA, FindFirstFileW, FindNextFileA, FindNextFileW, FindFirstFileExW, FindFirstFileExA.

Сначала вызывается оригинальная функция, затем если имя файла который нужно скрыть совпало с тем, которое вернула функция, то для функций типа CreateFile… возвращается INVALID\_HANDLE\_VALUE, а для функций типа Find...File… возвращается имя следующего файла, например, снова вызывая функцию FindNextFile() и возвращая вновь полученное имя файла.

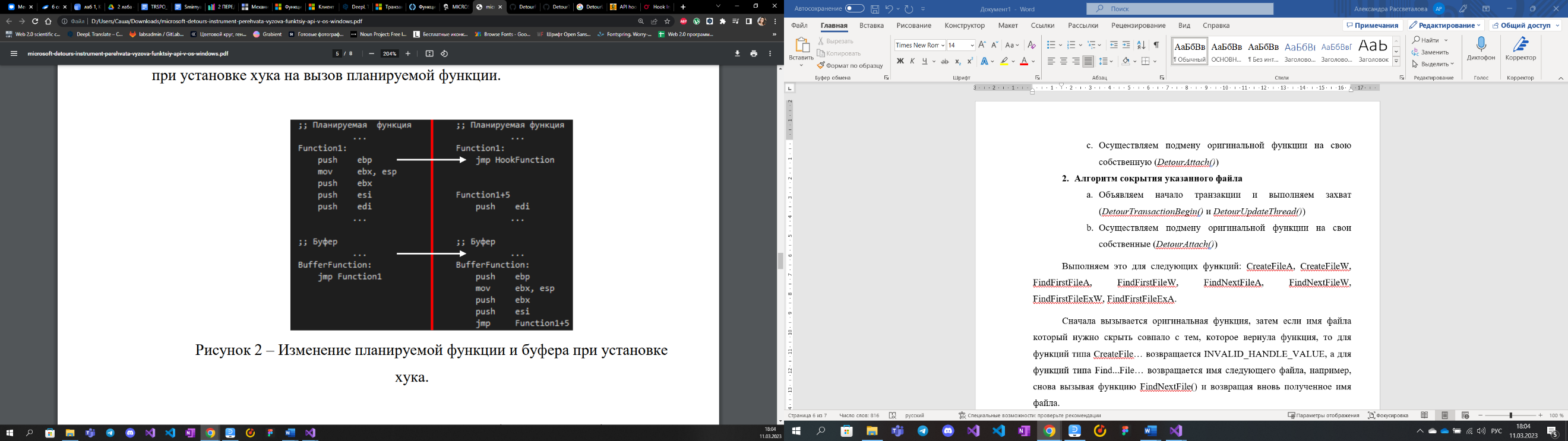
Происходит это следующим образом: находиться адрес функции в стандартной (функция DetourFindFunction()), после чего определяется другая функция, которая будет запускаться вместо оригинальной. Далее происходит связывание оригинальной и новой функции – как минимум первые 5 байт оригинальной функции копируются в другую страницу (трамплин) с целью освободить место для записи инструкции безусловного перехода на новую функцию. Для правильного перехвата оригинальная функция и хук-функция должны иметь одинаковую сигнатуру вызова, включая количество аргументов и соглашение о вызове. Использование одного и того же соглашения о вызове гарантирует, что регистры будут должным образом сохранены и что стек будет должным образом выровнен между хук и оригинальной функциями. Принцип работы такой схемы показан на рисунке 2.

Рисунок 2– Изменение оригинальной (планируемой) функции и буфера при установке хука.

В Detours установка и снятие хука осуществляется с помощью транзакций. Для того чтобы открыть транзакцию и зафиксировать её существуют такие функции, как DetourTransactionBegin() и DetourTransactionCommit(). Перехват планируемой к запуску функции из исходной осуществляется с помощью вызова функции DetourAttach(). Её вызов должен осуществляться после открытия транзакции и до её фиксации. Данная функция принимает на вход 2 аргумента: указатель на планируемую функцию и указатель на хук-функцию.

Для того чтобы это сделать необходимо сразу после функции создания транзакции DetourTransactionBegin(), внутри неё сразу вызвать функцию DetourUpdateThread(), которая на вход принимает всего одно значение, а именно дескриптор потока, который ожидает данную транзакцию .

При попытке вызвать планируемую функцию из исходной, её вызов перехватывается и перенаправляется на хук-функцию, откуда в случае необходимости она может быть запущена. Ответственность за верное копирование аргументов при вызове планируемой функции через функцию с инструкциями из буфера лежит на хук-функции.

В случае отслеживания вызовов функции используется ассемблерная вставка, поскольку мы подменяем работу функции, поэтому необходимо сохранить, а затем восстановить значения регистров, для продолжения корректной работы программы. В случае сокрытия файла происходит подмена возвращаемого значения, поэтому сохранение стека не требуется.

**3.2 Тестирование**

**3.2.1 Перехват вызовов функции**

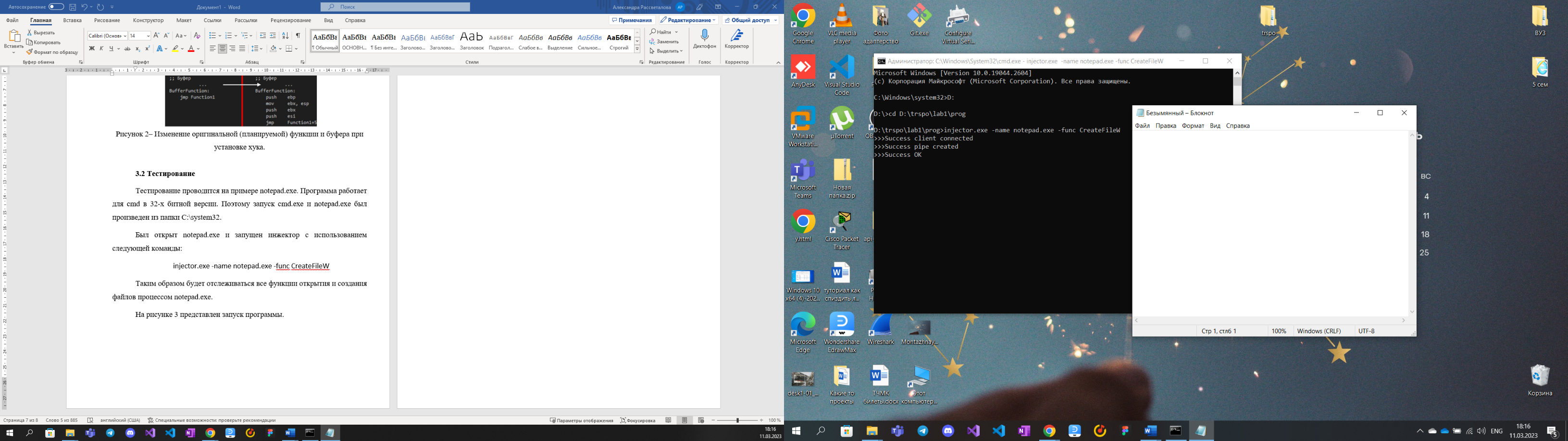
Тестирование проводится на примере notepad.exe. Программа работает для cmd в 32-х битной версии. Поэтому запуск cmd.exe и notepad.exe был произведен из папки C:\system32.

Был открыт notepad.exe и запущен инжектор с использованием следующей команды:

injector.exe -name notepad.exe -func CreateFileW

Таким образом будет отслеживаться все функции открытия и создания файлов процессом notepad.exe.

На рисунке 3 представлен запуск программы.

Рисунок 3 – Запуск программы

Изображение выглядит как текст, монитор, внутренний, компьютер

Автоматически созданное описаниеДалее была произведена попытка открыть файл Hello.txt используя интерфейс блокнота, тем был инициирован вызов функции CreateFileW(), рисунок 4.

Рисунок 4 – Вызовы функции CreateFileW()

Первый вызов функции был инициирован при нажатии Файл –> Открыть. Второй вызов функции был инициирован, при выборе файла .txt и нажатии на «Открыть» в диалоговом окне.

**3.2.2 Сокрытие заданного файла**

Для демонстрации работы разработанной программы в режиме сокрытия заданного файла были открыты две командные строки от имени администратора и произведен запуск командой:

injector.exe -pid 412 -hide D:\trspo\new.txt

Изображение выглядит как текст

Автоматически созданное описаниеТаким образом мы скрываем от командной строки текстовый файл. Содержимое папки было просмотрено до и после запуска программы, можно увидеть, что после запуска файл скрыт от командной строки, рисунок 5.

Рисунок 5 – Сокрытие файла от командной строки

Далее от командной строки было скрыто приложение notepad.exe. После чего невозможно открыть блокнот из командной строки, но ели открыть новую командную строку, то из нее блокнот будет открыт, так как хук производился для одной командной строки, рисунок 6.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Сокрытие notepad.exe от командной строки

# **ВЫВОД**

В результате выполнения лабораторной работы было изучено понятие перехвата вызовов функций в ОС Windows, а также техники внедрения динамических библиотек в запущенный процесс. Для разработки приложения были использованы функции WinAPI и функции библиотеки Detours. Для реализации взаимодействия инжектора с DLL были задействованы структуры и функции, определяющие общение клиента и сервера через именованный канал(pipe). В итоге были разработаны программа инжектор и DLL-библиотека для ОС Windows, реализующие мониторинг событий в процессе и изменение поведения его функций.