1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт прикладной математики и механики

**Высшая школа кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

1. «**Мониторинг событий в процессе**»
2. по дисциплине «Технологии разработки современного программного обеспечения»
3. Выполнил
4. студент гр. 3651003/70801 Гасанов Э.А.

<*подпись*>

1. Преподаватель
2. доцент Никольский А.В.

<*подпись*>

1. Санкт-Петербург
2. 2020

# Формулировка задания

Написать программу, которая осуществляет мониторинг вызова функций в процессе. 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 32х 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:
   1. –pid < Target\_Process\_PID> – целевой процесс выбирается по значению PID.
   2. –name <Target\_Process\_Name> – целевой процесс выбирается по имени.
   3. –func <Function\_Name> – имя функции, вызов которой требуется отследить.
   4. –hide <File\_Name> – имя файла, который надо скрыть от процесса.
5. Считается, что доступны следующие сценарии запуска программы:
   1. Monitor.exe –pid 123 –func CreateFile
   2. Monitor.exe –name explorer.exe –func CreateFile
   3. Monitor.exe –name explorer.exe –hide “C:\hello.txt”

# Проделанная работа

В ходе выполнения лабораторной работы были созданы инжектор и dll для мониторинга за вызовами в целевом процессе. Инжектор работает в соответствии с приведенным ниже алгоритмом:

1. Выделяется блок памяти в адресном пространстве удаленного процесса через VirtualAllocEx.

2. Вызвается WriteProcessMemory, куда копируется строка с полным именем файла DLL в блок памяти, выделенный в п.1.

3. Используя GetProcAddress, получим истинный адрес функции LoadLibraryA или LoadLibraryW внутри Kernel32.dll.

4. Вызвав CreateRemoteThread, создаётся поток в удаленном процессе, который вызовет соответствующую функцию LoadLibrary, передав ей адрес блока памяти, выделенного в п. 1. На этом этапе DLL внедрена в удаленный процесс, а ее функция DllMain получила уведомление DLL\_PROCESS\_AТТАСН и может приступить к выполнению нужного кода.

Для очистки после завершения удаленного потока потребуется несколько дополнительных операций.

5. Вызовом VirtualFreeEx освободается блок памяти, выделенный в п. 1.

6. С помощью GetProcAddress определим истинный адрес функции FreeLibrary внутри Kernel32.dll.

7. Используя Create RemoteThread, создаём в удаленном процессе поток, который вызовет FreeLibrary с передачей HMODULE внедренной DLL.

Для передачи параметров dll мы используем именованный канал. Когда dll уже будет загружена в адресное пространство, опять же с помощью именованного канала будут переданы сообщения мониторинга за вызванными функциями.

Алгоритм работы dll представлен ниже.

# Алгоритм работы мониторинга вызовов функции

Используя библиотеку detours, мы совершаем следующие действия в dll:

Перезаписываются первые нескольких байт оригинальной функции(5 байт) командой безусловного перехода (jmp) к функции обхода (detour). Эта функция вызывает функцию trampoline, содержащую перезаписанные байты оригинальной целевой функции и вызывающую целевую функцию. Далее целевая функция возвращает управление функции detour, а та, в свою очередь, - функции источника.

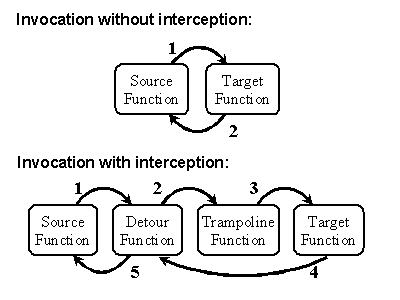


Рисунок 1, иллюстрация работы detours

# Алгоритм работы функции сокрытия заданного файла

В ходе выполнения работы выполнялась вторая часть задания - изменять поведение функции FindFirstFile, FindNextFile, CreateFile таким образом, что для процесса пропадает указанный заранее файл.

Проделанные изменения:

* FindFirstFile: ищет каталог файла или подкаталог, название которого соответствует указанному имени файла. Для сокрытия файла от этой функции при совпадении имен файлов возвращаем INVALID\_HANDLE\_VALUE.
* FindNextFile: продолжает поиск файла из предыдущего вызова функции FindFirstFile. Для сокрытия файла от этой функции при совпадении имен файлов возвращаем 0.
* CreateFile: создает или открывает каталог. Функция возвращает дескриптор, который может быть использован для доступа к объекту. Для сокрытия файла от этой функции при совпадении имен файлов возвращаем INVALID\_HANDLE\_VALUE.

# Вывод

В результате выполнения данной работы был изучен механизм Hook&Inject. Написаны программа и dll-библиотека, позволяющие осуществлять мониторинг функций, вызываемых заданным процессом.