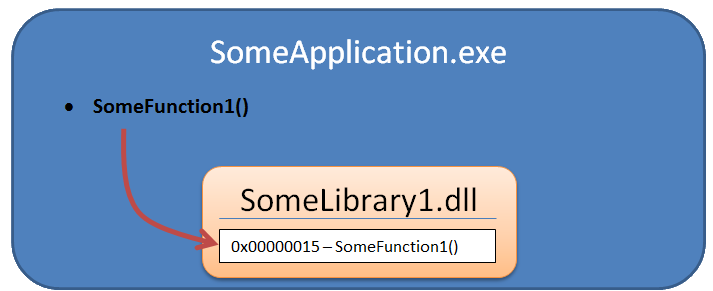
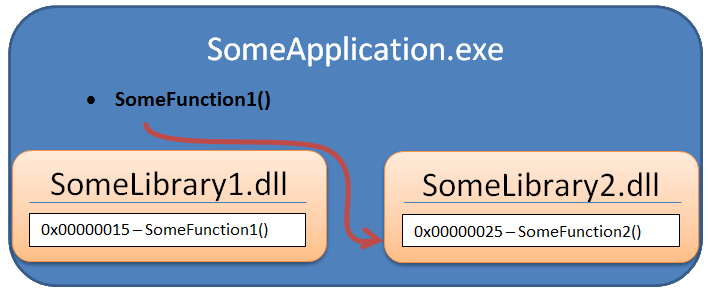
Хуки — это просто

Хуки — это технология перехвата вызовов функций в чужих процессах. Хуки, как и любая достаточно мощная технология, могут быть использованы как в благих целях (снифферы, аудио\видеограбберы, расширения функционала закрытого ПО, логирование, багфиксинг) так и со злым умыслом (трояны, кряки, кейлоггеры). О хуках уже не раз писали и [на Хабре](http://habrahabr.ru/post/90377/) и [не на Хабре](http://wasm.ru/series.php?sid=8). Но вот в чём беда — почему-то каждая статья о хуках буквально со второго абзаца начинает рассказывать о «таблице виртуальных функций», «архитектуре памяти» и предлагает к изучению огромные блоки ассемблерного кода. Известно, что каждая формула в тексте снижает количество читателей вдвое, а уж такие вещи — так и вовсе вчетверо. Поэтому нужна статья, которая расскажет о хуках просто. Под катом нет ассемблера, нет сложных терминов и буквально два десятка строк очень простого кода на С++. Если вы давно хотели изучить хуки, но не знали с чего начать — начните с этой статьи.

Реальная задача

Для лучшего понимания того, что мы делаем — поставим себе какую-нибудь реальную задачу. Давайте, например сделаем так, чтобы браузер Firefox при заходе на Хабр писал в своём заголовке **«Привет, Хабр!»** вместо того, что там пишется сейчас (а сейчас там пришется **"\*\*\* / Хабрахабр — Mozilla Firefox"**, где \*\*\* — меняется в зависимости от раздела). Да, я знаю, что это можно сделать правкой исходников Firefox, браузерными плагинами, юзерскриптами и еще десятком способов. Но мы в учебных целях сделаем это хуками.

Совсем чуть-чуть теории

Когда Вы запускаете любое приложение — операционная система создаёт его процесс. Грубо говоря, exe-файл копируется в память, далее определяется какие именно библиотеки (dll-файлы) ему нужны для работы (эта информация записана в начале каждого exe-файла), эти библиотеки ищутся (в папке с программой и в системных папках) и загружаются в память процесса. Потом определяется, какие именно функции библиотек использует программа и где они находятся (в какой библиотеке и где именно в этой библиотеке). Строится табличка вида «функция SomeFunction1() — библиотека SomeLibrary1.dll — %адрес\_функции\_SomeFunction1()%». Когда программе понадобиться вызвать эту функцию — она найдет в своей памяти нужную библиотеку, отсчитает нужный адрес и передаст туда управление.  
  
  
Суть хукинга — заставить программу поверить, что нужная ей функция находится в другом месте.  
  
  
Делается это таким образом — мы пишем свою библиотеку SomeLibrary2.dll, в которой будет находится наша функция SomeFunction2(). Далее мы загружаем эту библиотеку в память чужого процесса (в ОС Windows есть специальная функция для этого) и изменяем ту самую табличку, о которой я писал чуть выше, так, чтобы теперь она содержала запись «функция SomeFunction1() — библиотека SomeLibrary2.dll — %адрес\_нашей\_функции\_SomeFunction2()%». Для того, чтобы понять, как вручную сделать всё описанное в этом абзаце, нужно знать весьма прилично всего — как устроена память в Windows, как вызываются функции, как им передаются аргументы и т.д. Это сложно. Ну на самом деле не очень, просто можно обойтись и без этого. Если вам это нужно — почитайте какую-нибудь продвинутую статью (а хоть бы из тех, что указаны в начале). Мы пойдем другим путем — используем готовую библиотеку Microsoft Detours, которая сделает всю грязную работу за нас.

Пару слов о Microsoft Detours

* Проста в изучении и использовании  
  https://habrastorage.org/getpro/habr/post_images/8c7/748/0eb/8c77480eb257567e7f3e80485d064f8d.png Весьма эффективна  
  https://habrastorage.org/getpro/habr/post_images/8c7/748/0eb/8c77480eb257567e7f3e80485d064f8d.png Хорошая документация  
  https://habrastorage.org/getpro/habr/post_images/8c7/748/0eb/8c77480eb257567e7f3e80485d064f8d.png Содержит много примеров в исходниках  
  https://habrastorage.org/getpro/habr/post_images/8c7/748/0eb/8c77480eb257567e7f3e80485d064f8d.png Разработана Microsoft — неплохо «дружит» с ОС  
  https://habrastorage.org/getpro/habr/post_images/8c7/748/0eb/8c77480eb257567e7f3e80485d064f8d.png **Бесплатна для** исследовательских целей и **некоммерческих проектов**  
  https://habrastorage.org/storage2/e07/f13/004/e07f13004e06a4d3ba4cbff009a8599c.png Не требует знания ассемблера  
    
  https://habrastorage.org/getpro/habr/post_images/ad2/135/039/ad2135039877b140f320d52c934d97cf.png Закрыта  
  https://habrastorage.org/getpro/habr/post_images/ad2/135/039/ad2135039877b140f320d52c934d97cf.png Стоит приличных денег для коммерческого использования или х64-архитектуры – 10000тыс.дол.США

**На просторах интернета есть также и бесплатные и \_очень\_ продвинутые аналоги detours**  
  
В целом, я бы посоветовал начинать изучение хуков именно с Detours — если это будет всего лишь вашим разовым развлечением, то этого вполне хватит, у вас быстро всё получится и вам понравится. Если же хуки понадобятся в серьёзном проекте — вы легко переключитесь на бесплатные и открытые (но чуть более сложные) библиотеки типа [mhook](http://codefromthe70s.org/mhook.aspx), купите Detours или напишете свой велосипед (для последних двух решений нужны весьма веские причины).  
О том где взять и как собрать Detours я писал вот [тут](http://habrahabr.ru/post/139344/).

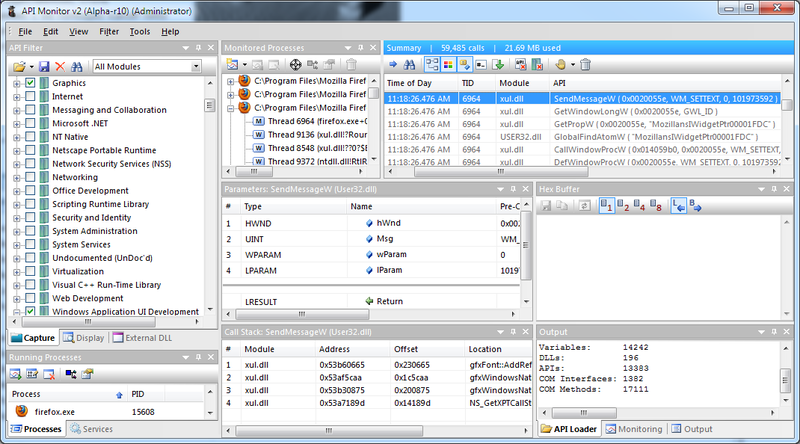
Хитрый план

1. Понять, на какую функцию ставить хук.
2. Сделать свою библиотеку с функцией, которая будет заменять исходную и делать нужные нам вещи.
3. Установить хук (загрузить библиотеку в память нужного процесса и переставить указатель на нужную нам функцию).
4. PROFIT!

Куда ставить хук

MSDN весьма ясно намекает нам, что заголовок окна можно установить функцией [SendMessage](http://msdn.microsoft.com/en-us/library/windows/desktop/ms644950%28v=vs.85%29.aspx) — при этом вторым параметром должно быть передано [WM\_SETTEXT](http://msdn.microsoft.com/en-us/library/windows/desktop/ms632644%28v=vs.85%29.aspx), а последним — сам текст. Но тут могут быть нюансы:

1. Вместо SendMessage может использоваться PostMessage или [что-то еще](http://msdn.microsoft.com/en-us/library/windows/desktop/ms632590%28v=vs.85%29.aspx)
2. SendMessage может быть вообще не функцией, а макросом, ссылающимся на другую функцию (в дальнейшем мы увидим, что так оно и есть)
3. Firefox, как некоторые кроссплатформенные приложения, может вообще не использовать функции Windows для рисования стандартных элементов окна, используя вместо этого какие-то собственные кросплатформенные элементы GUI (к счастью, это не так — но вдруг!)

Так что нужно всё хорошенько проверить. Нам поможет прекрасная бесплатная программа [API Monitor](http://www.rohitab.com/apimonitor). Она позволяет присоединиться к определенному процессу и подсмотреть, какие именно функции он вызывает и с какими параметрами. Вы, может быть, уже догадались — делает она это тоже с помощью хуков. Итак запускаем Firefox и API Monitor. Первым делом в API Monitor нужно указать фильтр — какую именно группу функций мы хотим мониторить. Если выберем вообще всё — исследуемая программа будет работать очень медленно (а может даже зависнет), выберем слишком мало — упустим нужное. Поэтому тут придётся думать и выбрать лишь ту группу, где потенциально могут находится функции работы с элементами GUI Windows. Давайте выберем группы Graphics и Windows Application UI Development а в панели Running Processes дважды кликнем по нашему Firefox. Начиная с этого момента API Monitor в панели справа будет показывать вызовы всех API-функций и их параметры.  
  
Переходим в Firefox, открываем Хабр, дожидаемся изменения заголовка на нужный и возвращаемся в Api Monitor чтобы остановить мониторинг. Скорее всего, вы будете удивлены количеством вызванных функций — их могут быть сотни тысяч буквально за несколько секунд мониторинга. А мы ведь еще и следим далеко не за всем. Да-да, это всё реально происходит внутри безобидного открытия всего одного сайта в браузере! А вы еще жалуетесь, что эта пара секунд — слишком долго. :)  
  
  
Найти нужную нам функцию поможет поиск по вкладке с результатами мониторинга. Вбиваем в поиск «WM\_SETTEXT» и убеждаемся, что действительно имеются вызовы функции SendMessageW с этим параметром — с высокой вероятностью это и есть установка заголовка окна. Обратите внимание на «W» в конце названия функции — оно означает, что используется её юникодная версия. Для установки хуков важно знать точное имя подменяемой функции и теперь мы его знаем.

Делаем свою библиотеку

**1.** Запускаем Visual Studio.  
**2.** Создаём новый проект: File->New->Project. Тип Visual C++ -> Win32 -> Win32 Project. В диалоге создания проекта указываем тип «Dll».  
**3.** Открываем файл dllmain.cpp и пишем туда вот такой код:  
  
#include <windows.h>

#include "C:\Program Files\Microsoft Research\Detours Express 3.0\src\detours.h"

LRESULT (WINAPI \* TrueSendMessageW)(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam) = SendMessageW;

\_\_declspec(dllexport) LRESULT WINAPI MySendMessageW(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam)

{

if (Msg == WM\_SETTEXT && wcsstr((LPCTSTR)lParam, L"/ Хабрахабр - Mozilla Firefox") != NULL)

return TrueSendMessageW(hWnd, Msg, wParam, (LPARAM)L"Привет, Хабр!");

return TrueSendMessageW(hWnd, Msg, wParam, lParam);

}

BOOL WINAPI DllMain(HINSTANCE hinst, DWORD dwReason, LPVOID reserved)

{

if (dwReason == DLL\_PROCESS\_ATTACH)

{

DetourRestoreAfterWith();

DetourTransactionBegin();

DetourUpdateThread(GetCurrentThread());

DetourAttach(&(PVOID&)TrueSendMessageW, MySendMessageW);

DetourTransactionCommit();

}

else if (dwReason == DLL\_PROCESS\_DETACH)

{

DetourTransactionBegin();

DetourUpdateThread(GetCurrentThread());

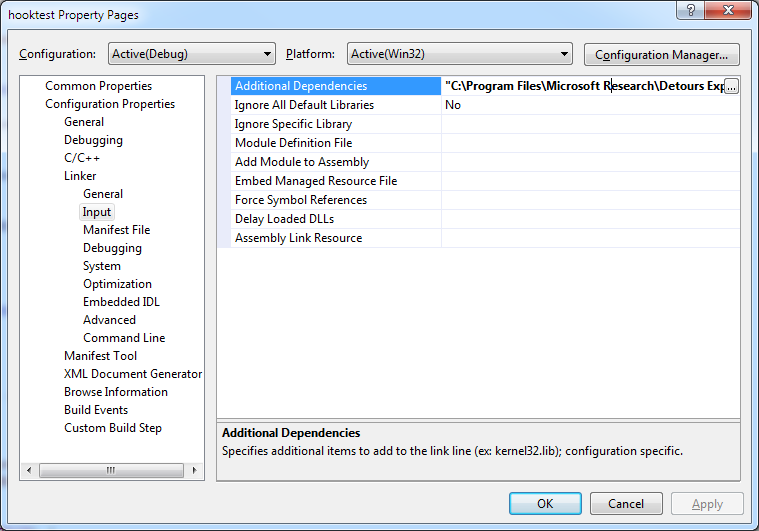
DetourDetach(&(PVOID&)TrueSendMessageW, MySendMessageW);

DetourTransactionCommit();

}

return TRUE;

}

**4.** Открываем свойства проекта и на вкладке настроек линкера добавляем в поле Additional Dependencies значение «C:\Program Files\Microsoft Research\Detours Express 3.0\lib.X86\detours.lib». Внимание, у вас путь может быть другой — смотря куда установили библиотеку Detours.  
  
  
  
**5.** Компилируем проект: Build -> Build Solution. На выходе получаем длл-ку (пусть будет называться hooktest.dll)  
  
Давайте разберем исходник. В начале мы подключаем заголовочные файлы Windows (чтобы пользоваться функцией SendMessageW) и Detours (чтобы иметь возможность ставить\снимать хуки).  
В сложной на первый взгляд строке №3 мы всего лишь сохраняем реальный указатель на функцию SendMessageW в переменную TrueSendMessageW. Это нам понадобиться для двух целей:

1. Для вызова настоящей функции SendMessageW из нашей «подделки».
2. Для восстановления указателя на реальную функцию в момент, когда мы захотим снять хук.

Далее идет наша поддельная функция MySendMessageW. Она предельно проста. Если попалось сообщение WM\_SETTEXT и в его тексте есть упоминание Хабра — заменяем его на своё. Иначе — работаем как прозрачный прокси. Обратите внимание на префикс **\_\_declspec(dllexport)** — он нужен чтобы этой функцией смогли воспользоваться другие процессы.  
  
Функция DllMain вызывается операционной системой в определенных случаях — например, в моменты аттача\детача библиотеки к процессу. Тут тоже всё просто. В момент аттача нам нужно установить хуки, в момент детача — снять. Библиотека Detour требует делать это транзакциями, и в этом есть смысл — представьте себе что будет, если сразу несколько желающих захотят поставить хуки в один процесс. Самое важное в этом коде это строка  
DetourAttach(&(PVOID&)TrueSendMessageW, MySendMessageW);

Именно она заставляет процесс «поверить» что теперь вместо настоящей функции SendMessageW нужно вызывать нашу MySendMessageW. Ради этой строки всё и затевалось. Если кому интересно, однажды я писал аналог этой функции вручную. С учетом всех возможных комбинаций типов функций и архитектур это заняло у меня несколько недель. Вы вот только что их сэкономили — поздравляю.

Устанавливаем хук

Microsoft Detours предлагает разные варианты установки хуков — мы воспользуемся самым простым. В комплекте примеров, которые идут с библиотекой, есть программа withdll.exe — она принимает в качестве параметров путь к приложению и библиотеку, которую нужно подгрузить в память этого приложения после его запуска. Запускаем всё это как-то вот так:  
withdll.exe -d:hooktest.dll "C:\Program Files\Mozilla Firefox\firefox.exe"

PROFIT!

Открываем Хабр:  
  
