

Программное обеспечение представляет собой фильтрующий драйвер, работающий в режиме ядра, который получает скан-коды[[1]](#footnote-1) нажатых и отпущенных клавиш, преобразует их в символы и записывает в файл.

Языком программирования, на котором написано ПО, является C.

Среда разработки: Visual Studio 2015

Поддерживаемые платформа: MS Windows 10 (x64)

Программное обеспечение, необходимое для работы:

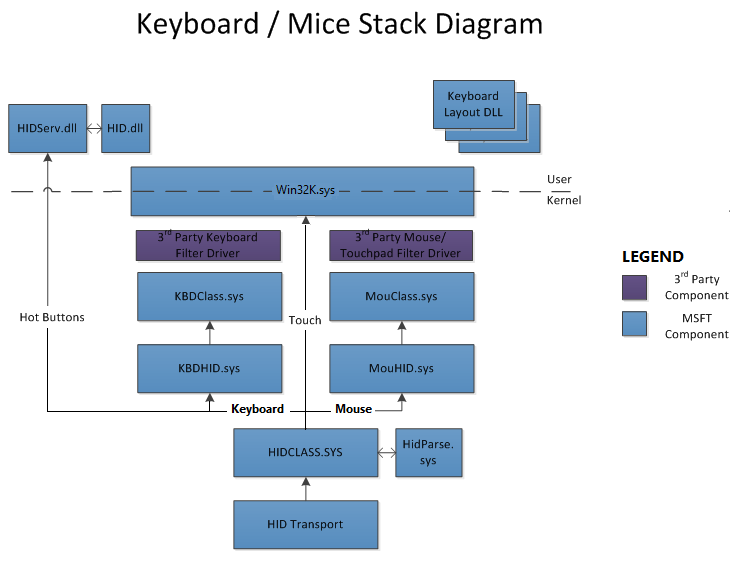
Windows Driver Kit – WDK, содержит необходимые библиотеки ntddk.h и ntddkbd.h

! Для установки драйвера необходимо отключить проверку цифровой подписи !

Принцип работы

В ОС Windows драйверы одного и того же устройства формируют цепочку драйверов или многоуровневые драйверы. Драйверы самого низкого уровня работают непосредственно с железом, а драйверы высокого уровня отвечают за представление данных. Используя многоуровневые драйверы можно перехватить любое устройство системы, избежав низкоуровневого кода.

Фильтрующий драйвер представляет собой один слой, работающий на самом высоком уровне, и обеспечивает протоколирование клавиш. Для реализации данного подхода создаётся устройство и добавляется в цепочку других устройств (рисунок Х). Кейлоггер является самым верхнем в этой цепочке, и для перехвата данных он будет просматривать пакеты, перемещающиеся по стеку драйверов.

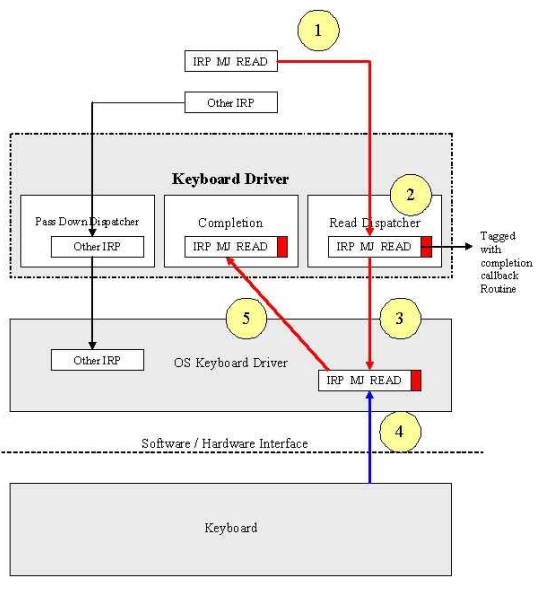


Стек драйверов клавиатуры и мыши

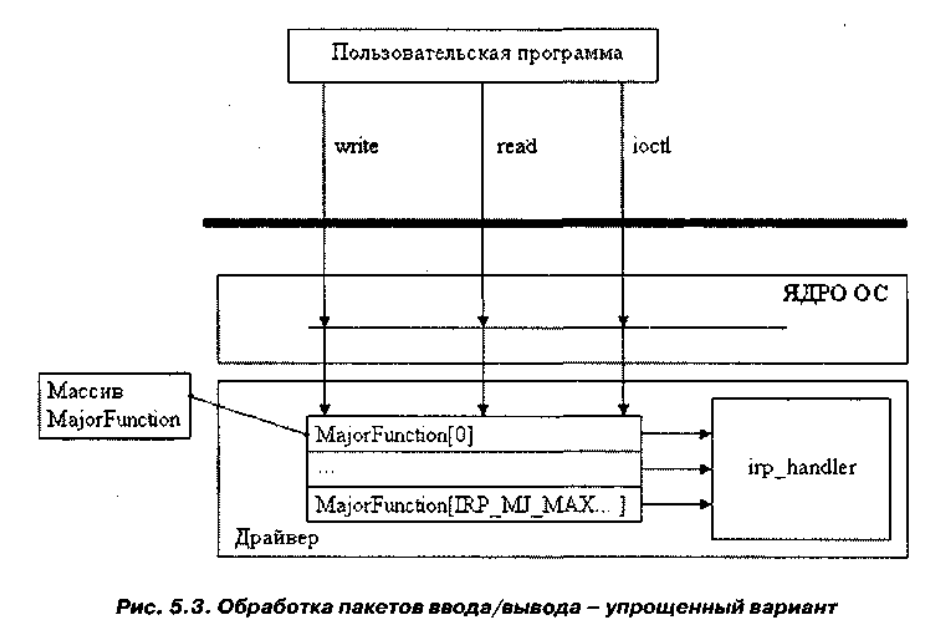
KBDHID.sys – Считывает скан-код нажатой клавиши и помещает его в IRP.

KBDCLASS.sys – Драйвер клавиатуры, обеспечивающий стандартную функциональность для всех клавиатур.

Рассмотрим, как цепочка устройств обрабатывает информацию о нажатиях клавиш (рисунок XXX).



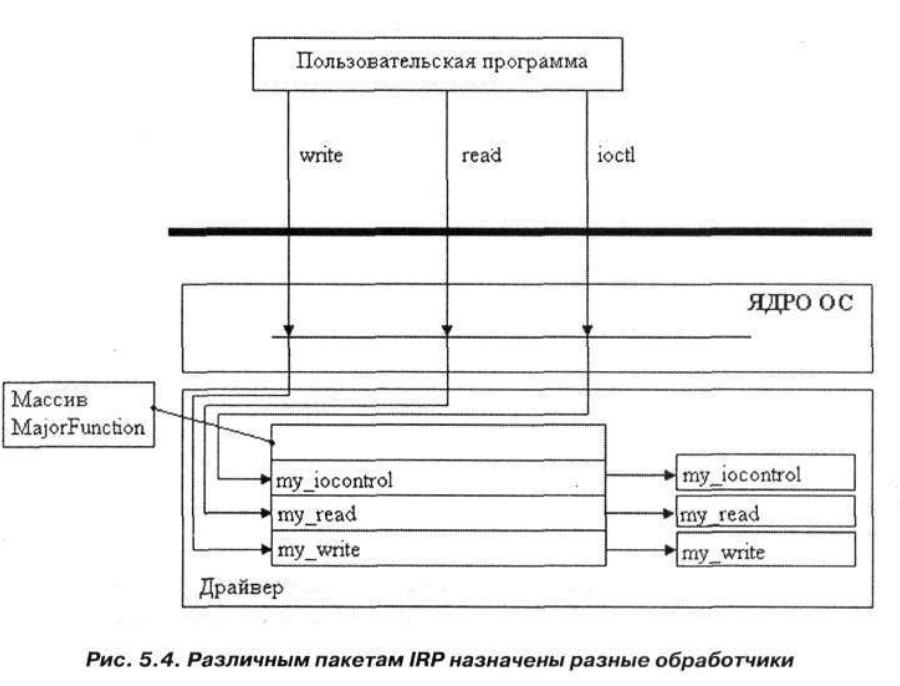
1. Прежде всего подаётся запрос на чтение нажатия клавиши, формируется пустой IRP[[2]](#footnote-2).
2. Для перехвата и обработки каждого IRP драйвер регистрирует специальную функцию (рисунок XXX). В структуре DRIVER\_OBJECT находится массив MajorFunction, который в свою очередь содержит IRP\_MJ\_MAXIMUM\_FUNCTION указателей на функции-обработчики. Регистрация функции заключается в присвоении адресов функций-обработчиков элементам массива MajorFunction.



Фильтрующий драйвер перехватывает IRP типа IRP\_MJ\_READ, устанавливает функцию, которая будет выполнена по его возвращении IRP (XXX), и отправляет его дальше. Все IRP другого типа драйвер просто отправляет вниз по стеку.

1. IRP идёт сверху вниз по стеку, пока не достигнет низкоуровневого драйвера, который ожидает нажатия клавиши.
2. Когда клавиша нажата, драйвер:
   * получает скан-код нажатой клавиши из буфера клавиатуры;
   * помещает скан-код клавиши в IRP;
   * передает IRP со скан-кодом «наверх»
3. На пути обратно выполняются те функции, которые были заданы фильтрующим драйвером (XXX), и кейлоггер имеет возможность получить скан коды из IRP.

Кейлоггер фильтрует только запросы типа IRP\_MJ\_READ, поскольку с помощью них получают коды нажатых и отпущенных клавиш.



Кейлоггеры режима ядра значительно более сложные, чем кейлоггеры пользовательского режима — как для реализации, так и в плане обнаружения. Они требуют специальных знаний для написания, но позволяют создавать кейлоггеры, абсолютно незаметные для всех приложений пользовательского режима. Для перехвата могут использоваться как документированные в Microsoft Driver Development Kit, так и недокументированные методы.

Достоинства — гарантированный перехват всех нажатий, невозможность обнаружения без использования драйвера, благодаря чему не все антикейлоггеры их определяют. Недостатки — необходимость установки собственного драйвера.

Kaspersky Internet Security проактивно детектирует такие кейлоггеры как Keylogger путем мониторинга клавиатурного стека устройств (должна быть включена опция «Обнаружение клавиатурных перехватчиков» подсистемы «Анализ активности приложений» модуля проактивной защиты)

A Final Caveat:  
There is an issue related to unloading a filter driver which needs to be mentioned. As we know, the dispatch routine “tags” keyboard IRPs with a completion routine on their way down the stack and they sit there waiting to be “completed” when the user presses a key on the keyboard. These IRPs are therefore “pending” completion (i.e. until they are filled with data and sent back up the stack). If we attempt to unload the driver while there is an IRP pending completion, the system will crash. This is clearly due to the fact that the memory address for the completion routine will be invalid when the OS tries to call it after unloading the driver. The obvious way to solve this is to simply not unload the driver. Like a rootkit, there is little need to unload a keylogger so this may be an acceptable solution. As an alternative, we can keep a count of the number of outstanding IRPs and refuse to unload until the count becomes 0. This alternative, however, has the unfortunate side effect that the driver won’t unload until the user presses another key (ie. until the IRP is filled with keyboard data and passed back up the stack and handled properly by the completion / logging routines).

1. Закреплённый за каждой клавишей определенный номер, однозначно связанный с распайкой клавиатурной матрицы и не зависящий напрямую от обозначений, нанесенных на поверхность клавиш [↑](#footnote-ref-1)
2. Input/output Request Packages – пакеты запроса ввода/вывода, которые диспетчер ввода/вывода передает драйверу устройства, когда программа пользовательского уровня обращается к этому устройству. IRP представляют собой структуру данных, содержащие буферы для хранения и передачи данных. [↑](#footnote-ref-2)