

Waggle Waggle

What is Keylogging?

- Something used to monitor the keys that are pressed on a computer
- Can be implemented in hardware or software
- Possible uses both legitimage and not
 - Allows for things to happen based on key presses
 - Hotkeys
 - Writing text
 - Allows for recording of text, passwords, etc.

Why is it bad?

- Allows for criminals to intercept passwords, personal information, confidential information, etc.
- Bypasses common protections like disk encryption, good password storage (hashing), etc.
- Significant interaction with a computer is through typing, and that can all be intercepted

How a keyboard works

- A keyboard is a small computer
- A local microcontroller scans keys being pressed
- Each key has a number assigned to it (scan code)
- Number maps to keyboard matrix map
- Two scan codes are generated per key press
 - One for when a user presses the key
 - One for when a user releases the key
 - Some keys have a function when they are pressed and held (Shift, Control, Alt)

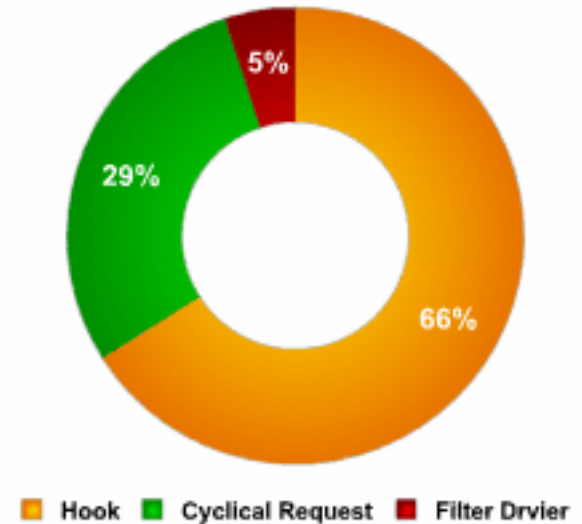
How a keyboard works

- When a key is pressed an electrical circuit is closed
- When the microcontroller scans next and detects the key is pressed it sends an interrupt request and scan code to the computer
- Same thing occurs when the key is released

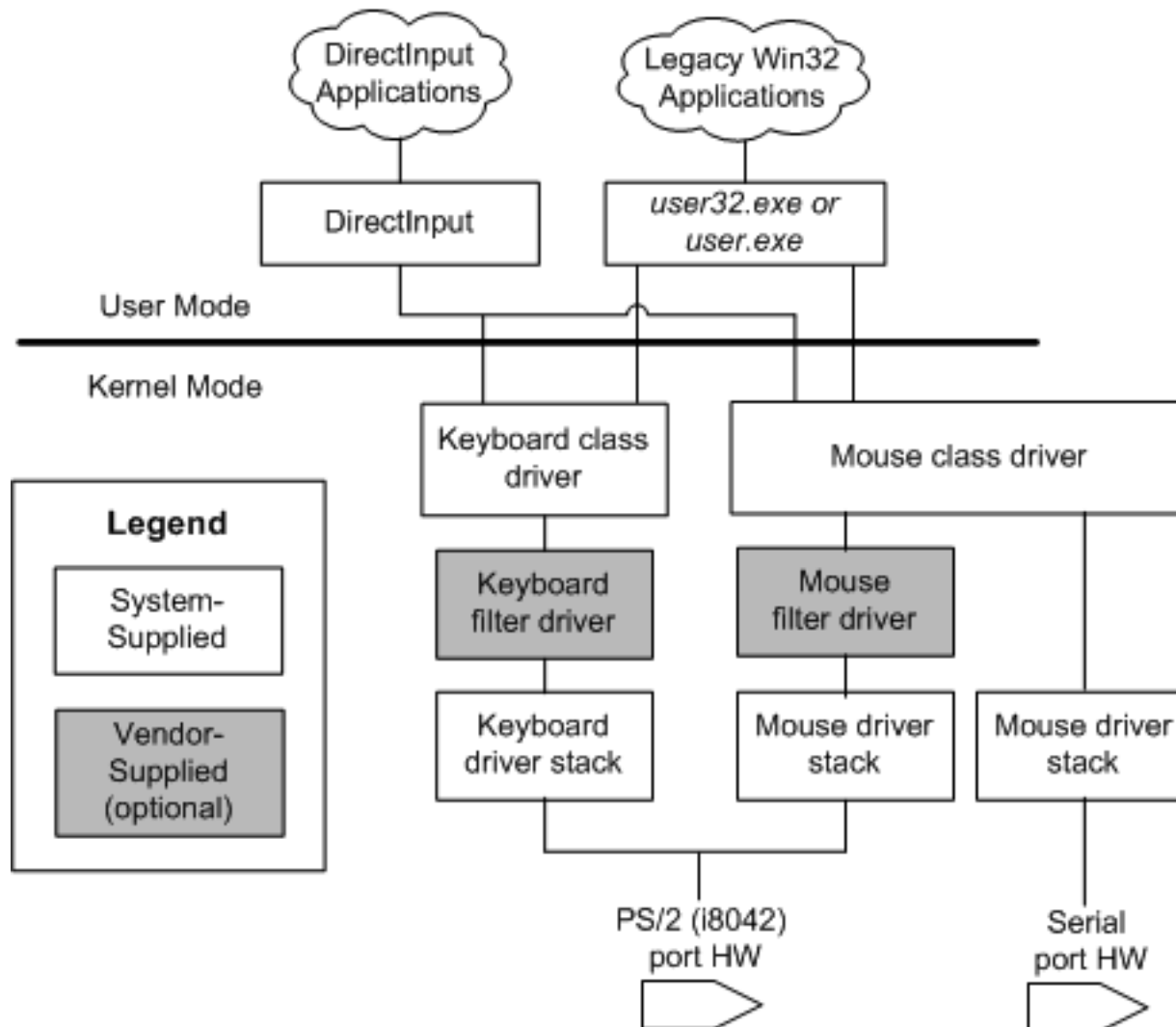
Hardware Keyloggers

Software Keyloggers

- Use various techniques to monitor keys as they are sent to the operating system
- On Windows:
 - A system hook which intercepts notification that a key has been pressed
 - A cyclical information keyboard request from the keyboard
 - Using a filter driver

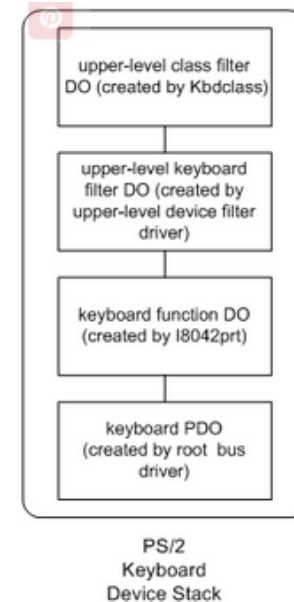


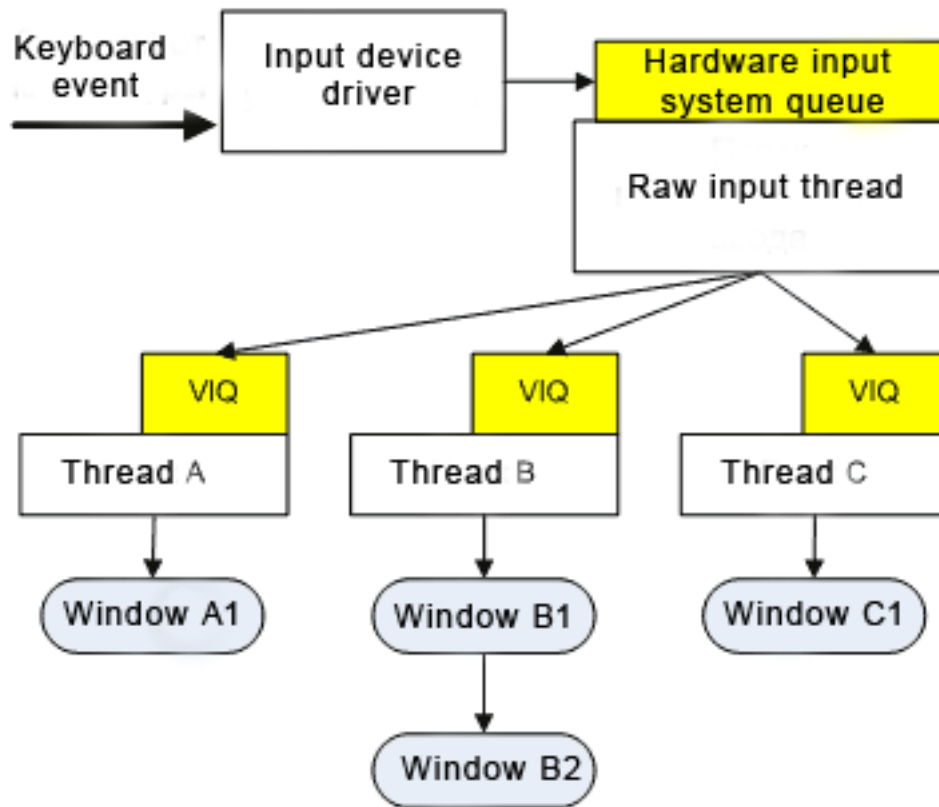
Windows driver stack for keyboards



Device Object Stack

1. Physical device object (PDO) – driver bus
2. Functional device object (FDO) – i8042ptr port
3. Optional: filter object for keyboard device
 - Developed by third parties
4. High level filter objects – DeviceKeyboardClass0





Raw Input Thread

- All incoming keyboard events are placed in hardware input system queue
- Transformed to Windows Messages
- Placed at the end of the virtualized input queue (VIQ) of the active thread
- Key scan codes are replaced with virtual key codes
 - Transforms keys from simply locations to actions

Last Mile

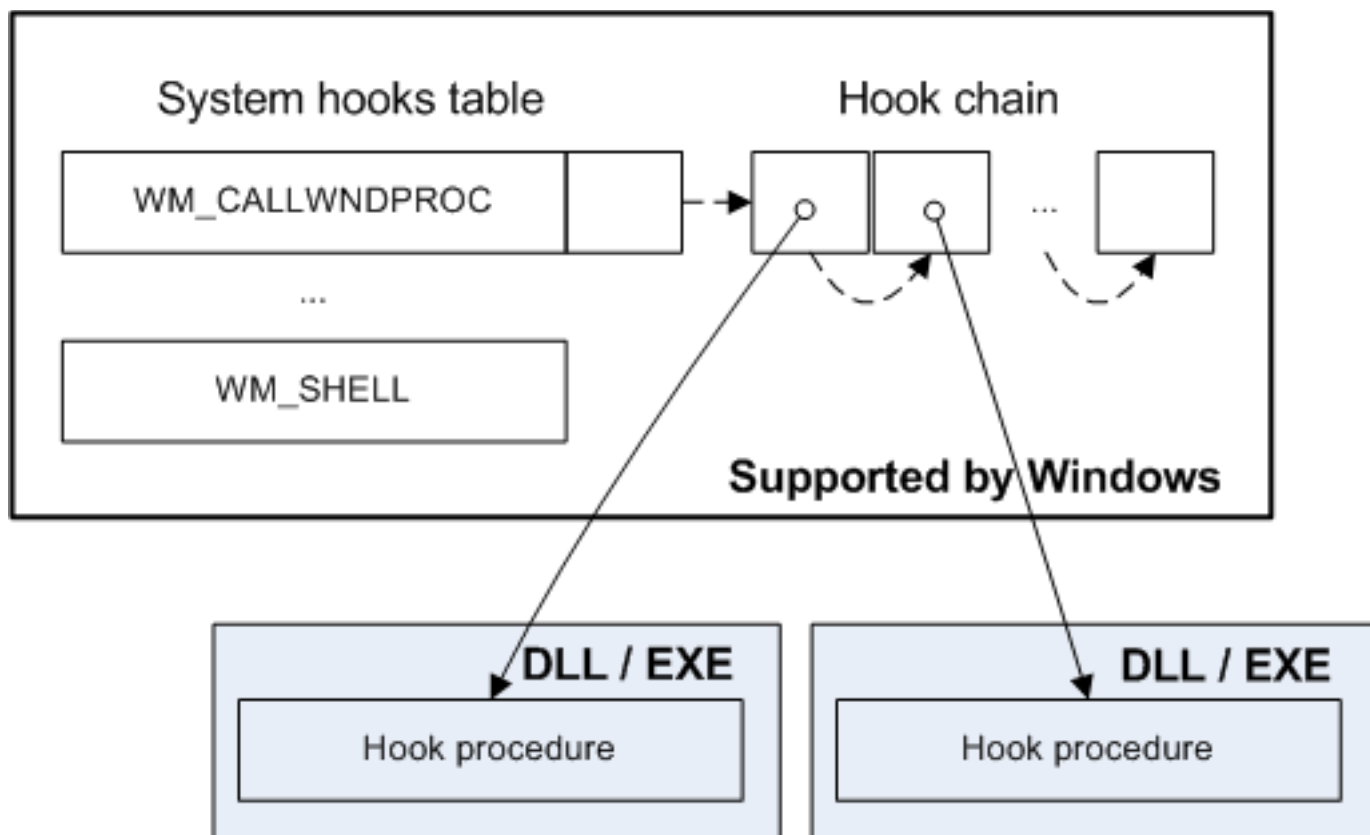
- Once a message has entered the thread message queue they are pulled from the queue with GetMessage
- Redirected to the window procedure with DispatchMessage
- Window procedure processes messages for the window where input is currently focussed
- If the window has an input focus all keyboard messages from the system queue will reach the appropriate function of the window

Keyboard Hooks

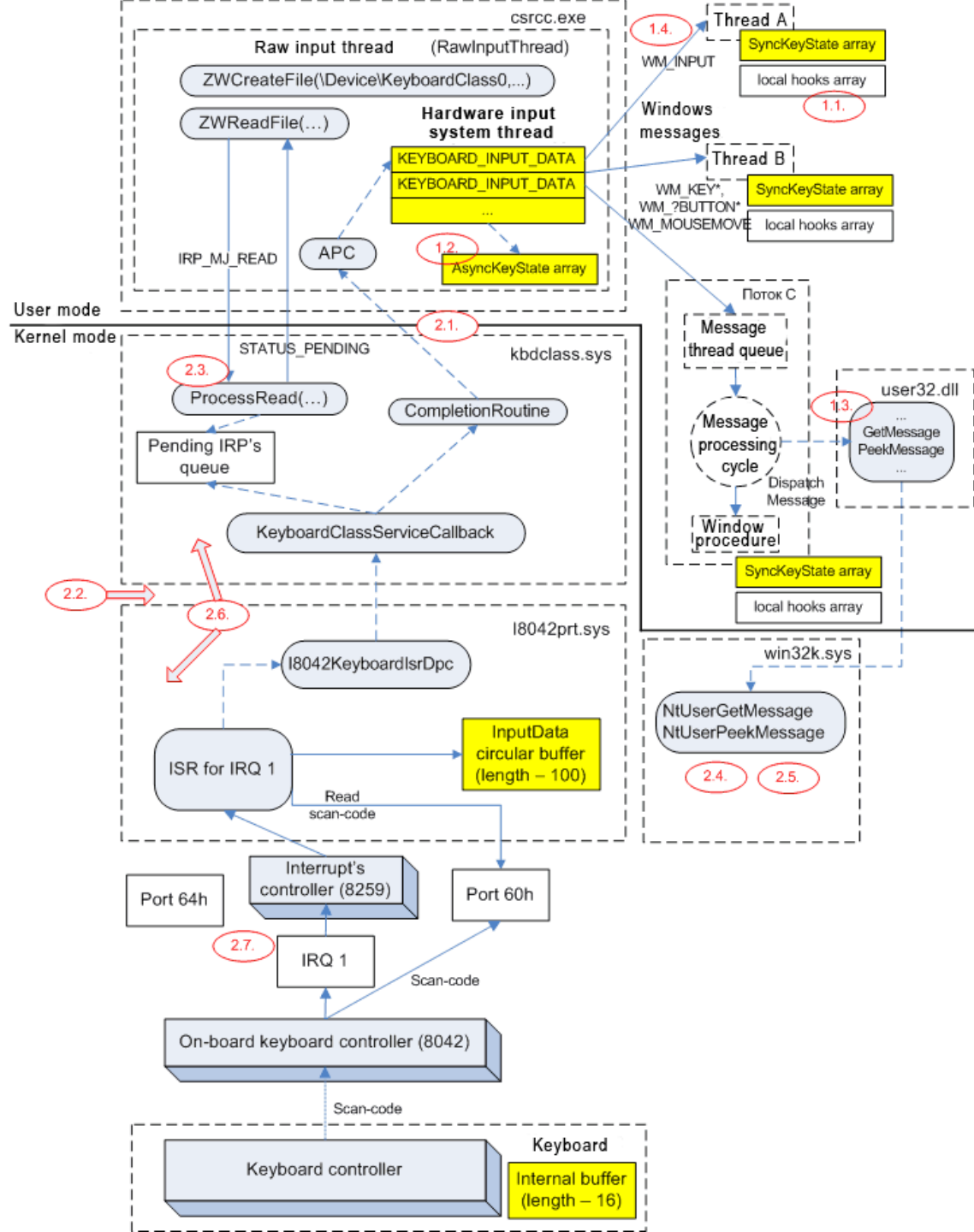
- Hooking is a mechanism used to intercept events using specific functions
- This function reacts to an event
- Filter functions are functions that receive notification of events
- A filter function is bound to a hook using the Win32 API:
 - SetWindowsHookEx
 - UnhookWindowsHookEx
- Hooks can be set system wide or for a thread

Hooks

- If multiple filter functions can be bound to a hook and are put in a function queue
- Most recently bound hook starts the queue and first function bound ends the queue
- The actions that can be performed by filter functions depends on the type of hook, but may include:
 - Track the occurrence of an event
 - Modify message parameters
 - Initiate message processing
 - Prevent the next filter function from being called



So where does keylogging occur?



User mode keyloggers

1.1 Setting hooks for keyboard messages

- Set a global hook for all keyboard events for all threads in the system

1.2 Using cyclical query of the keyboard

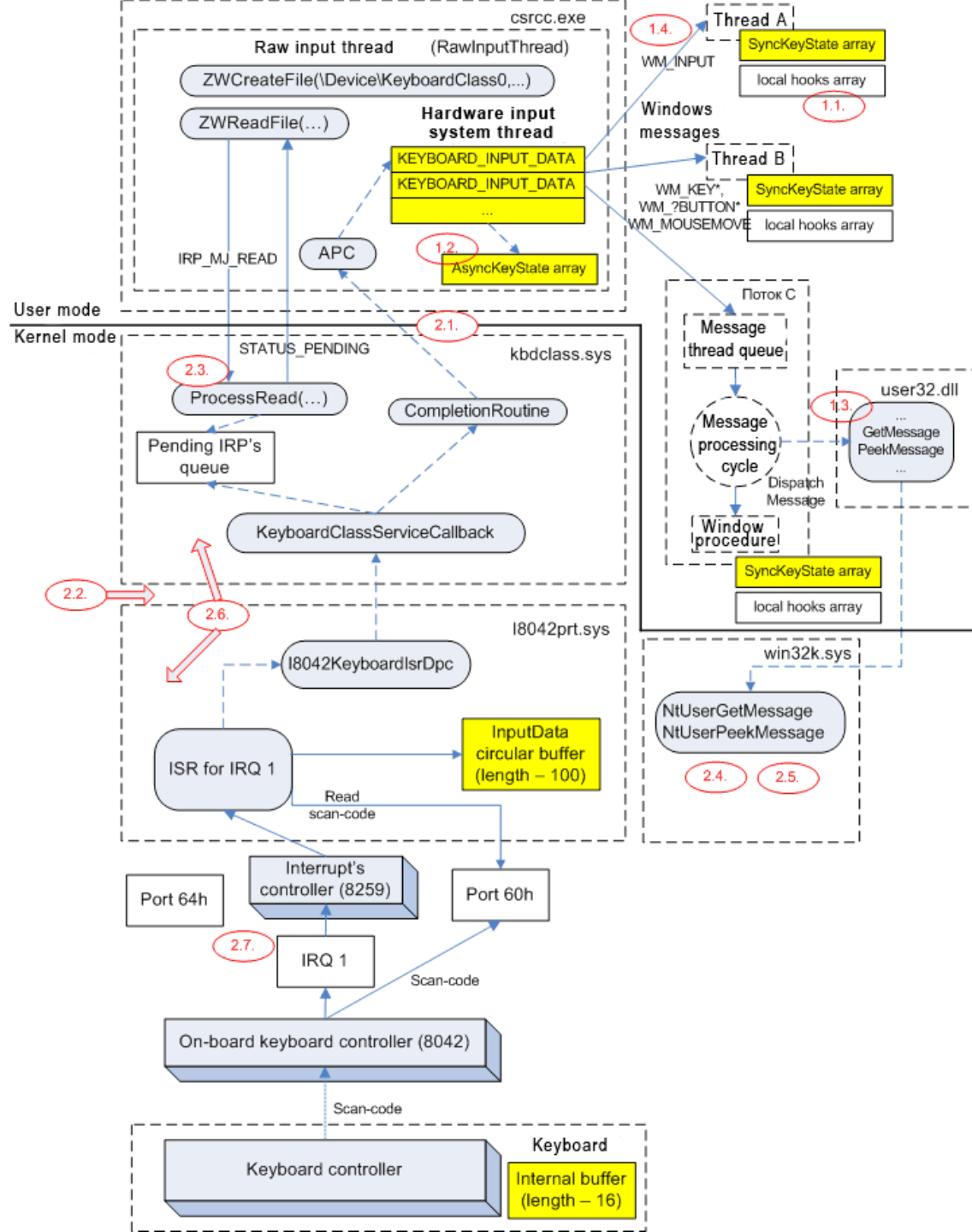
- The state of the keyboard is queried constantly

1.3 Hooking message processing functions by injecting into processes

- Injection into all processes and intercept GetMessage or PeekMessage functions

1.4 Using the raw input model

- Keylogger registers itself as a device that wants input from the keyboard



Kernel Mode Keyloggers

2.1 Using the keyboard driver filter Kbdclass

- Installs a filter above the device created by the Kbdclass driver

2.2 Using the filter driver of the i8042prt function driver

- Installs a filter on top of the device created by the i8042prt driver

2.3 Modifying the dispatch table of the Kbdclass driver

- Changes the IRP_MJ_READ entry point in the dispatch table for the Kbdclass driver

2.4 Modifying the system service table KeServiceDescriptorTableShadow

- Patches the entry point for NtUserGetMessage/PeakMessage in the second table of system services of the win32k.sys driver

2.5 Modifying the code of the NtUserGerMessage or NtUserPeekMessage function by splicing

- Splices the NtUserGetMessage or NtUserPeakMessage to modify the code

2.6 Substituting a driver in the keyboard stack of drivers

- Drop in a whole new Kbdclass driver or keyboard driver

2.7 Implementing a handler driver for interrupt 1 (IRQ 1)

- Write a kernel mode driver which hooks the keyboard interrupt and directly contacts the keyboard input and output ports

Sources / References

- <https://securelist.com/analysis/publications/36138/keyloggers-how-they-work-and-how-to-detect-them-part-1/>
- <https://securelist.com/analysis/publications/36358/keyloggers-implementing-keyloggers-in-windows-part-two/>
- <http://phrack.org/issues/59/14.html>