



## Hardware Keylogger Detection

An IRM Research White Paper by

**Andy Davis**



## IRM Research

---

Information technology constantly changes and advances. IRM is dedicated to keeping pace with new technology and continuing to innovate in the field of information security. This ensures that we are well informed of new issues and technologies, expanding our knowledge and providing world class services to our clients.

If you would like further information about the subject discussed in this paper then please send a request via the following link:

<http://www.irmplc.com/index.php/164-Enquiry>

## Hardware Keylogger Detection

---

Hardware keylogger technology poses a real risk to the confidentiality of data processed by PCs and other devices that utilise a keyboard. During this investigation a selection of common keyloggers were tested to discover if they could be detected after being introduced to a system. After extensive testing the following observations were made:

- Many keyloggers are shipped with default passwords, which, if not changed could result in keyloggers being detected by typing these passwords at the keyboard. However, the investigation was unable to discover any way to automate this process.
- Initially it was surmised that when the memory of keylogger devices was filled, recording would restart at the beginning, overwriting the previously recorded data. This appeared not to be the case for the devices tested. The KeyKatcher device stopped recording once the memory was full and the KeyGhost and Amecisco keyloggers overwrote data, but at the end of memory rather than the beginning. Furthermore, as the Amecisco keylogger recorded control commands such as those sent when a user presses the caps lock key, the device could be filled automatically using software.
- The detection of devices using current and voltage measurements was also tested and it was discovered that the introduction of a keylogger typically results in an extra 10mA of current being drawn from the PC, which could easily be detected by a hardware-based detection solution. However, the current drawn and subsequent change in voltage could not be detected by embedded motherboard sensors.

The investigation highlighted the difficulties associated with the detection of hardware-based keyloggers and therefore the following recommendations should be followed:

- Users should remain vigilant and report any unusual devices connected to their keyboards in addition to any spurious text that suddenly appears within one of their documents for no apparent reason (which would occur if a user unwittingly typed the keylogger password).
- Software-based keyboards could be used for the entry of sensitive information such as authentication credentials, which would result in the information not being recorded by a hardware keylogger.

The most robust solution with respect to authentication systems would be the introduction of additional factors such as token-based devices, smartcards or biometric sensors.

## Abstract

---

This paper provides an analysis of how hardware-based keyloggers work and offers suggestions as to how they may be detected. In addition, active countermeasures are proposed that may be used to reduce the impact of keyloggers that may be installed but can not be detected. Due to limited available resources and timescales the research that resulted in this paper is not exhaustive, but the intention is to promote further research in this area, as little has been written on the subject previously.

## Introduction

---

The recent high profile attack on the Japanese Bank, Sumitomo Mitsui and subsequent discovery that the attack utilised hardware keystroke logger (keylogger) technology has dramatically increased interest by the banking community in this area of attack.

### Hardware Keyloggers Overview

These are small (a few centimetres long) devices that are either inserted between the keyboard plug and the socket in the back of the computer or are embedded within the case of the keyboard. Commercial devices can store up to and in some cases in excess of two million keystrokes. The data is retrieved from the device by entering a user-selected password while the keyboard is connected to a computer and a text editor (such as Windows 'notepad') is running. The device releases the information into the text editor so that it can be saved into a file. Some companies, for example, Amecisco are now providing a service whereby they will install a hardware keylogger into any make of keyboard that is sent to them.

### Products Tested

There are a range of hardware-based keyloggers available; however, the following are considered to be among the market leaders and therefore those concentrated on in this assessment:

- KeyKatcher (PS/2 external device)<sup>1</sup>
- KeyGhost (PS/2 external device)<sup>2</sup>
- Amecisco (Embedded)<sup>3</sup>

## How does a PC keyboard work?

---

To understand how to detect a keylogger, first it is important to have an understanding of how a PC keyboard works under normal circumstances. A keyboard consists of a number of keys arranged in a matrix. The state of these keys (pressed, released or held down) is monitored by the 'keyboard encoder' (normally based on the Intel 8048), which sends the appropriate data associated with key state to the PC 'keyboard controller' (based on the Intel 8042). This controller decodes the data received from the keyboard and translates the data into meaningful information about which key has been pressed.

When a key state changes, a 'scan code' is generated by the keyboard encoder and sent to the controller, which is normally embedded within the motherboard of the PC. There are two types of scan codes; 'make' (sent when a key is pressed or held down) and 'break' codes (sent when a key is released).

The set of make and break codes for every key comprises a 'scan code set'. There are three standard scan code sets, named one, two, and three; however, most modern keyboards default to set two.

Although the original keyboard controllers used the Intel 8042 device, newer PCs used compatible devices and the controller in the PC used for the tests performed during this investigation was an Intel 8742.<sup>4</sup>

## The Low Level Keyboard Command Set

As well as interpreting scancodes, the controller can also send commands to the keyboard to modify certain characteristics or perform various diagnostic tests. The low level control of the keyboard controller can be accessed via Ports 0x60, 0x61 and 0x64 using Disk Operating System (DOS) ; an overview of how these can be used is provided in the tables below. This information was obtained from various low level keyboard programming resources on the Internet. <sup>5 6</sup>

### Port 0x60

This is used to issue commands to the keyboard and read scancodes and command responses back from the keyboard.

Command / Data (R/W)	Code	Data	Response
Set LEDs (W)	0xed	Bit 0 = Scroll lock Bit 1 = Number lock Bit 2 = Caps lock Bits 3-7 = 0x00	ACK (0xfa)
Keyboard Echo Request (W)	0xee	N/A	Keyboard Echo Reply (0xee)
Select Scan code set (W)	0xf0	0x00 = Return current 0x01 = Set 1 0x02 = Set 2 0x03 = Set 3	0x01 = Set 1 0x02 = Set 2 0x03 = Set 3
Identify Keyboard (W)	0xf2	N/A	Timeout = XT ACK (0xfa) = AT ACK 0xab 0x41 = MFII
Set Typematic rate (W)	0xf3	Bits 0-4 = rate:  0x00 = 30 keys/sec 0x01 = 26.7 keys/sec 0x02 = 24 keys/sec 0x04 = 20 keys/sec 0x08 = 15 keys/sec 0x0a = 10 keys/sec 0x0d = 9 keys/sec 0x10 = 7.5 keys/sec 0x14 = 5 keys/sec 0x1f = 2 keys/sec  Bits 5-6 = Pause before repeat:  0x00 = 250ms 0x01 = 500ms 0x02 = 750ms 0x04 = 1000ms  Bit 7 = always 0x00	ACK (0xfa)
Internal Diagnostics (W)	0xff	N/A	0xaa = successful BAT (Basic Assurance Test) 0xfc = unsuccessful BAT

Resend data please (sent by keyboard) (R)	N/A	N/A	0xfe
Keyboard error – too many keys being pressed simultaneously (sent by keyboard) (R)	N/A	N/A	0x00
General keyboard error (sent by keyboard) (R)	N/A	N/A	0xff
All other data on port 0x60 relates to the scan codes of keys pressed on the keyboard. The scan codes represent unique numbers for a key-press and key-release of each key on the keyboard (R)	N/A	N/A	Scan code for key being pressed

**Table 1:** Summary of port 0x60 commands

### Port 0x61

This port is used to acknowledge that a scan code had been received. It does this by disabling and then immediately re-enabling the keyboard

Command / Data (R/W)	Code	Data	Response
Disable/Enable keyboard (W)	N/A	Bits 0-5 = PPI (Programmable Peripheral Interface – nothing to do with the keyboard)  Bit 6 = Hold keyboard clock low (keyboard can not send any data)  Bit 7: 0x00 = Enable keyboard 0x01 = Disable keyboard	N/A

**Table 2:** Summary of port 0x61 commands

### Port 0x64

This port is used for data and status and control purposes:

Command / Data (R/W)	Code	Data	Response
Statusport (R)	N/A	N/A	Bit 0: 0x00 = Output buffer empty 0x01 = Keyboard data is in buffer Bit 1: 0x00 = Command buffer empty 0x01 = User data is in buffer Bit 2: 0x00 = Reset 0x01 = Self test successful Bit 3: 0x00 = 0x60 was last

			accessed port 0x01 = 0x64 was last accessed port Bit 4: 0x00 = Keyboard locked 0x01 = Keyboard enabled Bit 5 = PS/2 mouse interface Bit 6: 0x01 = Timeout occurred Bit 7: 0x01 = Last transmission parity error
Keyboard self test (W)	0xaa	N/A	0x55
Interface test (W)	0xab	N/A	0x00 = No error 0x01 = Clock low 0x02 = Clock high 0x03 = Data low 0x04 = Data high 0x05 = Total error
Deactivate keyboard (W)	0xad	N/A	N/A
Activate keyboard (W)	0xae	N/A	N/A
Read Input port (the Input port contains the keys being pressed and some control information) (W)	0xc0	N/A	Output sent to port 0x60 Bit 0 = Keyboard data in pin Bit 1 = PS/2 mouse in pin Bit 2-5 = reserved Bit 6 = Colour/mono screen Bit 7: 0x00 = keyboard locked 0x01 = keyboard not locked
Put low nibble of the Input port over bits 4-7 of the Status port	0xc1	N/A	N/A
Put high nibble of the Input port over bits 0-3 of the Status port	0xc2	N/A	N/A
Read Output port	0xd0	N/A	N/A
Write parameter to Output port	0xd1	Parameter send to port 0x60	N/A
Write keyboard buffer (parameter written to input buffer as if received from the keyboard).	0xd2	Parameter send to port 0x60	N/A
Write mouse buffer (parameter written to input buffer as if received from the mouse).	0xd3	Parameter send to port 0x60	N/A
Write mouse device (sends parameter to the mouse)	0xd4	Parameter send to port 0x60	N/A
Read test port (value returned on test port)	0xe0	N/A	N/A
Pulse Output port – Pulses commands low nibble onto low nibble of the Output port.	0xf0 – 0xff	N/A	N/A

**Table 3:** Summary of port 0x64 commands

## How do Hardware Keyloggers Work?

---

Essentially a hardware keylogger just taps into the data line within the keyboard cable and records all scancodes and control information sent between the keyboard and PC. For legal reasons, during this investigation no devices were reverse-engineered, as each of the manufacturers expressly prohibit any tampering with the devices.

A section of the KeyKatcher user agreement is shown below:

*'This license is limited to the rights above. Therefore, you are expressly prohibited from taking any other action with the Device, including but not limited to:*

- a) modify, reverse engineer, de-compile or disassemble the Device or the Program;*
- b) make any attempts to defeat the code protection which is in place on the internal microcontroller;*
- c) make any attempts to read or copy the Program;*
- d) attempt to cut open the casing around the Device;'*

As the commercial devices could not be reverse engineered to reveal the internals of a keylogger, an open source product has been used. The company Keelog.com provide instructions to build a DIY hardware keylogger on their web site<sup>7</sup> and they kindly allowed the details to be reproduced here.

### DIY Keylogger Parts list

The following easily-available components are required in order to construct a DIY hardware keylogger:

- 1 x Atmel AT89C2051 microcontroller (AT89C1051 or AT89C4051 will do as well)
- 1 x 24C512 serial EEPROM chip
- 1 x 12MHz crystal (as small as possible)
- 2 x 33pF capacitors
- 1 x 10uF capacitor (as small as possible)
- 1 x 10k $\Omega$  resistor

## DIY Keylogger Circuit Diagram

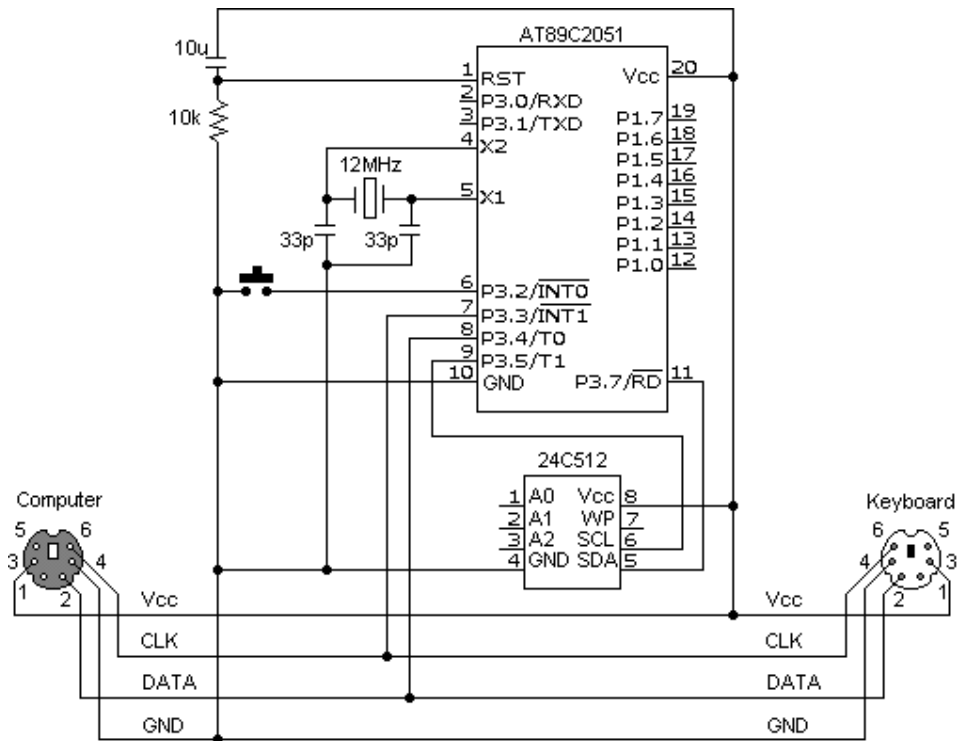


Figure 1: DIY Keylogger circuit diagram

## DIY Keylogger Software

The software required to be programmed into the Atmel microcontroller is provided as source code<sup>8</sup> and software that will retrieve the captured keystrokes is available as an executable file<sup>9</sup>.

The previous two sections have served to provide an overview of the technology and protocols associated with the use of keyboards under normal circumstances and with keylogger technology. The next stage is to identify areas of investigation based on this information that may reveal the presence of such a device from either a software or hardware perspective.

## Hardware Keylogger Detection

---

The keyboard protocol command set provides a great deal of scope for communication between the PC and the keyboard and therefore the possibility of anomalies introduced by keylogging hardware.

### Do keyloggers introduce anomalies in command responses?

Software tools were created to send each known command to the keyboard and the responses were then recorded, both with and without the various keylogger attached. Examples of commands sent included:

- Controlling the state of the Caps, Number and Scroll lock LEDs
- Setting the typematic repeat rate
- Running the BAT (Basic Assurance Test)

Using these standard control commands there were no differences in the responses when keyloggers were introduced. Next tools were created to 'fuzz' the protocol. Fuzzing is a technique whereby every possible combination of control and data messages are constructed in order to test the responses when unusual combinations are used. However, again, there were no differences in any of the responses when keyloggers were present.

### Command Responses Timing Issues?

Although the actual responses from the keyboard were no different regardless of whether a keylogger was present, it was surmised that the additional circuitry may introduce a delay. The previously written tools were therefore modified in order to identify if any timing differences could be detected in the responses when a keylogger was introduced.

The timing routines used had a resolution of milliseconds; however, no consistent timing differences were observed. It may be the case that a time delay is introduced by keyloggers, but it was not possible to detect as it was so small.

### Emulating keypresses

One of the commands available (0xd1, which is sent to port 0x64) enables keypresses to be emulated. The scancodes associated with the keypresses to be emulated are sent to port 0x60 after the 0xd1 command has been issued.

The intention here was to be able to create software-generated keypresses so that a dictionary attack could be performed on a keylogger in order to gain access to the administration interface. Unfortunately, although the generated scancodes resulted in text being displayed on the screen, as they were generated by the keyboard controller the data never passed through the keyloggers and was therefore never interpreted or recorded by them.

## The Authentication Mechanism and Menu System

All the tested keyloggers had a text-based administration interface, driven by a menu that was accessible by typing a password while a text editor was running. The keyloggers produce the interactive menu by generating scancodes that are sent to the keyboard controller (as if they were typed on the keyboard).

### Default passwords

Many of the keyloggers tested were delivered with default passwords for the administration interface; these are listed in Table 4.

Keylogger	Default Password
KeyKatcher	keykatch
KeyGhost (PS/2)	vghostlog
Amecisco	N/A – different passwords selected for each installation

**Table 4:** Summary of default passwords

In the instruction manuals for each of the devices tested it explained that if the password was changed and subsequently forgotten then the devices would be rendered useless and would need to be returned to the manufacturer for reprogramming. Therefore, users may be inclined not to change the password from the default value. This information could be used in an incident response situation where a keylogger device has been discovered and the decision is made to attempt to gain access to discover what information has been recorded.

### Changing the Keylogger Password

For each of the devices examined the password was changed in order to attempt to deduce any information from the way the mechanism worked. The findings are detailed in Table 5.

Keylogger	Password Change Process and Constraints
KeyKatcher	The password is changed instantly and must be between 6 and 8 characters in length
KeyGhost (PS/2)	The password is changed as part of a reformatting procedure and wipes the memory in the process (which takes approximately 1 minute). The password must be between 8 and 12 characters in length.
Amecisco	The password is changed instantly and must be less than 16 characters.

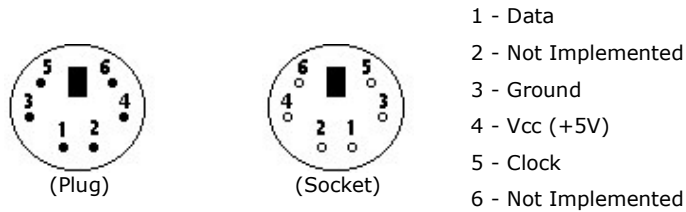
**Table 5:** The password change process for each keylogger

The only information that could be gleaned from the password change process was that the KeyGhost appeared to store the password in a different way to the others, as a full memory erasure was required when the password was changed.

As the devices were required to be returned to the manufacturer if a password was forgotten, it was surmised that there may be a simple process that the manufacturer used to perform this technique.

### Hardware-based Password Reset Solutions

One simple solution that could be employed to reset the password involves the use of the two spare pins on a PS/2 connector. Figure 2 details the pin allocations for PS/2 connectors:



**Figure 2:** The PS/2 physical interface

As can be seen, pins 2 and 6 are not implemented and could therefore be used as part of mechanism for resetting the password. The following tests were performed on the keyloggers tested:

- The pins were shorted together when connected / not connected to the PC
- +5V was applied to each pin
- Each pin was grounded

However, none of the techniques resulted in the password being reset.

### Software-based Password Reset Solutions

Another mechanism that could be employed to reset the password is a 'backdoor' system password that is hard-coded into the software. As these could be any difficult-to-guess key combination it was not surprising that no passwords of this nature were identified during the evaluation. The only definitive way to establish if hard-coded passwords were present would be to reverse-engineer the device and disassemble the system code, which, as explained earlier was not an option during this investigation.

### Memory Limitations

The range of devices tested all had different memory capacities, but an area that was investigated was how each keylogger reacted once the memory had been filled. Potential responses may be as follows:

- Memory is overwritten at the beginning
- Memory is overwritten in a location other than the beginning
- Keylogger stops recording
- Keylogger stops responding
- Keylogger outputs an error

The actual responses of each keylogger when their memory was filled with data are shown in Table 6.

Keylogger	What Happens When the Memory is Full
KeyKatcher	Stops recording
KeyGhost	Continues recording overwriting the end of memory
Amecisco	Continues recording overwriting the end of memory

**Table 6:** Summary of how keyloggers react once the memory is full

The implications of these findings are that if a keylogger can be filled with data then it makes the process of identifying sensitive information such as passwords more difficult. Furthermore, if the device can be filled with meaningless data, generated by software, then an active keylogger mitigation process could be employed whereby every time a user logs off from their machine a process could be activated to fill any keylogger that was present with data that overwrites any sensitive information that may have been captured.

The only way that a keylogger could be automatically filled with data is if they stored control commands (which could easily be generated using software). Therefore each device was tested to discover if control information (such as changing the state of the scroll lock LED) was recorded. The results are shown in Table 7.

Keylogger	Are Keyboard Control Commands Recorded?
KeyKatcher	No
KeyGhost	No
Amecisco	Yes

**Table 7:** Summary of keyloggers control command recoding capabilities

The results show that an active technique that overwrites memory with control commands could therefore be used to mitigate the presence of an Amecisco keylogger.

## Keylogger Detection using Voltage or Current Measurement

Keylogger devices are small electronic circuits, which need to draw power from the PC that they are connected to. Therefore, it should be possible to detect an increase in the current drawn from the PC power supply.

### Keyboard Current Measurement

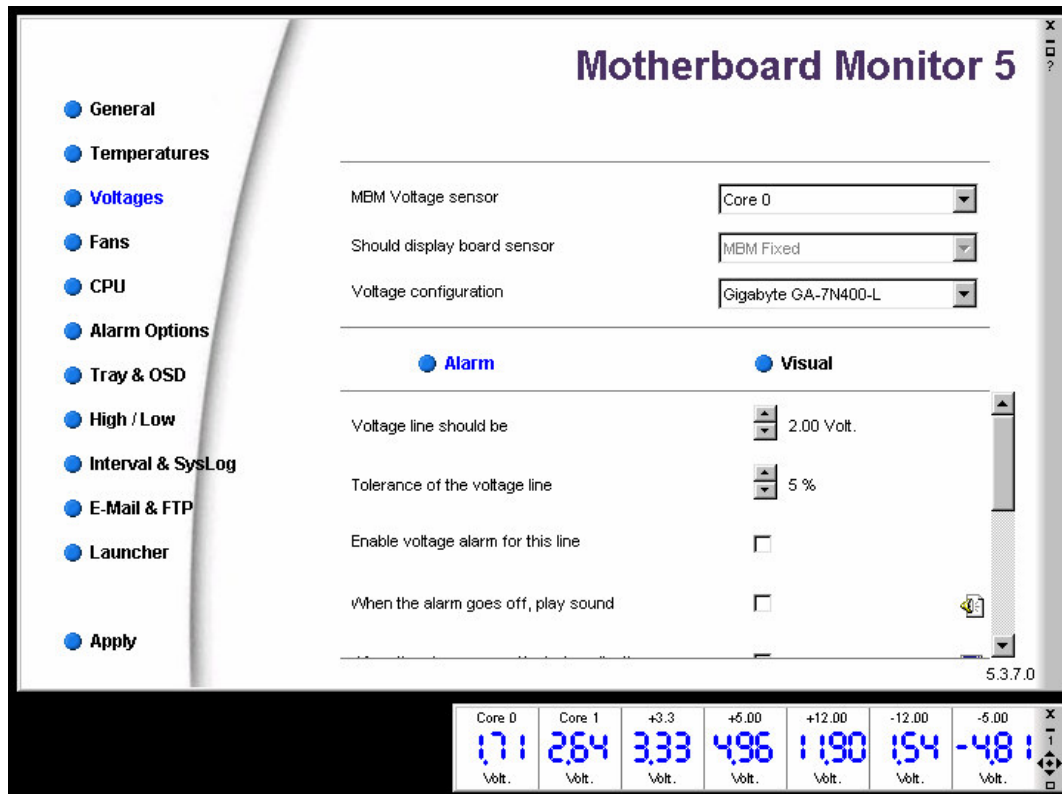
An ammeter (a device that measures current) was placed inline between the keyboard and PC and measurements were taken with and without keyloggers. Although there is quite a large variation in the amount of current drawn by different brands of keyboard, by introducing the keyloggers the current drawn typically increased by approximately 10mA, which could easily be detected by the ammeter.

Therefore, as keylogging hardware will always need to draw current to power the onboard circuitry, one solution could be a hardware device built into the PC motherboard. However, there would need to be a degree of intelligence in this device, as extra current is also drawn when users press the keys that result in LEDs being activated e.g. caps lock.

### PC Voltage Measurement

Many modern motherboards provide the facility to measure the voltage available to different elements of the system. Theoretically, any increase in current drawn may result in a drop in voltage, which could be measured and therefore indicate the introduction of a keylogger.

The tool 'Motherboard Monitor 5'<sup>10</sup> was used to display the voltage measurements for the different elements of the test system, which used a Gigabyte GA-7N400L motherboard, as shown in Figure 3.



**Figure 3:** A screenshot of Motherboard Monitor 5

Over a period of an hour the PC was used to perform a range of normal tasks that may or may not have affected the voltages. These tasks were:

- Executing a range of programs (which generated hard disk activity)
- Accessing the CDROM drive
- Accessing the floppy drive
- Using the keyboard and pressing keys such as caps lock

During this period the voltages were recorded regularly. Once this 'control' state had been established the same process was repeated with each keylogger connected to the system and the results recorded.

When the results were compared to the 'control' it was clear that no obvious voltage changes could be directly attributed to the introduction of a keylogger device. This was most likely due to the voltage changes being so small that they could not be detected by the motherboard sensors.

## References

---

(1) KeyKatcher keylogger

<http://www.keykatcher.com>

(2) KeyGhost keylogger

<http://www.keyghost.com>

(3) Amecisco keylogger

<http://www.amecisco.com>

(4) Intel 8742 Keyboard controller specification sheet

<http://www.intel.com/design/archives/periphrl/docs/29025601.pdf>

(5) Wout Mertens Guide to Keyboard Programming V1.1

[http://www.osdever.net/documents/pdf/wout\\_kbd.pdf](http://www.osdever.net/documents/pdf/wout_kbd.pdf)

(6) The AT-PS/2 Keyboard Interface, Adam Chapweske

<http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/keyboard/atkeyboard.html>

(7) The DIY Hardware Keylogger

<http://www.keelog.com/diy.html>

(8) Source code for the DIY Hardware Keylogger

<http://www.keelog.com/files/diy.asm>

(9) Retrieval Software for the DIY Hardware Keylogger

<http://www.keelog.com/files/keygrab.zip>

(10) Alex van Kaam's Motherboard Monitor 5

<http://mbm.livewiredev.com/>

## About the Author

---

**Andy Davis** is the IRM Chief Research Officer where his responsibilities include managing the technical research programme and the software security team. Andy also performs scenario-based penetration testing, software security analysis and bespoke security consultancy for a range of clients throughout Europe, North America and South East Asia, including UK Government Departments, Law enforcement, CNI (Critical National Infrastructure) and financial institutions. Andy is a certified CESG CHECK Scheme team leader and CLAS consultant with six years of commercial consultancy experience at IRM and ten years of Government Information Security experience gained from working at GCHQ in Cheltenham. Four of these years were spent operationally seconded to other areas within the UK Intelligence Community, which included the participation in several software security research placements at NSA, Fort Meade, MD.

### About IRM

Information Risk Management Plc (IRM) is a vendor independent information risk consultancy, founded in 1998. IRM has become a leader in client side risk assessment, technical level auditing and in the research and development of security vulnerabilities and tools. IRM is headquartered in London with Technical Centres in Europe and Asia as well as Regional Offices in the Far East and North America. Please visit our website at [www.irmplc.com](http://www.irmplc.com) for further information.