Detecting Hardware Keyloggers

Fabian Mihailowitsch
October 13, 2010
Who?

- Fabian Mihailowitsch
- Former Software Developer
  - German energy combine
- IT-Security Consultant
  - cirosec GmbH
  - Penetration Tests
  - Source Code Reviews

Contact

- Email: fm@cirosec.de
- www.cirosec.de
Hardware Keylogger

- PS/2
- USB

Hardware Keyloggers are undetectable by Software

„Visual inspection is the primary means of detecting hardware keyloggers, since there are no known methods of detecting them through software. “, en.wikipedia.org, 26.09.10

Talk: Detection of Hardware Keyloggers with Software ;)

Fabian Mihailowitsch (cirosec GmbH)
Why?

- Less research on this topic
  - Few information
  - No practical way to detect HKL

- Because HKL are a threat
  - 2005 (GB): Sumitomo Bank
    - Attackers tried to steal 423 million USD
    - Multiple HKL were installed
  - How about your company?

- Solution to identify HKL in large enterprises
  - Visual inspection is impractical
  - Only possible via software
Hardware Keylogger

- Hardware Keylogger
  - USB
  - PS/2
  - Keyboard Module
  - Mini- / PCI card

- Installed between PC and Keyboard
  - Records key strokes

- Captured data are retrieved
  - Software
  - Keyboard
    - Ghost typing
    - Flash drive
  - Wi-Fi-Access
    - Email
    - TCP connect
  - Bluetooth
Hardware Keylogger

Features
- Up to 2 GB flash memory
- Encryption
- Password protection
- Timestamping
- Time use charts
- Search functions
- Upgradeable firmware

Pricing
- PS/2: 32.00 USD
- USB: 58.00 USD
Hardware Keylogger – The companies

- **Big ones**
  - KeyDemon, KeeLog, ... (PL)
  - KeyCarbon (US)

- **Most companies rebrand KeyDemon**
  - KeyCobra
  - KeyLlama (once own products)
  - ...

- **Also „famous“ (older products)**
  - KEYKatcher (US)
  - KeyGhost (NZ)
  - KeyShark (DE)

- **The others**
  - WirelessKeylogger (UK)
  - Exotic Stuff (mostly CN)
  - Some Open Source Keylogger
PS/2 – How does it work

- **Keyboard**
  - Wire matrix
  - Microcontroller
  - Sends scancode (make/break)

- **PC**
  - Keyboard Controller (KBC)
    - 0x60: I/O-Buffer
    - 0x64: Status
PS/2 – How does it work

- Communication KBC <-> Keyboard
  - Obvious
    - Scancodes
  - Not that obvious ;)
    - Set LEDs
    - Choose scancode
    - Set repeat rate
    - Keyboard self-test / reset
    - Ping
    - ...

Example (Ping)

KBC sends "ping" (0xEE) via 0x60
KB sends "pong" (0xEE) to 0x60
PS/2 – How does it work

- PS/2 is a serial interface

- Communication
  - DATA
  - CLK
  - Bidirectional
  - Keyboard defines clock (30 – 50 ns)

- Data frames
  - KB (11 bit): startbit, D0-D7 [data], odd parity, stopbit
  - KBC (12 bit): startbit, D0-D7 [data], odd parity, stopbit, ACK (KB)
PS/2 – How does it work

- PS/2 is a serial interface

- Communication
  - DATA
  - CLK
  - Bidirectional
  - Keyboard defines clock (30 – 50 ns)
Detecting PS/2 Hardware Keylogger

- Current measurement
  - Additional electronic components
    - Additional power consumption ;)  
      - KeyDemon = 65 mA
      - KeyKatcher = 54 mA
  - More current is drawn
  - Cannot be measured by software
Detecting PS/2 Hardware Keylogger

- Keylogger are password protected
  - Entered via Keyboard
  - Ghost typing
  - Shipped with default password
  - Password restore is complex

- Brute Force password
  - Via software
  - Check ghost typing
Detecting PS/2 Hardware Keylogger

- **Problem**
  - Tested HKL don’t tap the data line
  - HKL are placed „inline“
  - HKL knows the data flow
  - KBC can’t send fake keystrokes
Detecting PS/2 Hardware Keylogger

- However
  - Some KB commands (0x60) lead to fake key presses
  - Maybe keyboard response is interpreted...

- Brute Force password
  - Translation Table (KB command -> key press)
  - Brute Force attack via Software

- Practical?
  - Limited amount of chars (~10)
  - Not all passwords can be Brute Forced
  - Works for: KeyGhost, KEYKatcher (some)
Detecting PS/2 Hardware Keylogger

Demo
Detecting PS/2 Hardware Keylogger

- Changes on the line
  - HKL are placed „inline“

- HKL might change signals on the line
  - Different signals (data)
  - Own clock (30-50 ns)
  - Slight dislocation of data/clock signal
  - Maybe more... ;}
Detecting PS/2 Hardware Keylogger

- Analyze the data flow
  - Tap signal at the keyboard
  - Tap signal after the keylogger
Detecting PS/2 Hardware Keylogger

- Result:

![Diagram showing keylogger and keyboard signals]
Detecting PS/2 Hardware Keylogger

- Clock is set to low
  - Delay of the HKL
Detecting PS/2 Hardware Keylogger

- Clock is set to high
  - Same timing
Detecting PS/2 Hardware Keylogger

- Clock cycles are shorter for HKL
  - Probably HKL generates own clock signal
  - Can be detected on the wire
  - No possibility to detect via software
  - Exact clock state cannot be retrieved by KBC

- But the clock signal starts later...
  - Remember when clock was pulled low
  - HKL might cause a delay on the wire
Detecting PS/2 Hardware Keylogger

- Time Measurement
  - Tested HKL were placed „inline“
  - Microprocessor has to analyze the signal and pass it on
  - This additional logic increase signal propagation time
Detecting PS/2 Hardware Keylogger

- Time Measurement
  - Tested HKL were placed “inline”
  - Microprocessor has to analyze the signal and pass it on
  - This additional logic increase signal propagation time
Detecting PS/2 Hardware Keylogger

Basic idea

- Send command to KB, wait for response and measure run time
- Like a „ping“

```
_start:
xor %ecx, %ecx
mov $0x9999, %cx

_wait1:
in $0x60, %al
xor %eax, %eax
in $0x64, %al
test $0x2, %al
jne _wait1
mov $0xF2, %al
out %al, $0x60

_wait2:
xor %eax, %eax
in $0x60, %al
cmp $0xFA, %al
jne _wait2
loop _wait1
ret
```

Repeat 9999x:

Send „Identify Keyboard“ (0xF2)

Wait until Keyboard responds with „MF-II“ (0xFA)
Detecting PS/2 Hardware Keylogger

- Delay introduced by the HKL is very (!) small
  - Previous code can’t be used in „normal OS state“
    - scheduler, interrupts, ...
    - Measurement isn’t exact enough
- Code must run exclusively
  - Get the most accurate measurement
Detecting PS/2 Hardware Keylogger

Solution

- Loadable Kernel Module
- Get CPU exclusively
  - Deactivate interrupts for processor
  - Disable kernel preemption
  - SMP locking
- Run ASM code („ping“)
- Measure runtime of the code
  - Interrupts are disabled
  - Read processors time stamp counter (rdtsc)
  - Counter is increased every clock cycle
  - Use the number of clock cycles
- Restore everything and write result to kernel message buffer
Detecting PS/2 Hardware Keylogger

- **Time Measurement**
  - **Results**

<table>
<thead>
<tr>
<th>Setup</th>
<th>Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>338103523280</td>
</tr>
<tr>
<td>KeyGhost</td>
<td>338562656160</td>
</tr>
<tr>
<td>KeyKatcher Mini</td>
<td>338625304965</td>
</tr>
<tr>
<td>KeyKatcher Magnum</td>
<td>338421058298</td>
</tr>
</tbody>
</table>

- „Inline“ HKL can be detected using Time Measurement
  - Measure without HKL
  - Define Baseline (e.g. 338200000000)
  - Measure again
  - Win ;)

Fabian Mihailowitsch (cirosec GmbH)
Defeat PS/2 Hardware Keylogger

- Fill Keylogger memory via software
  - Some stop logging
  - Some overwrite memory at the beginning
  - Keystrokes are overwritten / not recorded

- Keyboard commands
  - Some commands lead to fake keypress (see Brute Force)
  - Send those repeatedly
  - ~100 logged keys in 10s
  - 109 minutes to fill 64kB

- Keyboard command „0xFE“
  - Resend
  - Keyboard responds by resending the last-sent byte
  - ~ 4 logged keys in 10 s

- Practical?
  - Most PS/2 HKL have a few KBytes memory
  - Nevertheless takes too much time
  - Works for: KeyGhost, KEYKatcher (some)
Defeat PS/2 Hardware Keylogger

- Stop HKL from sniffing keystrokes

- Keyboard sends scan codes
  - Make / Break codes
  - Defined in scan code set
  - Scan codes set can be chosen via KB command „0x0F“

- 3 scan code sets
  - 1: XT keyboards
  - 2: MF2 keyboard
  - 3: AT keyboards

- Tested Keyloggers support scan code set 2 and 3

- Choose scan code set 1...
  - Keylogger doesn’t log correctly
  - Logs can’t be used
  - New mapping scancode <-> keycode is necessary for OS
    - hdev
    - HAL
    - setkeycode
USB – How does it work

- Host controller + Hubs + devices build tree structure

- Device has various endpoints
  - Buffer in / out
  - Configuration via endpoint 0
  - Low Speed devices (Keyboard): endpoint 0 + 2 endpoints with 8 Bytes

- Only host controller manages communication with devices
  - Polls buffer (device configuration)
  - Writes buffer

- Data are transferred as packets

- Data transfer types
  - Isochronous transfer (guaranteed data rate, no error correction)
  - Interrupt transfer (small amount of data, retransmission)
  - Bulk transfer (big amount of data, retransmission)
  - Control transfer (device configuration, ACKed in both directions)
USB – How does it work

- Different device classes
  - Plug and Play
  - Relevant: HID class
  - Defines communication

- KB sends 8 Byte input report
  - Interrupt Transfer
  - Periodically polled by host
  - Contains pressed keys
  - No make / break codes
  - Packet:

<table>
<thead>
<tr>
<th>Modifier keys</th>
<th>OEM use</th>
<th>Keycode</th>
<th>Keycode</th>
<th>Keycode</th>
<th>Keycode</th>
<th>Keycode</th>
<th>Keycode</th>
<th>Keycode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Byte 7</td>
</tr>
</tbody>
</table>

Fabian Mihailowitsch (cirosec GmbH)
USB – How does it work

- PC sends 1 Byte output report
  - USB Control Transfer
  - Control LEDs
  - Packet:

<table>
<thead>
<tr>
<th>NUM Lock</th>
<th>Caps Lock</th>
<th>Scroll Lock</th>
<th>Compose</th>
<th>KANA</th>
<th>Constant</th>
<th>Constant</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- No additional KB commands
  - Transfer handled via USB
  - Typematic rate, etc. configured on PC
Detecting USB Hardware Keylogger

- Current Measurement
  - Like PS/2
  - More current is drawn
  - Cannot be measured by software
    - Device configuration contains current
    - However no accurate information available
Detecting USB Hardware Keylogger

- Brute Force KL password
- KeyCarbon: software to retrieve keystrokes
Detecting USB Hardware Keylogger

- Brute Force KL password
  - KeyCarbon: software to retrieve keystrokes
  - Software needs to communicate with KL...
- USB sniffer:

<table>
<thead>
<tr>
<th>Type</th>
<th>Seq</th>
<th>Time</th>
<th>Request</th>
<th>Request Details</th>
<th>Raw Data</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>0001</td>
<td>01:53:281</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>04</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0002</td>
<td>01:54:436</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>04</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0003</td>
<td>01:54:456</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>04</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0004-0003</td>
<td>01:54:471</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>04</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0005-0002</td>
<td>01:54:471</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>04</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0006</td>
<td>01:54:612</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0007</td>
<td>01:54:612</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0008-0007</td>
<td>01:54:612</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0009-0006</td>
<td>01:54:612</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0010</td>
<td>01:54:612</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0011</td>
<td>01:54:612</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0012-0011</td>
<td>01:54:623</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0013-0010</td>
<td>01:54:623</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>05</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0014</td>
<td>01:54:663</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0015</td>
<td>01:54:663</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0016-0015</td>
<td>01:54:663</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0017-0014</td>
<td>01:54:663</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0018</td>
<td>01:54:693</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0019</td>
<td>01:54:693</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0020-0019</td>
<td>01:54:693</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0021-0018</td>
<td>01:54:693</td>
<td>Control Transfer</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0022</td>
<td>01:54:693</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
<tr>
<td>URB</td>
<td>0023</td>
<td>01:54:693</td>
<td>Class Interface</td>
<td>Set Report (Output)</td>
<td>07</td>
<td>out</td>
</tr>
</tbody>
</table>
Detecting USB Hardware Keylogger

- Software needs to communicate with KL...
  - 1 Byte output reports (set LEDs)
  - Fixed header + HKL password + footer
  - Password char is encoded with 4 Bytes

- Brute Force (default) passwords
  - Create Lookup Table for PW chars
  - Perform attack via software
  - Works for: KeyCarbon models
Detecting USB Hardware Keylogger

- Changes to USB Properties / Topology
  - Keyboard only:

![Device Tree Diagram]

Fabian Mihailowitsch (cirosec GmbH)
Detecting USB Hardware Keylogger

- Changes to USB Properties / Topology
  - Keyboard + KeyCarbon:

![Device Tree Diagram]

Fabian Mihailowitsch (cirosec GmbH)
Detecting USB Hardware Keylogger

- Changes to USB Properties / Topology
  - Additional USB HUB if KeyCarbon is present

  „Why is the device undetectable, in practice, by software? The device shows up in Windows ‘Device Manager’ as a generic USB hub. This generic USB hub has no ID strings, and is indistinguishable from the generic USB hub found in 90% of all USB hubs. “

- Well...

<table>
<thead>
<tr>
<th>Device Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
</tbody>
</table>

USB HUB Controller used: Texas Instruments (TUSB2046B)
Detecting USB Hardware Keylogger

- Changes to USB Properties / Topology
  - KeyGhost changes device properties
    - USB Speed
      - Keyboard: \textit{bMaxPacketSize0} 08 / Speed: Low
      - KeyGhost: \textit{bMaxPacketSize0} 64 / Speed: Full
    - Device Status
      - Keyboard : Bus Powered (0x0000)
      - KeyGhost : Self Powered (0x0001)
  - More details later...
Detecting USB Hardware Keylogger

- Time Measurement
  - Like PS/2
  - HKL are placed inline -> introduces a delay
Detecting USB Hardware Keylogger

- **Time Measurement**
  - Basically the same idea like for PS/2
  - Has to be adjusted for USB

- **PC can send 1 Byte output report to KB (LED)**
  - sent as Control-Transfer
  - Control-Transfer are ACKed
  - Like PS/2 „ping“
  - Can be used for runtime measurement ;)

- **Implementation**
  - Send output report to KB
  - Wait until ACKed
  - Do it various times (10.000)
  - Measure runtime

- **Measurement can be performed from userland**
  - e.g. libusb
Detecting USB Hardware Keylogger

- Time Measurement
  - Results

<table>
<thead>
<tr>
<th>Setup</th>
<th>Milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>40034</td>
</tr>
<tr>
<td>KeyGhost</td>
<td>56331</td>
</tr>
<tr>
<td>KeyCarbon</td>
<td>43137</td>
</tr>
</tbody>
</table>

- USB HKL can be detected using Time Measurement
  - Create baseline for default setup (HUBs, etc.)
  - Measure again
  - Win ;)

Fabian Mihailowitsch (cirosec GmbH)
Detecting USB Hardware Keylogger

- Different keyboard behaviour
  - Normal behaviour:
    - Interrupt read (8 Byte): \x81\x06\x00\x22\x00\x00\x00\x04
    - Send USB Reset
    - Interrupt read (8 Byte): \x00\x00\x00\x00\x00\x00\x00
  
  - KeyGhost behaviour:
    - Interrupt read (8 Byte): \x81\x06\x00\x22\x00\x00\x00\x04
    - Send USB Reset
    - Interrupt read (8 Byte): \x81\x06\x00\x22\x00\x00\x00\x04
Detecting USB Hardware Keylogger

- Different keyboard behaviour
  - Analysis on the wire...
  - Reason: keyboard never receives USB Reset
Detecting USB Hardware Keylogger

- Keyboard never receives USB Reset

- USB single-chip host and device controller (ISP1161A1BD)
  - Acts as Device for PC (causes changes to device properties)
  - Acts as Host Controller for KB

- Behaviour can be tested via software
  - e.g. libusb

- Note: Time Measurement for this design bug is possible too
Conclusion

- **PS/2**
  - All tested models were placed „inline“
  - Time Measurement as general technique to detect them
  - Scancode 1 as general technique to defeat them

- **USB**
  - Detection via USB behaviour (USB speed, etc.)
  - Individual bugs
  - More research to come...

- **All tested HKL contained bugs that can be used to detect them**
  -Generic and individual bugs
  - Each HKL has to be analyzed separately
  - Bugs can be combined (Pattern)

- **PoC code**
  - Soon: [https://code.google.com/p/hkd/](https://code.google.com/p/hkd/)
Thank you for your interest!

Questions and Feedback