Key-Logger, Video, Mouse

How to turn your KVM into a raging key-logging monster
MEET THE TEAM

Yaniv Balmas
"This should theoretically work"
Security Researcher
Check Point Software Technologies

Lior Oppenheim
"The mad scientist"
Security Researcher
Check Point Software Technologies
TOOMANYCOMPUTERS

- Computers
- More computers
- A LOT OF COMPUTERS
WHAT IS KVM?

• **Keyboard, Video, Mouse**

• KVM Connects the same Keyboard, Video and Mouse to one or more computers.
KVM Evolution

1981
`A-B Switch`

2002
Desktop KVM

2015
Enterprise KVM
WHERE ARE THEY?

• On top of your server racks.
• On your desktop.
• In your security centres.

KVMS ARE EVERYWHERE!!
Introducing Gen-KVM
ITRUNSCODE

- On screen configuration display.
- Configurable hot-keys.
- Control device functionality through keyboard.

Exploitable?
First Attempt
(Funny meme here)
SOFTWARE

• Opening the KVM box.
• Manuals, Cables, Warranty and CD…

• CD contains **A Firmware Upgrade Utility**!

• Can the firmware be extracted from the upgrade utility?!
• Since x86 is no new territory, we can reverse engineer this!
MEET THE BLOB

Low Entropy

No Strings

Undetermined Freq. Analysis
• Firmware upgrade process is done via a custom serial connection.

• It is possible to extract the (possibly) decoded firmware binary from the serial protocol.

• It's just a matter of analyzing the serial protocol.
PCBLAYOUT

PLD X2

8052 X1

External RAM X1

Unknown X2
UART MAGIC

- 8051\2 Chips have an integrated UART port.
- Which IC pins should be tapped?
- If we find out, the firmware could be extracted using simple LOGIC.
• 30-45 China mail shipping days later.

• We can finally use LOGIC.
• Tapping the 8052 IC UART pins using Logic Analyzer.

• Reveals the UART port’s signals.
SIGNAL ANALYSIS

- Reviewing the signals in the UI.
- An obvious pattern emerges.
GREAT SUCCESS?

Looks Familiar?
GREAT FAIL!
• The BLOB is probably translated to 8051 Assembly.

• The translation is done somewhere within the 8052 chip.

• It might be possible to break the obfuscation!
Last XX Bytes are padded with 0x53
BREAKING CODE

8051 NOP = 0x00

0x53 ⊕ 0x53 = 0x00
ALLDONE!
8051 ASSEMBLY?
8051 Assembly?
EVERYTHING IS 8051!!!
BREAKING CODE

Final 8 Bytes are different.
ACLUE?

• What does these last 8 bytes mean?
• Are they a clue left for use by a mad embedded developer?
• If we could just get some more data…
FIRMWARE DIFFS!

- We have only analyzed a single firmware version.
- Perhaps other firmware versions could be insightful.

<table>
<thead>
<tr>
<th>Last 8 Bytes</th>
<th>Firmware Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>91 99 99 89 91 B2 99 00</td>
<td>3 3 3 1 2</td>
</tr>
<tr>
<td>B2 92 89 81 A1 99 A1 89</td>
<td>4 1 4 0 1</td>
</tr>
<tr>
<td>92 00 A1 A1 89 B2 89 91</td>
<td>4 2 4 1 1</td>
</tr>
<tr>
<td>91 92 A1 89 A1 A1 B2 00</td>
<td>4 2 4 4</td>
</tr>
<tr>
<td>B2 A1 A1 89 A9 00 92 91</td>
<td>4 2 4 1 5</td>
</tr>
<tr>
<td>A1 92 00 89 B1 91 A1 B9</td>
<td>4 2 4 1 6</td>
</tr>
<tr>
<td>92 00 A1 89 91 B2 A1 89</td>
<td>4 2 4 1 7</td>
</tr>
<tr>
<td>00 A1 92 91 C1 B2 A1 89</td>
<td>4 2 4 1 8</td>
</tr>
<tr>
<td>00 91 A1 B2 C9 89 A1 92</td>
<td>4 2 4 1 9</td>
</tr>
</tbody>
</table>
A PATTERN?

- Listing the binary values of these “patterns” from all firmware versions.
- If only these were ASCII values…

<table>
<thead>
<tr>
<th>Value</th>
<th>Hex</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x89</td>
<td>10001001</td>
</tr>
<tr>
<td>2</td>
<td>0x91</td>
<td>10010001</td>
</tr>
<tr>
<td>3</td>
<td>0x99</td>
<td>10011001</td>
</tr>
<tr>
<td>4</td>
<td>0xA1</td>
<td>10100001</td>
</tr>
<tr>
<td>5</td>
<td>0xA9</td>
<td>10101001</td>
</tr>
<tr>
<td>6</td>
<td>0xB1</td>
<td>10110001</td>
</tr>
<tr>
<td>7</td>
<td>0xB9</td>
<td>10111001</td>
</tr>
<tr>
<td>8</td>
<td>0xC1</td>
<td>11000001</td>
</tr>
<tr>
<td>9</td>
<td>0xC9</td>
<td>11001001</td>
</tr>
</tbody>
</table>
They could be!

- If we shift the bits 3 positions to the right.
- We get our ASCII values!

<table>
<thead>
<tr>
<th>Value</th>
<th>Hex</th>
<th>Binary</th>
<th>ROR 3</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x89</td>
<td>10001001</td>
<td>00110001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0x91</td>
<td>10010001</td>
<td>00110010</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0x99</td>
<td>10011001</td>
<td>00110011</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0xA1</td>
<td>10100001</td>
<td>00110100</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0xA9</td>
<td>10101001</td>
<td>00110101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0xB1</td>
<td>10110001</td>
<td>00110110</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0xB9</td>
<td>10111001</td>
<td>00110111</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>0xC1</td>
<td>11000001</td>
<td>00111000</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>0xC9</td>
<td>11001001</td>
<td>00111001</td>
<td>9</td>
</tr>
<tr>
<td>Position</td>
<td>Original</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SUCCESS!!!

Strings!

Assembly!
We can now design our own “custom” firmware-upgrade utility.

However, we do need a basic understanding of 8051 Assembly!
**8051 REVIEW**

+ Only 255 OP-Codes, and ~40 Instructions.

- Functions are not *really* functions.

- Just a single memory access register.

- Registers keep on changing for some reason.
KVM LOGIC

Keyboard Emulation

HID Parsing

Keyboard LEDs Control

Hotkeys Handling