Technical Note

Security assessment of Raspberry Pi boot sequence

Author:	Marc Durvaux
Version:	1.0
Date:	08/01/2015

1 Abstract

This technical note reports the results of an analysis of the Rapsberry Pi boot sequence, which aims at identifying a possible way of injecting malicious code during this phase. Under the assumption that the attempt of malicious code injection is performed through the connection of USB peripheral, the analysis shows that the risk is extremely limited given that the ROM boot sequence does not connect another device than a keyboard.

2 Revision history

Version	Date	Author	Description
1.0	08/01/2015	Marc Durvaux	Initial release

3 Table of content

4 Reference documents

(1) Raspberry Pi Foundation
http://www.raspberrypi.org/trademark-rules/
http://www.raspberrypi.org/about/
http://www.raspberrypi.org/documentation/

(2) Raspberry Pi architecture
http://meseec.ce.rit.edu/551-projects/fall2013/2-1.pdf

(3) Raspberry Pi Hardware
http://elinux.org/index.php?title=RPi_Hardware

(4) Raspberry Pi Documentation
https://github.com/raspberrypi/documentation
https://github.com/raspberrypi/documentation/tree/master/hardware/raspberrypi/schematics

(5) Raspberry Pi boot process
http://wiki.beyondlogic.org/index.php?title=Understanding_RaspberryPi_Boot_Process
Note: although this link provides useful information, the tests have shown that it is not completely accurate (e.g. the third stage, start.elf, is loaded even if the file bootcode.bin is missing)

(6) Raspberry Pi ARM based bare metal examples
https://github.com/dwelch67/raspberrypi

(7) NOOBS (New Out of Box Software) documentation
https://github.com/raspberrypi/noobs

5 Software and Hardware configuration

The tests have been performed under Raspbian installed by NOOBS v 1.3.10.
The hardware platform was a Raspberry Pi model B.

Signal probes have been connected to the following SD connector pins:

Pin	Signal (One-Bit SD Bus Mode)	Signal (Four-Bit SD Mode)	Logic analyzer channel
9	Unused	SD Serial Data 2	1 - Brown
1	Card / Non-SPI Mode Detection	SD Serial Data 3	2 – Red
2	Command / Response	Command / Response	3 - Orange

A breakout box has been inserted between the Raspberry Pi USB host connector and the USB peripheral (keyboard or storage key), to allow for probing the D+ signal on the logic analyzer channel 0 (white)

6 Tests performed

6.1 Verification of required boot files

Note: A backup directory has been created in the BOOT volume on the SD card in order to deactivate selected files without erasing them.

Reference (6) seems to provide the more accurate information on the required boot files in the “README” file :

```
From what we know so far there is a gpu on chip which:

1) boots off of an on chip rom of some sort
2) reads the sd card and looks for additional gpu specific boot files
bootcode.bin and start.elf in the root dir of the first partition
(fat32 formatted, loader.bin no longer used/required)
3) in the same dir it looks for config.txt which you can do things like
change the arm speed from the default 700MHz, change the address where
to load kernel.img, and many others
4) it reads kernel.img the arm boot binary file and copies it to memory
5) releases reset on the arm such that it runs from the address where
the kernel.img data was written

The memory is split between the GPU and the ARM, I believe the default
is to split the memory in half.  And there are ways to change that
split (to give the ARM more).  Not going to worry about that here.

From the ARMs perspective the kernel.img file is loaded, by default,
to address 0x8000.  (there are ways to change that, not going to worry
about that right now).
```

<u>Test results :</u>

- when the file “bootcode.bin” is missing, the boot process proceeds further.
- when the file “start.elf” is missing, the boot process hangs.

<u>Conclusion :</u>

For the following tests, when an aborted boot is required, both “bootcode.bin” and all “start*.elf” files will be moved to the backup directory
(* denotes a wildcard, the refer to the following files : start.elf, start_x.elf, start_cd.elf)

6.2 Observed activity during the boot process

Note : the timings measured from the power-up include the settling of the power supply and are measured manually. Hence the estimated precision is about 0.5 s.

6.2.1 Sequence from power-up to aborted boot

- T0 + 3.8 s : start of SD card access (figure 1, channels 1 to 3)
- T0 + 11 s : short USB activity (figures 2 and 3, channel 0)
  - figure 2 shows the activity when a USB keyboard is connected
  - figure 3 shows the activity when a USB key, formatted in VFAT is connected.

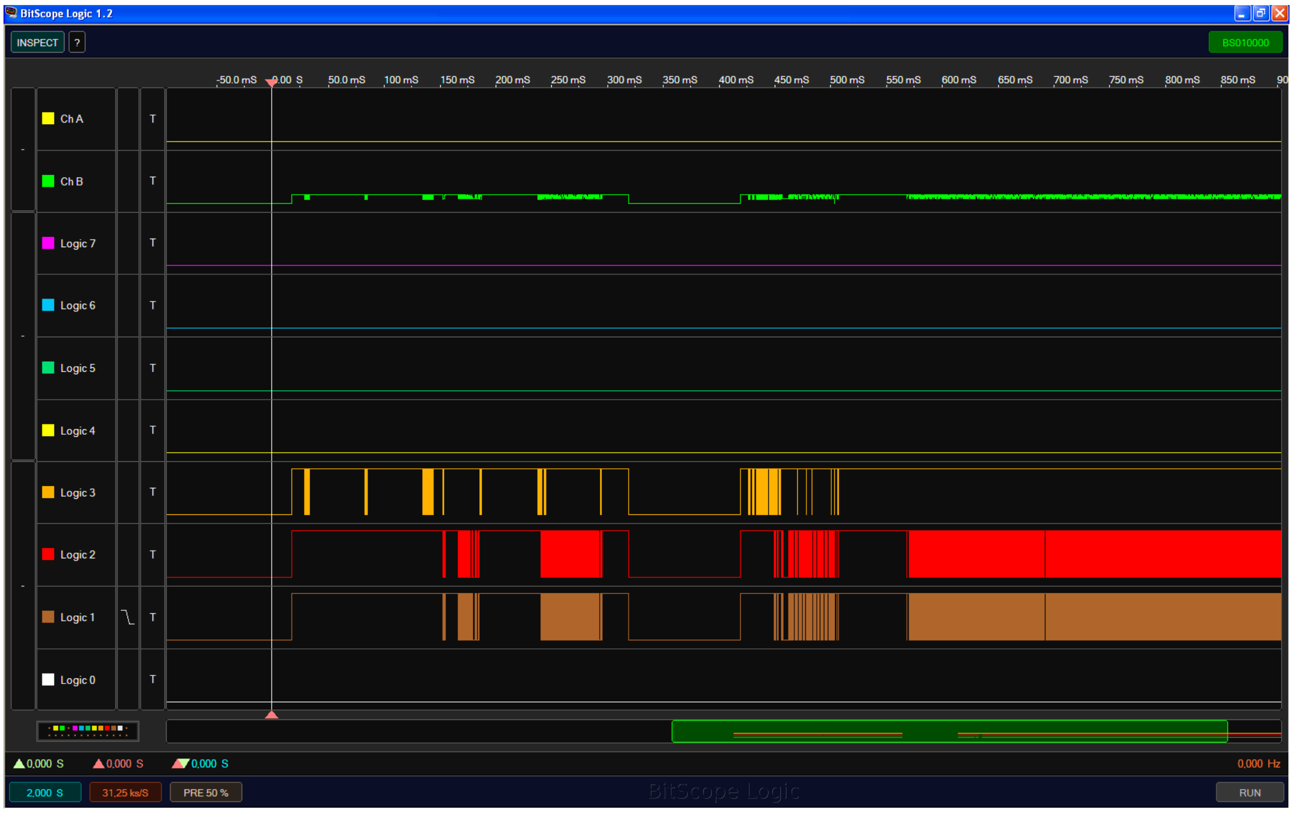

figure 1

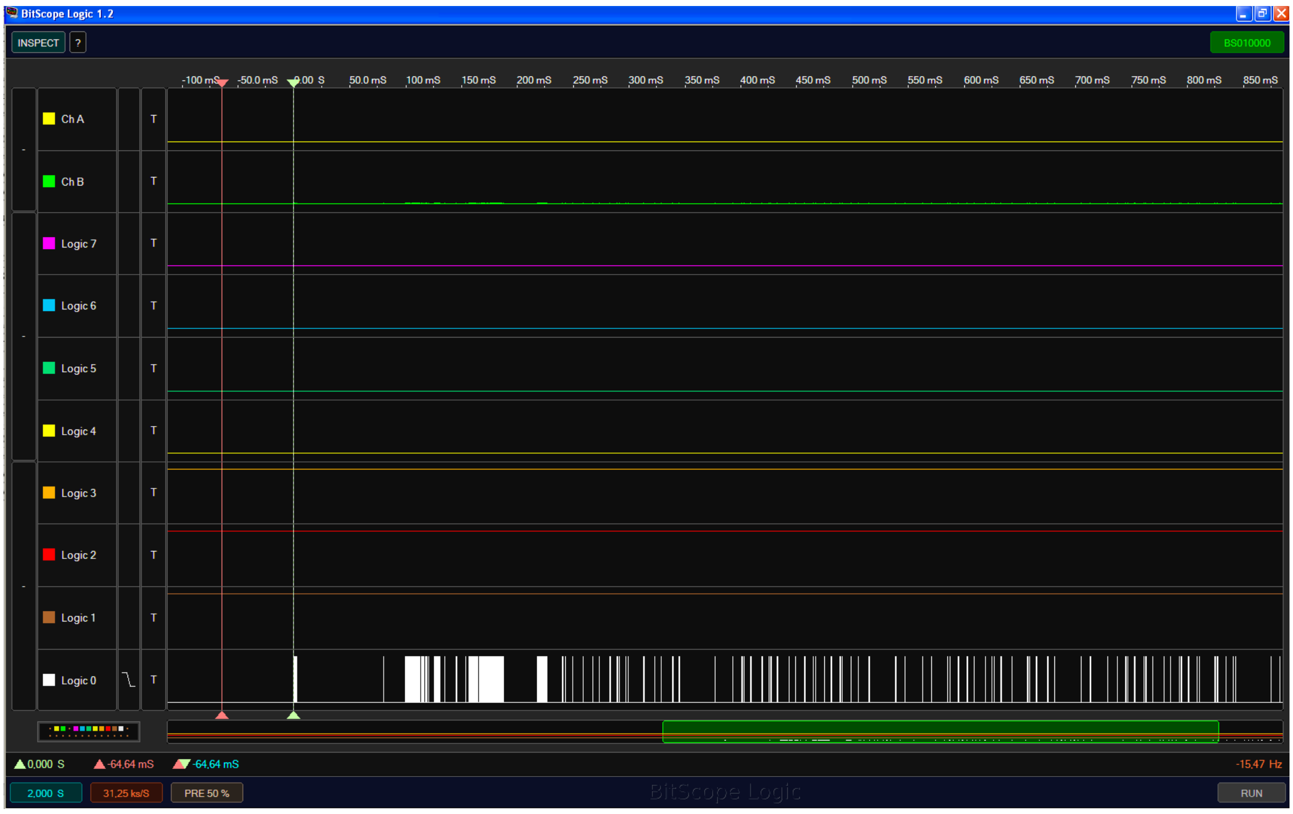

figure 2

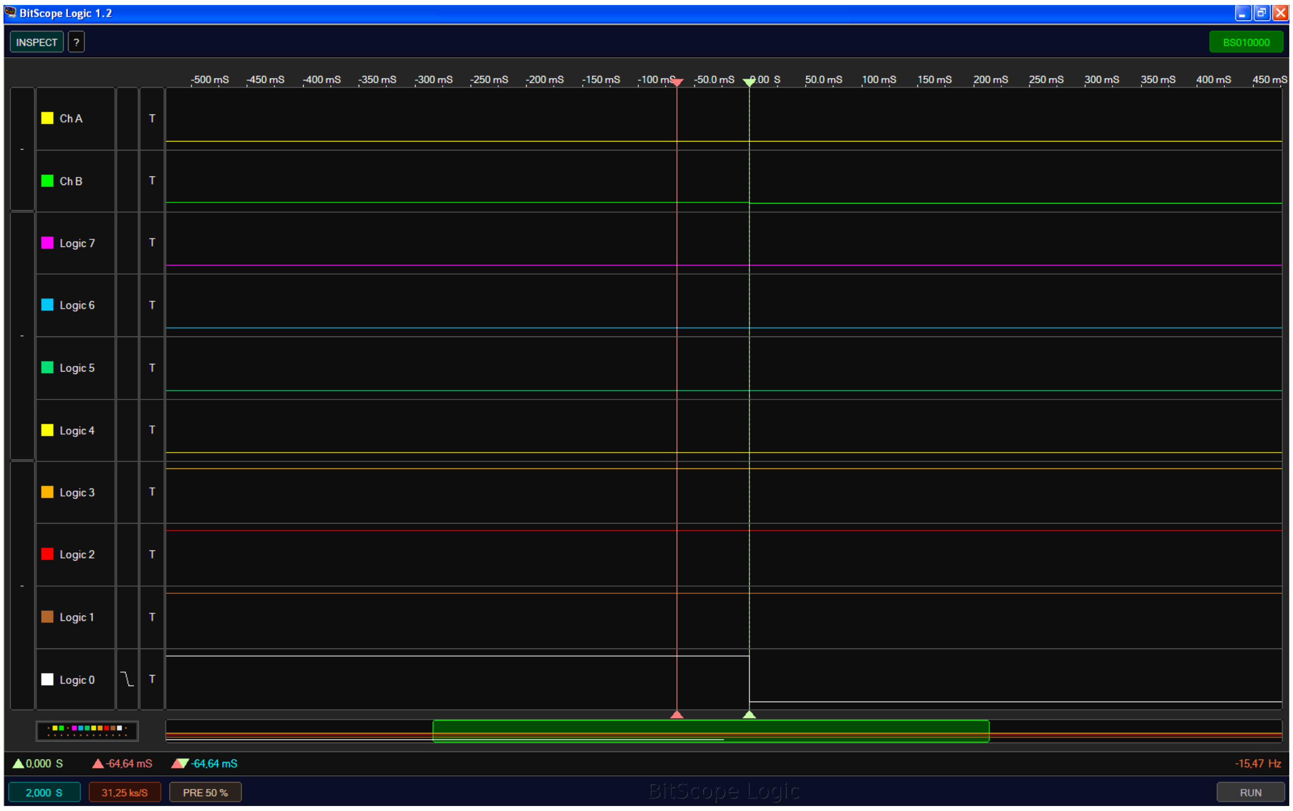

figure 3

6.2.2 Sequence from power-up to boot

• T0 + 3.8 s :	start of SD card access
• T0 + 11 s :	short USB activity
• T0 + 12 s :	start of Linux boot

The recorded waveforms do not differ from above figures 1 to 3.

7 Conclusion

The tests performed in section 6 show that the only USB activity detected during the boot phase, prior the operating system boot, is related to the keyboard.
From the Raspberry Pi documentation (reference 7), this is related to the optional boot selector. Hence there is no identified opportunity to launch a malicious code located on a USB key during the boot sequence.