A Side-channel Timing Attack of the MSP430 BSL

Travis Goodspeed
EMC^2, Oak Ridge National Lab
<travis@utk.edu>

June 28, 2008

Abstract

This paper presents a side-channel timing attack in the MSP430 serial bootstrap loader (BSL). Version 2.12, the latest available to the author at the time of this writing, is the only version known to be vulnerable.

1 Summary

The Texas Instruments MSP430 low-power microcontroller is used in many medical, industrial, and consumer devices. It may be programmed by JTAG or a serial bootstrap loader (BSL) which resides in masked ROM.

By design, JTAG may be disabled by blowing a fuse. The BSL may be disabled by setting a value in flash memory. When enabled, the BSL is protected by a 32-byte password. If these access controls are circumvented, a device's firmware may be extracted or replaced.

In version 2.12 of the BSL, as found in revision G of the MSP430-FG4618, a password comparison routine suffers from unbalanced timing, such that processing an incorrect password takes two clock cycles longer than a correct byte. By observing external timing, it is possible to determine the correctness of individual bytes, drastically reducing the amount of time required to guess a password.

This vulnerability has been verified in simulation, but it has yet to be exploited in hardware.

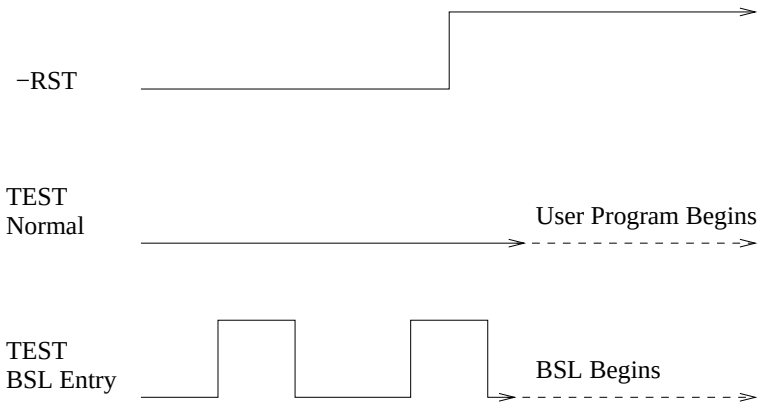

Figure 1: BSL Entry Sequence (Chips w/ Shared JTAG Pins)

2 Serial Bootstrap Loader (BSL)

The BSL of the MSP430 resides in masked ROM. If an entry sequence is performed, as is depicted in Figure 1, the BSL–rather than the user application–is run. The sequence involves two rising edges on the TEST pin preceding the rising edge of the -RST pin that power on the chip. Further, the TEST pin must be high on the rising edge of -RST. For those chips with dedicated JTAG pins, the same sequence is the same except that falling edges are sent on the TCK pin.

As the BSL continues to function after the JTAG fuse has been blown, those wishing to protect either firmware or cryptographic keys will often use the BSL for firmware updates. Each firmware image contains a password, and without that password little more can be done with the BSL than erasing all of memory.

Once the BSL has loaded, commands are accepted through a serial port. While there are many commands, the one of interest here is RX Password, which must preceed any attempt to read (TX Data) or write memory (RX Data). Mass Erase, which bulk-erases all of memory, requires no password by default.

3 Password Composition

The BSL password is the Interrupt Vector Table (IVT) of the chip, which resides at the top of memory and is composed of sixteen 16-

bit pointers to interrupt handlers. Of these 256 bits, the authors of *Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks*[1] conclude that 40 are random. They then calculate that a brute force would take 128 years for a guaranteed break. I've since reduced this to 32 years in *MSP430 BSL Passwords: Brute Force Estimates and Defenses*[2]. There might be room for further reduction, but the time required will never be so short as to be practical. Further, the method used to reduce the brute forcing time to the order of decades is only applicable to versions 1.60 and 1.61 of the BSL.

For more information on brute forcing and password reliability, see the aforementioned papers for risk assessments and a perl script which randomizes the interrupt table of a firmware image.

4 Timing

In versions prior to 2.12, every byte of the password is compared in an if/else statement with balanced timing. An example of balanced timing, which comes from BSL 2.01 of the MSP430F2274, is shown in Figure 2. In this control flow graph, edges are marked with their delay in cycles. The BIS[3] instruction on the left branch marks a bit of r11 to invalidate access, while the right branch does nothing. In doing nothing, however, the right branch was carefully designed to take exactly as many cycles as the left.

Version 2.12's comparison function, as shown in Figure 3, is different. It is unbalanced in that one branch takes two cycles longer than the other to execute. As this code is part of a loop and the longer path is that of an incorrect byte, the timing of this program will be retarded by two cycles for every incorrect byte.

5 Simulation

To demonstrate this in simulation, the author has written a C program for the MSP430 that wraps the BSL within an MSP430 simulator. The image was run 256 times, guessing passwords of every possible byte repeated. Timing was observed and recorded.

[1] by Alexander Becher, Zinaida Benenson, and Maximillian Dornseif

[2] Posted June '08, http://travisgoodspeed.blogspot.com/

[3] Bit Immediate Set

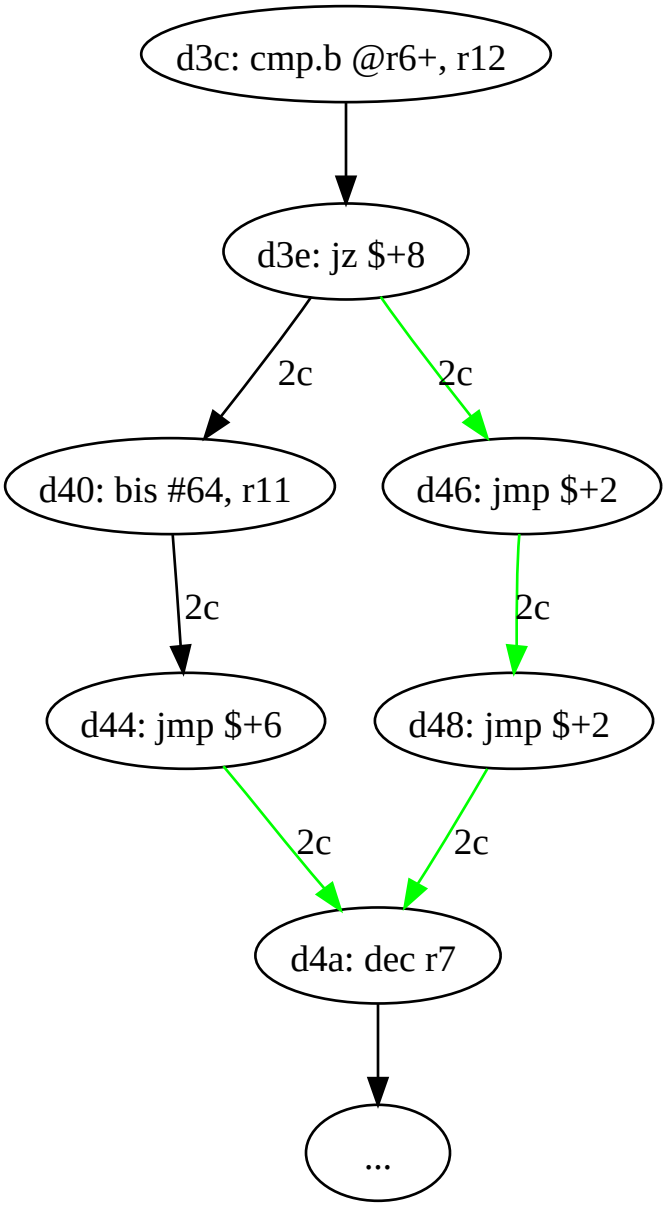

Figure 2: Byte Comparison in BSL 2.01

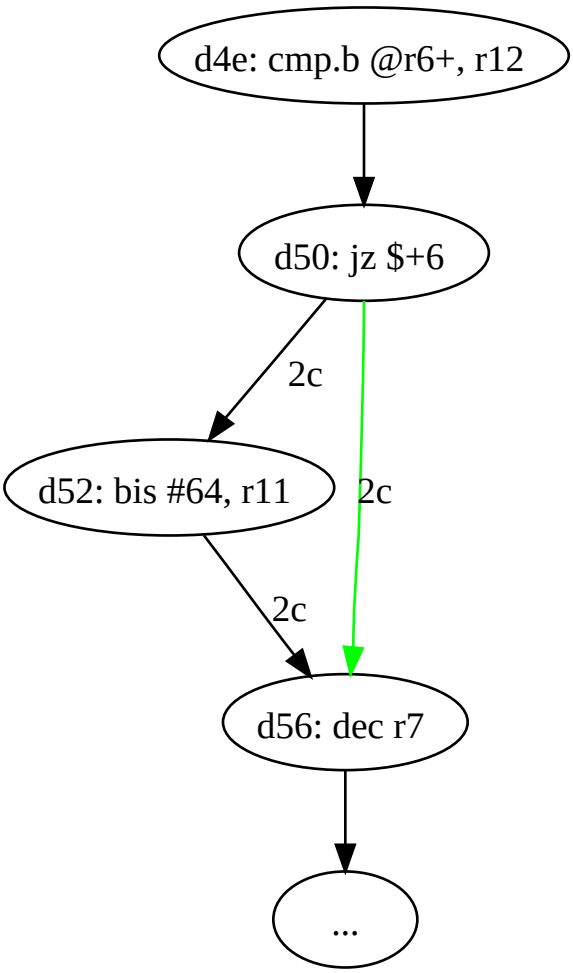

Figure 3: Byte Comparison in BSL 2.12

Shortened runtimes were found for repetitions of 0x00, 0x11, and 0x3A. Compared to an average (mode) runtime being 6543 cycles, a password of 0x00 repeated took only 6541 cycles to complete, a difference of 2 cycles. A password of 0x11 repeated took 6511 cycles, while 0x3A repeated took 6513 cycles. Thus the offsets were as shown in Table 1.

Guess	Cycles	Δ	$\Delta/2$
00*	6541	2	1
11*	6511	32	16
3A*	6513	30	15
all others	6543	0	0

Table 1: Runtimes of MSP430 BSL Wrapper

The rightmost column of the table gives us the frequencies of each byte within the BSL. There must be a single 0x00, sixteen 0x11, and fifteen 0x3A bytes. As the less significant byte, being of an aligned instruction address, must be even, 0x11 is likely the more significant byte of each of 16 fields. Thus we have fifteen vectors of 0x113A and one of 0x1100. As the reset vector always points to the bottom of flash, it is 0x1100 and the rest are 0x113A. This Sherlock Holmes rundown is quite unnecessary, however, as the timing has already exposed enough of the password to break it. The BSL password is shown in Table 2.

0x1100	0x113A
0x113A	0x113A
0x113A	0x113A
0x113A	0x113A
0x113A	0x113A
0x113A	0x113A
0x113A	0x113A
0x113A	0x113A

Table 2: Password of MSP430 BSL Wrapper

6 Exploitation

The BSL runs at 1mhz until clocked higher, and a modern MSP430 can be clocked as high at 16mhz. Therefore, a 16mhz MSP430 is quite capable of the timing necessary to break the password of a vulnerable chip. To that end, the author is presently designing a handheld device based upon the MSP430-F2012 for a hardware demonstration.

There are some complications, however. First, the BSL's timing is not hard-coded, but rather comes from a tare routine.[4] This routine calibrates the bit-banging serial port handler by observing the timing of a header byte received by the '430. This header byte, 0x80, must be sent with perfectly regular timing if measurements are to be valid. Any drift will affect the execution time of the password comparison.

Hardware to exploit this vulnerability is quite simple. The 2012 chip is powered by JTAG, which doubles as a method of communicating with the operator's workstation. A BSL cable then connects to the victim board. Three LEDs display status and two buttons allow for user input. To facilitate untethered operation, broken keys are stored in the flash memory of the device for later retrieval by JTAG. You'll find a schematic diagram in Figure 4.

Once completed, the device's software will use a bit-banged serial port and the system timers to observe the moments of response that are found in the target device. Different firmware could serve to disable the BSL in vulnerable field units, but TI's MSP430 BSL Replicator[5] might be better suited for such a purpose, as it has sufficient memory to store a complete replacement firmware for the target device.

7 Conclusion

For applications requiring BSL password protection, the author recommends that BSL Version 2.12 be disabled entirely. To determine the version of BSL on a particular chip, read the two bytes of memory beginning at 0xFFA. 0xFFA is the major version number, and 0xFFB is the minor version number. Both are in BCD, so 0x02 0x12 means 2.12, not 2.18.

The BSL may be disabled by setting the BSLKEY variable to

[4] See 0xE86 of BSL 2.12 in Rev. G of the MSP430FG4618.

[5] described in SLAA089

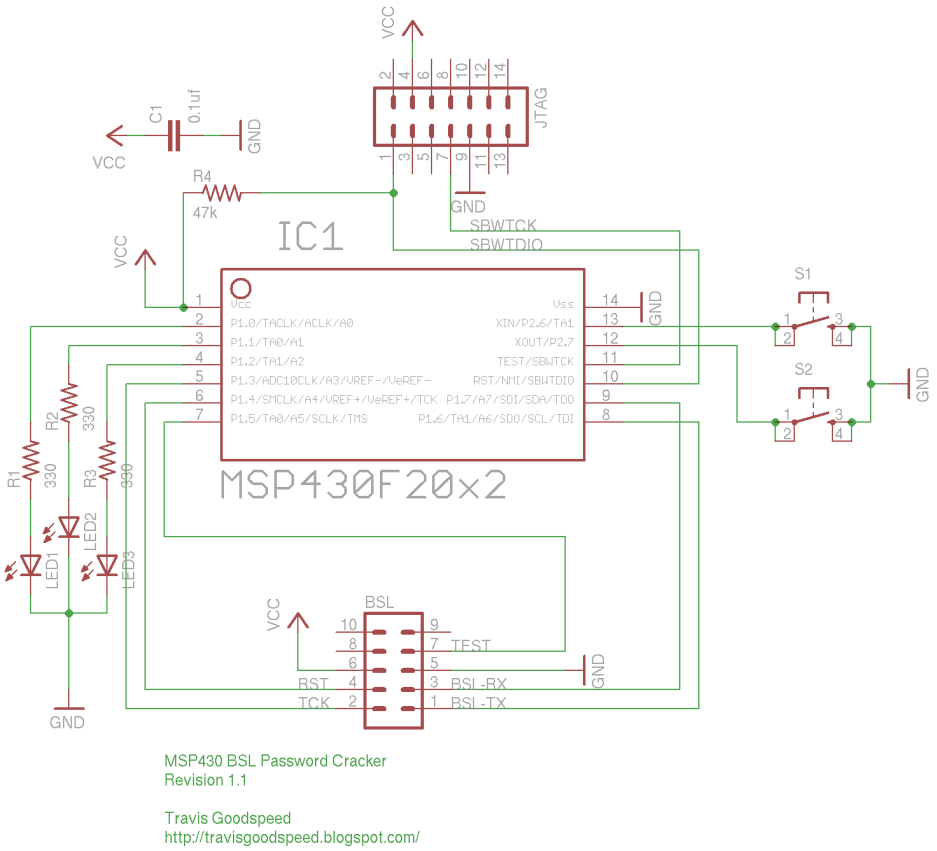

Figure 4: BSLCrack 1.1

0xAA55. As the location of this variable varies, see the datasheet of your particular chip for the address of BSLKEY.

The official documentation of the MSP430 BSL is to be found in SLAA089 from Texas Instruments. Various articles on MSP430 security can be found at the author's website,
`http://travisgoodspeed.blogspot.com/`