Side Channel Timing Attacks
on MSP430 Microcontroller Firmware

by Travis Goodspeed
EMC^2
Oak Ridge National Lab
<travis at utk.edu>
A timing vulnerability exists in version 2.12 of the MSP430's BSL which allows the BSL password to be guessed by an attacker.
Prospectus

- JTAG and Spy-Bi-Wire
- BSL -- Serial Bootstrap Loader
- Manufacturing Considerations
- BSL Details
- Password
- Brute Force Attack
- Timing Attack
  - Theory
  - Simulation
  - Hardware
What is the MSP430?

• 16 bit RISC MCU
• Ultra-Low Power
  – 1µs clock startup
  – 0.8 µA standby
  – 250 µA/MIPS
• Usage
  – Low Power Wireless
  – Medical
EZ430 Kit

- $20 to $50
- Debugger/Carrier
- Target Boards
  - 2012 with an LED
  - 2274 with a Radio
JTAG and Spy-Bi-Wire

- Programming
- Debugging
  - Registers
  - Single-Stepping
  - Breakpoints
- Fuse Protection

<table>
<thead>
<tr>
<th>JTAG</th>
<th>Testpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcc</td>
<td>2 1</td>
</tr>
<tr>
<td>Vcc ext</td>
<td>4 3</td>
</tr>
<tr>
<td>TEST/Vpp</td>
<td>6 5</td>
</tr>
<tr>
<td>TDO</td>
<td>7</td>
</tr>
<tr>
<td>TDI</td>
<td>6</td>
</tr>
<tr>
<td>TMS</td>
<td>5</td>
</tr>
<tr>
<td>TCK</td>
<td>4</td>
</tr>
<tr>
<td>GND</td>
<td>3</td>
</tr>
<tr>
<td>RESET</td>
<td>2</td>
</tr>
<tr>
<td>nc</td>
<td>1</td>
</tr>
<tr>
<td>JTAG</td>
<td>14 13</td>
</tr>
</tbody>
</table>
JTAG and Spy-Bi-Wire
**BSL: Serial Bootstrap Loader**

- Just for Programming
- Serial Protocol
- Password Protected
- Disabled by Flash Token
  - Ignores JTAG Fuse

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>BSL-TX</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCK</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>4</td>
<td>3</td>
<td>BSL-RX</td>
</tr>
<tr>
<td>Vcc</td>
<td>6</td>
<td>5</td>
<td>GND</td>
</tr>
<tr>
<td>Vcc Ext</td>
<td>8</td>
<td>7</td>
<td>TEST</td>
</tr>
<tr>
<td>nc</td>
<td>10</td>
<td>9</td>
<td>nc</td>
</tr>
</tbody>
</table>
Initializing the BSL

Reset

User Program Begins

Test

BSL Begins

Test

TCK
BSL Commands

- RX Data
- RX Pass
- Erase Segment
- Mass Erase
- Erase Check
- Change Baud
- Load PC
- TX Data
- TX BSL Version

- RX Pass
- Mass Erase
- Erase Check
- Change Baud
- TX BSL Version
TX BSL Version

- Chip Info
- BSL Version

- F4 6F
- 32 40
- 00 00
- .. ..
- 00 00
- 02 12
Mass Erase

• Erases all of memory.
  – Resets password.
  – Nothing left to steal.
Change Baud

- Unprotected
  - BSL 1.60 and 1.61
- F1xx/F2xx
  - DCOCTL
  - BCSCCTL1
- F4xx
  - SCFI0
  - SCFI1

- Untrusted Data
- Control Registers
RXPass

- Receives BSL Pass
- Unlocks if correct.
BSL Password

- 256-bit Interrupt Vector Table
- 16 Vectors
  - 16 bits
  - pointers to interrupt handlers
- How many bits are actually random?
  - At least 40.
  - Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks
    - Becher, et al
<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$16 \times 16$</td>
<td>$256$ Bits</td>
</tr>
<tr>
<td>$15 \times 15$</td>
<td>$225$ Bits</td>
</tr>
<tr>
<td>$4 \times 15$</td>
<td>$60$ Bits</td>
</tr>
<tr>
<td>$4 \times 10$</td>
<td>$40$ Bits</td>
</tr>
</tbody>
</table>
Brute Forcing 40 Bits, Becher

• 12 Pass/Sec at 9600
  – 31 Pass/Sec at 38400
  – 81 Pass/Sec with modified behavior
  – Round up to $2^7$
• $2^{(40-7-1)}$ seconds
  – $2^{32}$ seconds
  – 128 years
# How Change Baud Rate Works

<table>
<thead>
<tr>
<th>baud</th>
<th>mhz</th>
<th>years</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600</td>
<td>1.05</td>
<td></td>
</tr>
<tr>
<td>19200</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td>38400</td>
<td>4.2</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>32</td>
</tr>
</tbody>
</table>
Brute Forcing 40 Bits, Goodspeed

- 12 Pass/Sec at 9600
  - 31 Pass/Sec at 38400
  - 81 Pass/Sec with modified behavior
  - Round up to $2^{^7}$
- $2^{^(40-7-1)}$ seconds
  - $2^{^32}$ seconds
  - 128 years
- Reclock from 4mhz to 16mhz
  - $2^{^30}$ seconds
  - 32 years
Becher's Password Fix

- Perl Script
- Randomizes Interrupt Vectors

```
FFE0  1100  \rightarrow  handler  1100
FFE0  3A62  \rightarrow  BR 0x1100  3A62
               \downarrow
               handler  1100
```
Password Comparison
MSP430F1612

- for( i=0; i<32; i++ )
  - b=getbyte();
  - if( b!=IVT[i] )
    - access=d愠ined;
  - else
    - wait;
Password Comparison
MSP430F2274

- for( i=0; i<32; i++ )
  - b=getbyte();
  - if( b!=IVT[i] )
    • access=denied;
  - else
    • wait;

0xd3e to 0xd4a
Password Comparison
MSP430FG4618

- for( i=0; i<32; i++)
  - b=getbyte();
  - if( b!=IVT[i] )
    - access=died;

0xd50 to 0xd56
Simulation

- MSP430simu
- C Wrapper
  - Patches BSL
- Tested 256 Passwords
## Simulation Results

<table>
<thead>
<tr>
<th></th>
<th>Cycles</th>
<th>Δ</th>
<th>Δ/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>others</td>
<td>6543</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>00*</td>
<td>6541</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11*</td>
<td>6511</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>3A*</td>
<td>6513</td>
<td>30</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IVT</th>
</tr>
</thead>
<tbody>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>113A</td>
</tr>
<tr>
<td>1100</td>
</tr>
</tbody>
</table>
Why was this done?

- Space Constraints?
  - C00 to FFF
  - 1024 Bytes
- Feature Creep
- Instruction Set

10 bytes
12 bytes
6 bytes
1612
2274
4618
Feature Creep

- 1.60
  - TX BSL Version
  - Change Baud
- 2.01
  - Change Baud Fixed
- 2.12
  - Set Memory Offset
  - 20-bit Extensions
Instruction Set Change

- MSP430
  - 16-bit architecture
- MSP430X
  - 20-bit address extension
  - Prefix Extension Word
BSLCrack

- Low Latency
  - 1 mhz target clock
  - 2 cycles of drift
- MSP430F2012
- 3 LEDs
- 2 Buttons
**BSLCrack Software**

- Verify Version
- False Password
  - 0x0001 repeated
- Byte Guesses
  - 0x00 repeated,
  - 0x01 repeated,
  - ...
- Enumerate Remainder
BSLCrack

- **Results**
  - Stored in ROM
  - Indicated by LED
  - Retrieved by JTAG
- **Construction**
  - No SMD Components
  - Breadboardable
Concluding Remarks

<travis at utk.edu>
http://travisgoodspeed.blogspot.com/
Questions?

<travis at utk.edu>
http://travisgoodspeed.blogspot.com/