Mathew Rowley

How many bricks does it take to crack a microcell?

http://67.219.122.21/blackhat2012/

matasano SECURITY

Mathew Rowley
@wuntee

- I hate hearing peoples backgrounds... But this is an exception.

- Senior Security Consultant at Matasano Security
- Computer science background - aka software guy
- This talk is is a hardware -> software talk

agenda
my epic battle...

- Focus on different aspects of reversing, not GSM/3G
- Device background
- wuntee vs the network
- wuntee vs the cage
- wuntee vs hardware
  - debug pins, SPI, JTAG, Serial
- wuntee vs software
  - UBoot, Kernel, Firmware

matasano SECURITY

how does the device work?

- Everyone know what a microcell is?
- Web based interface to provision phone numbers that can connect to the device
- Configuration somehow pushed to microcell
- Only those phone numbers can connect

Why?

- Dear Mathew, our cell service sucks - heres something for free that can do cool things
- Was working at Interpidus Group - focus on mobile security
- I do not know much about hardware stuff - have always wanted to learn

wuntee vs the network round 1

network communication

- Routed all traffic through a server running DHCP
- tcpdump shows
  - HTTPS traffic
  - IPSec tunnel
  - Multicast stuff
- MITM with Mallory?

matasano SECURITY

in the 1st round, with a TKO, the winner is....

the network

matasano SECURITY

wuntee vs the cage round 1

disassembly

- 2 screws under the bottom orange part
- Orange part comes off
- Two side panels come off
- Single board connected to the grey portion
- Rip them all off!!
- Can I boot?

matasano SECURITY

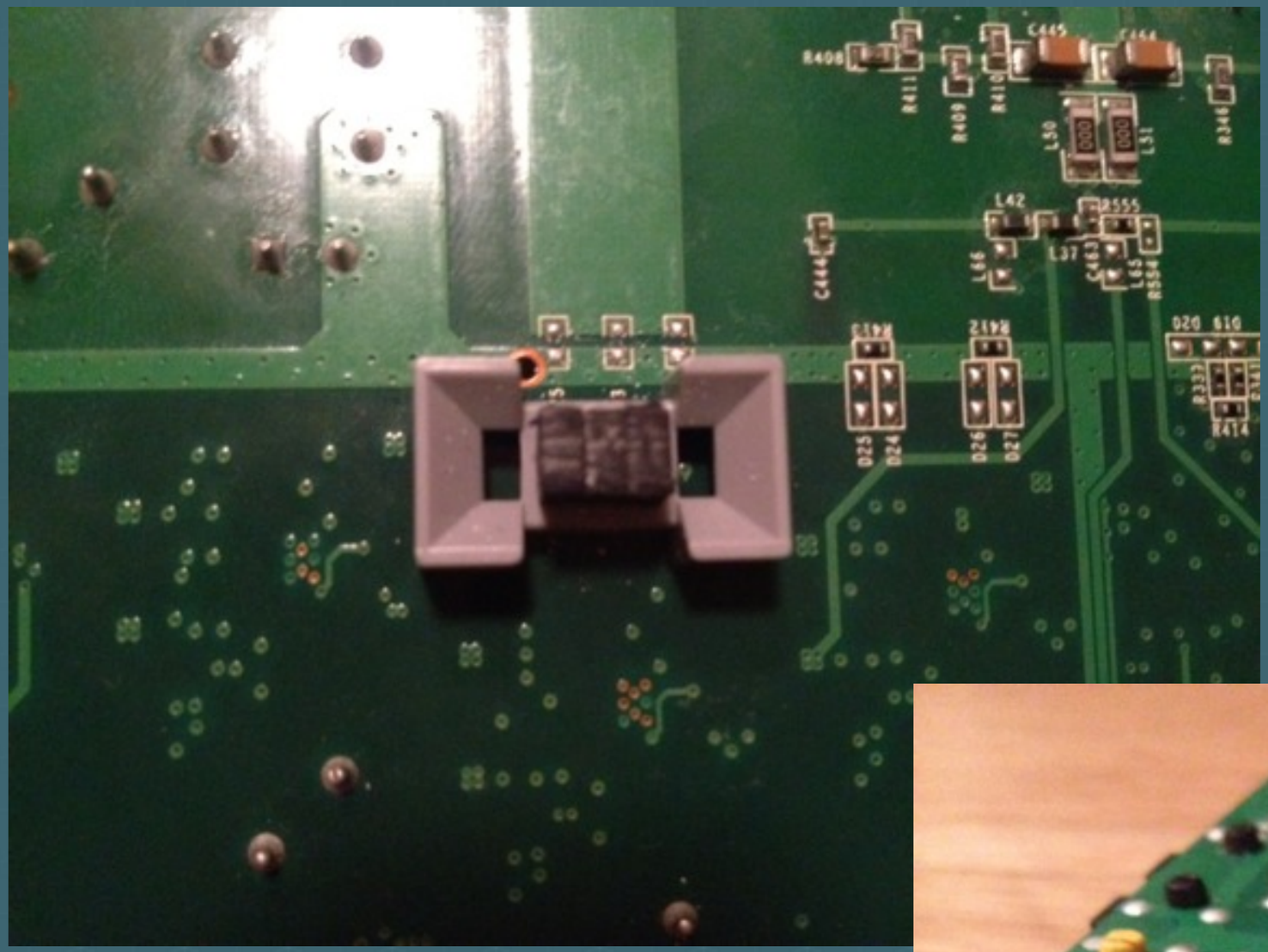
in the 1st round ‘the cage’ knocks wuntee down with a stiff brick to the face, but the battle is not finished...

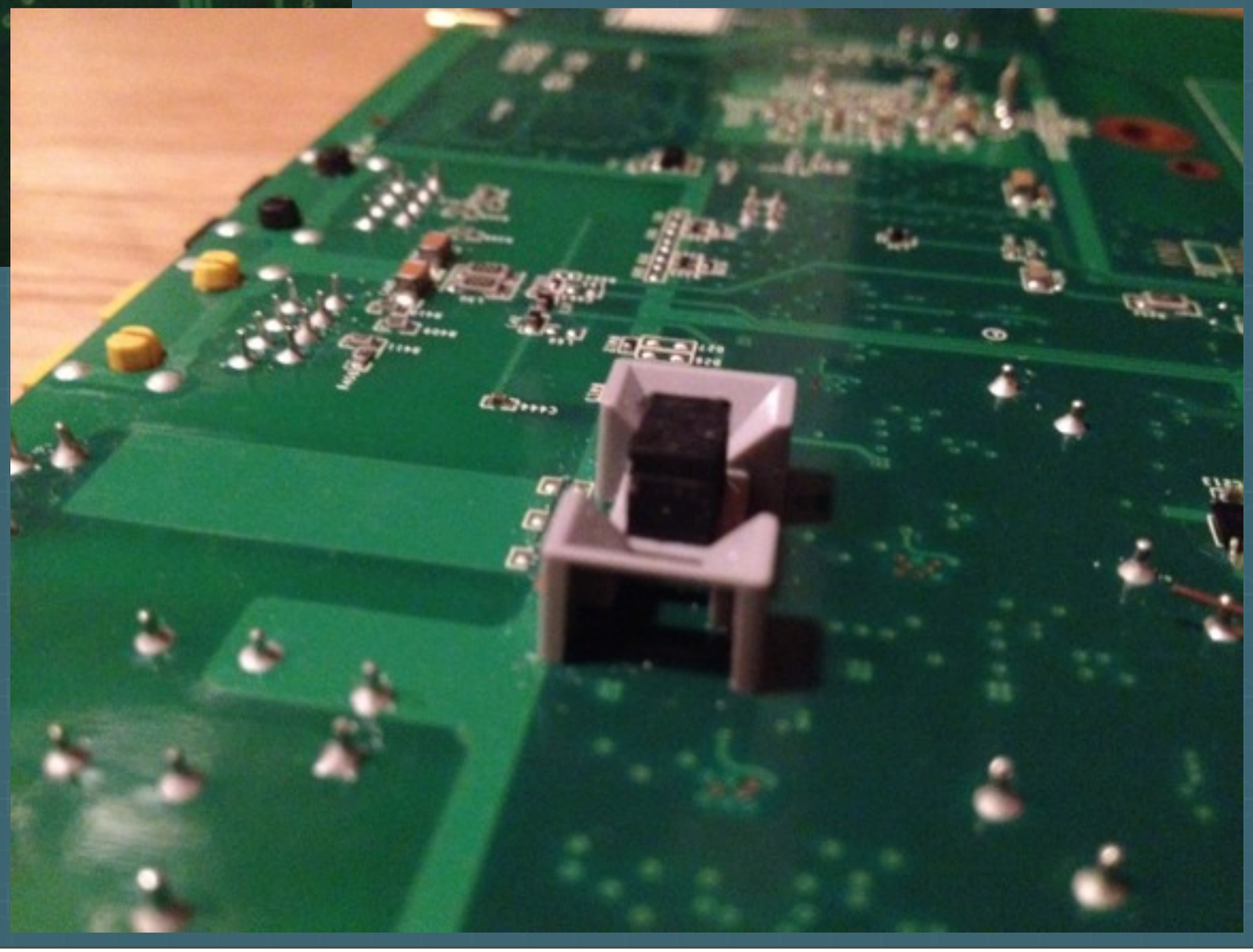
matasano SECURITY

wuntee's corner convincing customer service they are still ok to fight...

matasano SECURITY

wuntee vs the cage round 2


R408
R411
R409
R410
C445
C444
L30
L31
R346
L42
R555
L37
C463
L65
R554
L66
C444
R413
R412
D25
D24
D26
D27
D20
D19
R333
R414

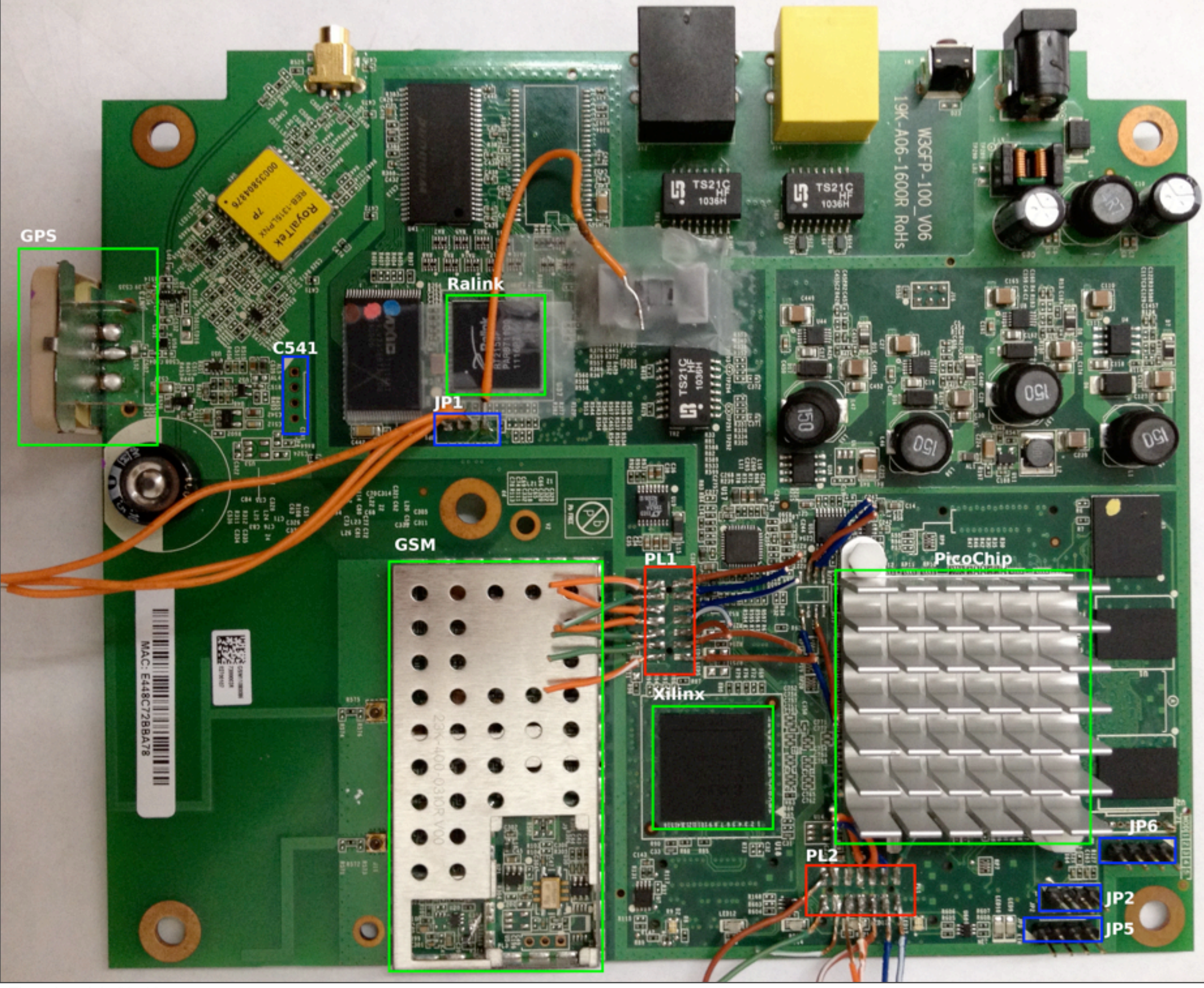


disassembly:
wuntee vs Microcell
round 2

- Went to Home Depot and purchased a thin saw
- Removed bottom orange part
- Sawed through the things attaching the jumpers
- Removed outer cage
- Powered on just fine

wuntee utilized the saw to successfully dismantle ‘the cage’

winner wuntee

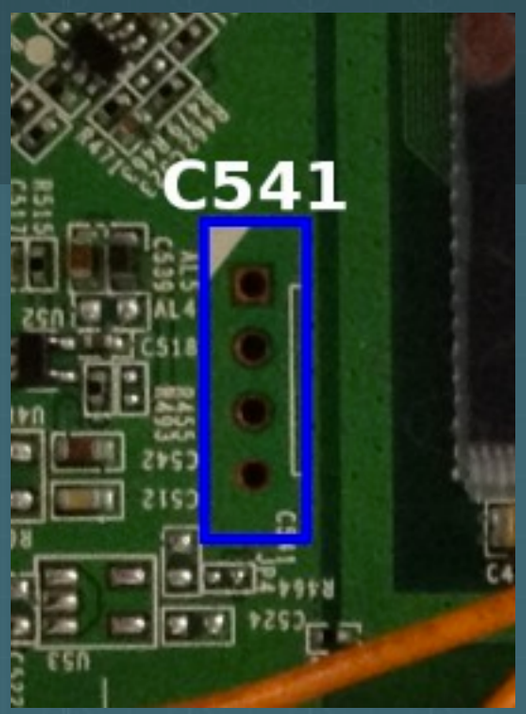

GPS
C541
Ralink
JP1
GSM
PL1
Xilinx
PicoChip
PL2
JP6
JP2
JP5
W3GFP-100_V06
19K-A06-1600R RoHs
TS21C HF 1036H
TS21C HF 1036H
TS21C HF 1036H
MAC: E448C72BBA78

debug pins

- C541
- JP1, JP2, JP5, JP6
- PL1
- PL2

wuntee vs debug pins

round 1 - CS541

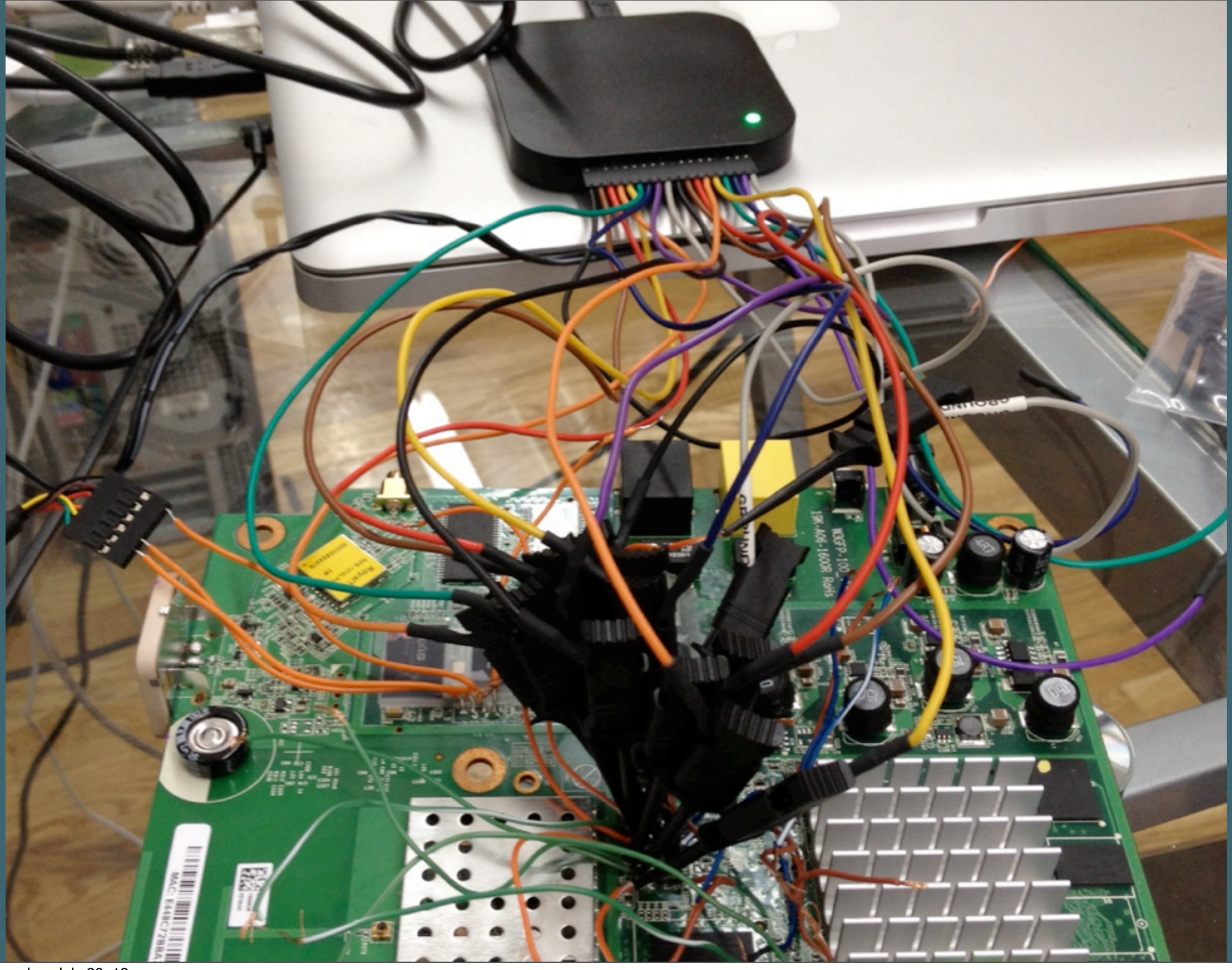


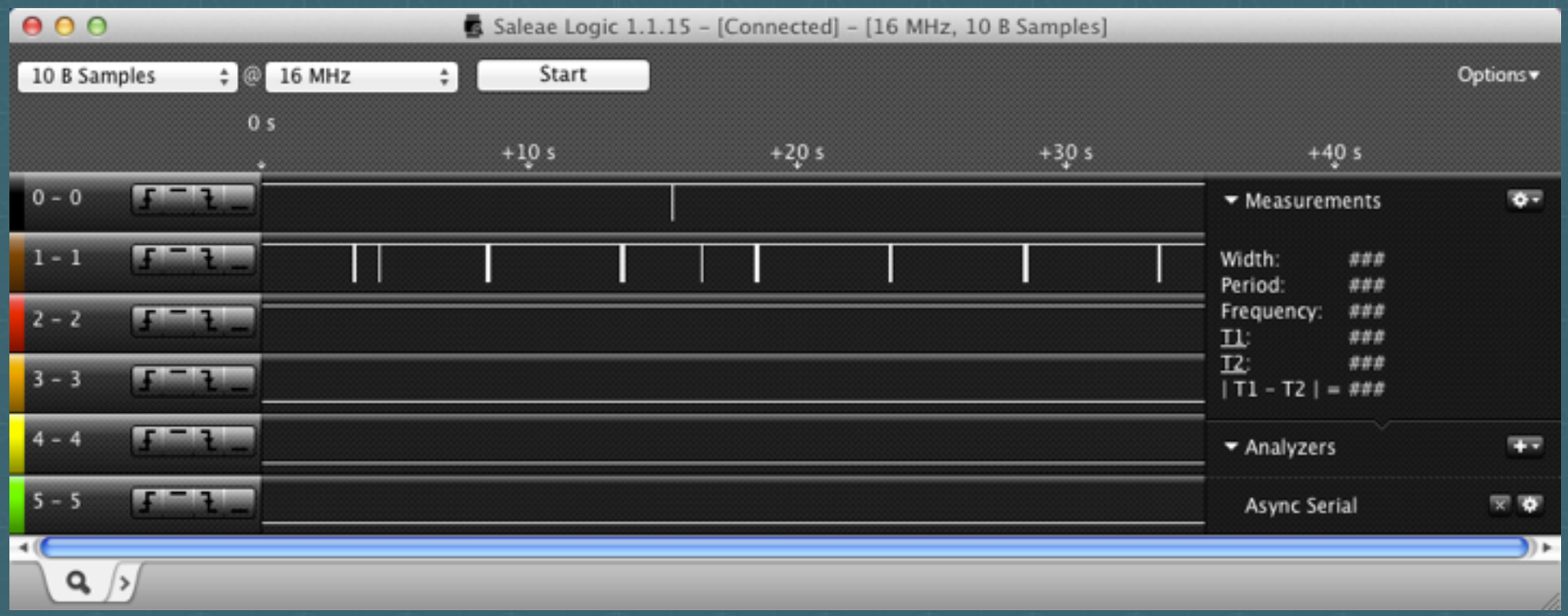
matasano SECURITY

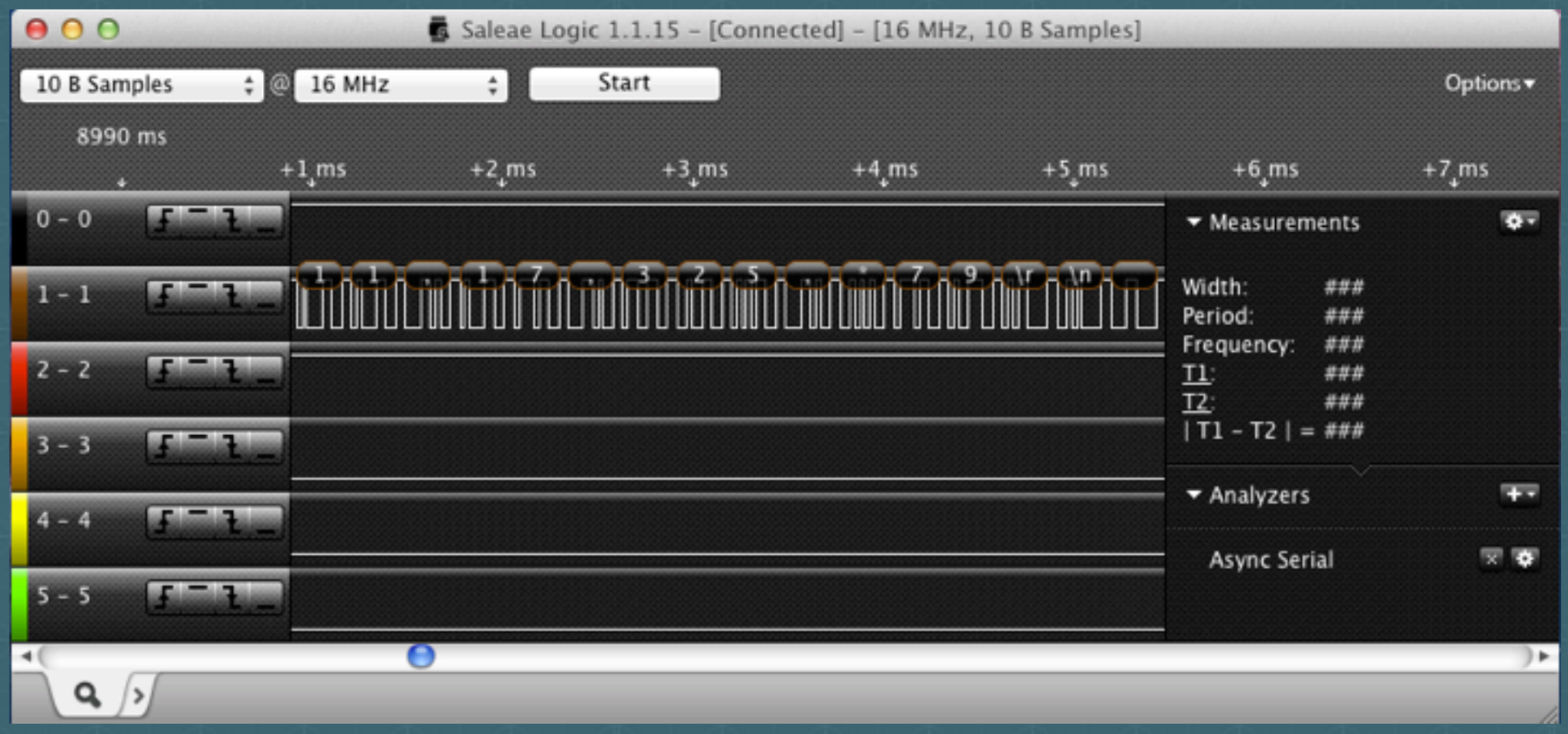
- Saleae Logic Analyzer 16
  - Ability to monitor pins on a board
  - Samples at specific rate/time frame
  - Auto analysis
- Workflow

1. Multimeter to determine ground and that Saleae wont blow up
2. Plug pins to analyzer and sample at high rate
3. Start the Logic software and plug in the device
4. Stop analyzer after you think some data has been transfered
5. Attempt to “Analyze”

matasano SECURITY







matasano SECURITY

DATA!

Export the “analyzed” data to CSV, import to Excel, copy/paste into vi and manipulate

```
'255''255''255''255''255''255''255''255''255''255''255''255''255''255''255
''255''255''255''255''160''162''0'0'221'$GPGGA232354.755000M0.0M0000*50
    '239''176''179''160''162''0'?'221'$GPGSV3111206428306510421648300324301 8*71
'12''209''176''179''160''162''0'?'221'$GPGSV3211194007103363522336179111 7325*79
'12''227''176''179''160''162''0'4'221'$GPGSV3311251023731081380702261*49

'192''176''179''160''162''0'*'221'$GPRMC232354.755V150612N*4A

'21''176''179''255''255''255''255''255''255''255''255''255''255''255''255''255'
'255''255''255''255''255''255''255''160''162''0''2''2''16''0''18''176''179''255
''255''255''255''255''255''255''255''255''255''255''255''255''255''255''255''25
5''255''255''255''160''162''0'0'221'$GPGGA232359.736000M0.0M0000*58
    '251''176''179''160''162''0'?'221'$GPGSV3111206428306510421648300324301 8*71
'12''209''176''179''160''162''0'?'221'$GPGSV3211194007103363522336179111 7325*79
'12''227''176''179''160''162''0'4'221'$GPGSV3311251023731081380702261*49

'192''176''179''160''162''0'*'221'$GPRMC232359.736V150612N*42
```

matasano SECURITY

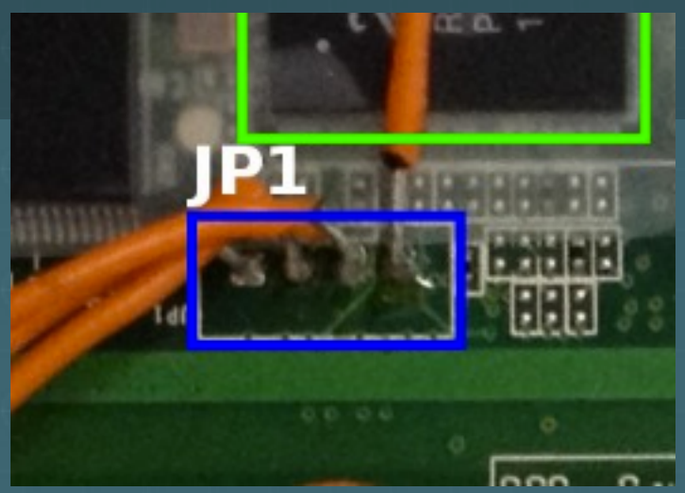
“$GPGSV32111”

- Google?
- Just GPS related data

- Nothing of interest to me
- Could remove GPS chip and send the correct data to spoof location?

wuntee vs debug pins round 1 draw

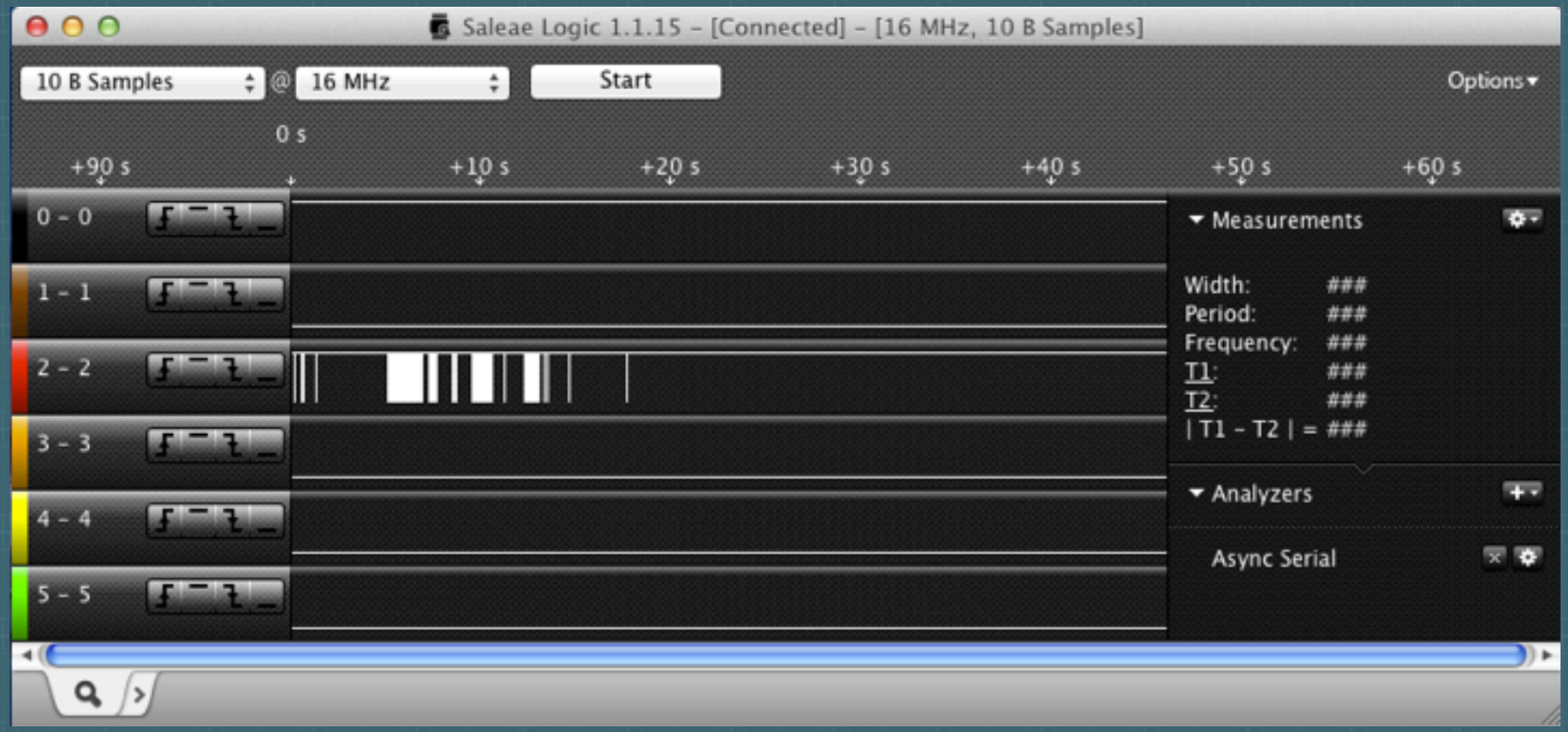
wuntee vs debug pins round 2 - JP1



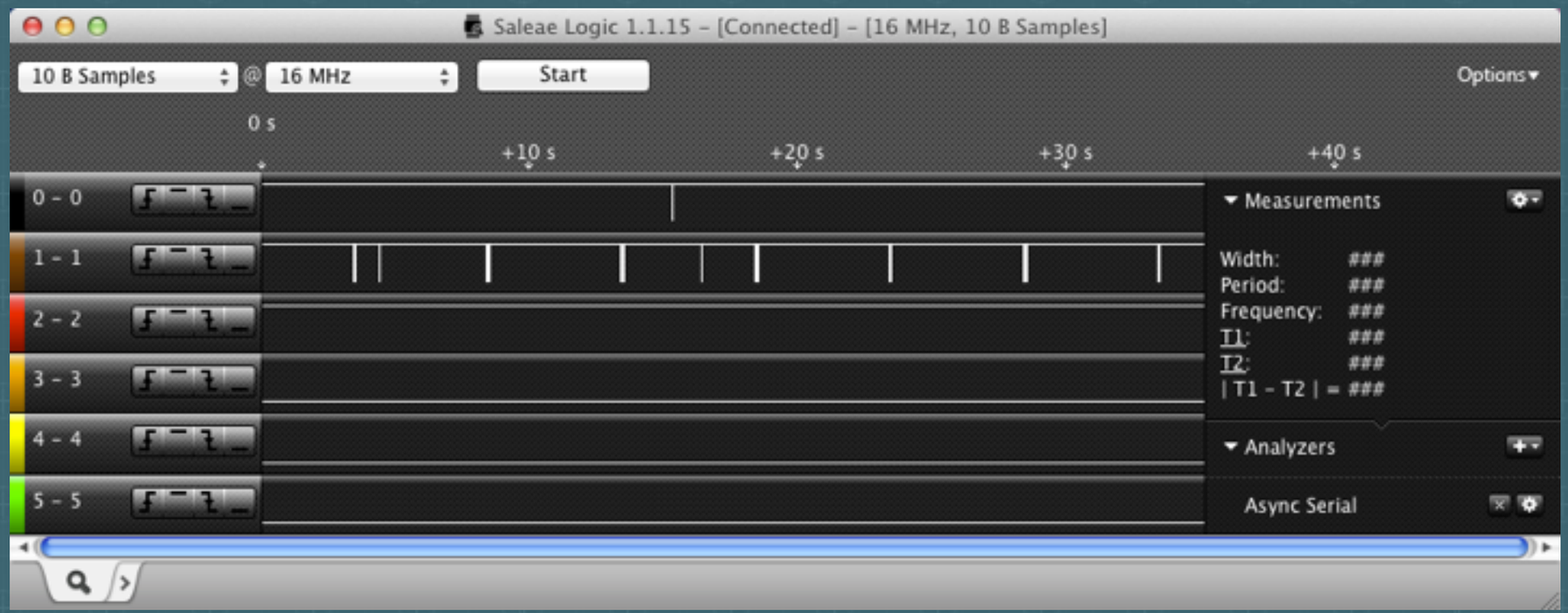
round 2

- Same workflow as before, hope for the best...

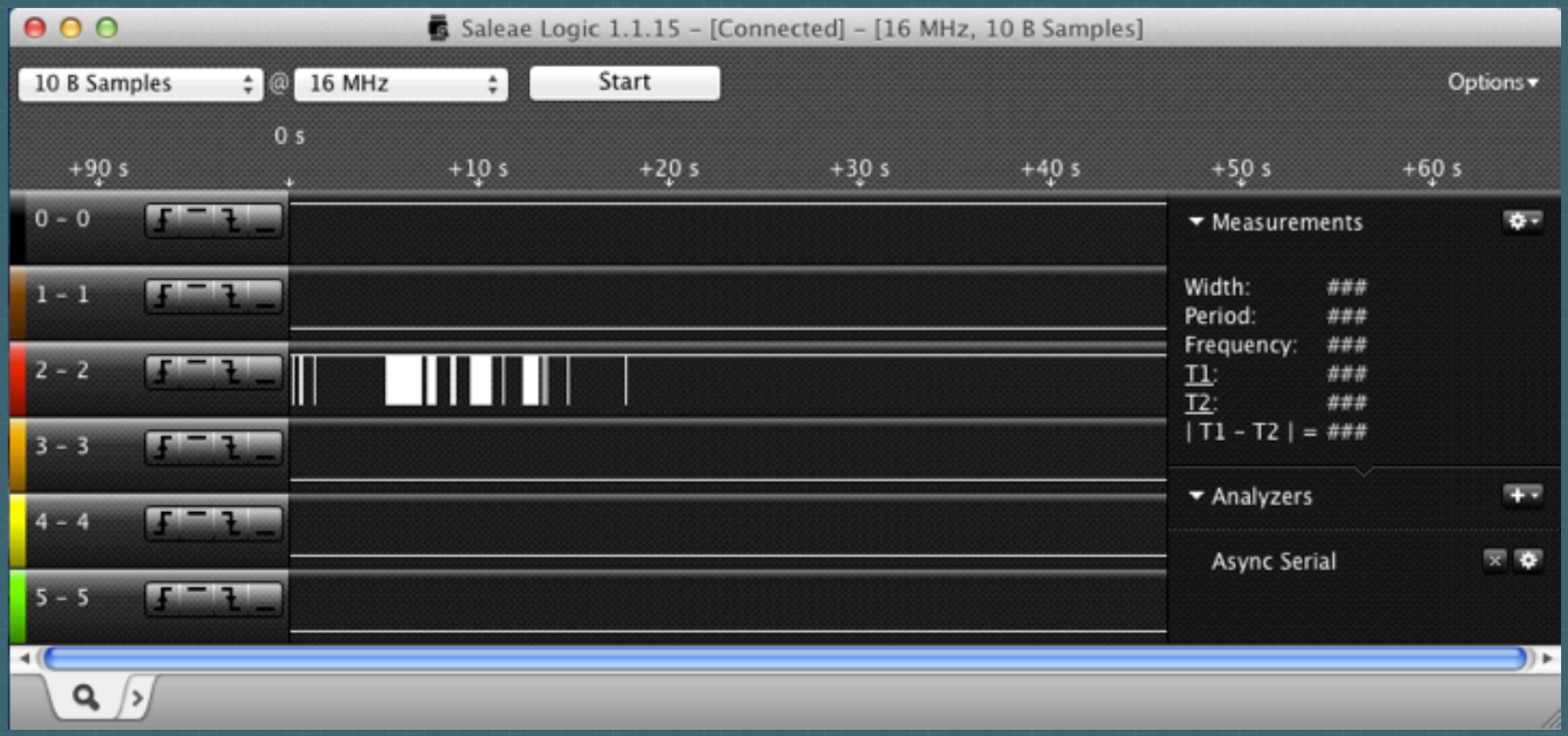
JP1



CS541



matasano SECURITY




```
Erased' '1' 'sectors
Writing' 'to' 'Flash...' 'done

' 'b_end' '=BF3FFFFF
Protecting' 'sectors' '9..9' 'in' 'bank' '1
Protected' '1' 'sectors
' '' '' '
3:' 'System' 'Boot' 'system' 'code' 'via' 'Flash.' 'boot_loc:0' '0xBF040000
##' 'Booting' 'image' 'at' 'bf040000' '...
' '' '' 'Verifying' 'Checksum' '...' 'OK
' '' '' 'Uncompressing' 'Kernel' 'Image' '...' 'OK
No' 'initrd
##' 'Transferring' 'control' 'to' 'Linux' '(at' 'address' '802a0000)' '...
##' 'Giving' 'linux' 'memsize' 'in' 'MB' '16

Starting' 'kernel' '...

\r\nLINUX' 'started...\r
\n' 'THIS' 'IS' 'ASIC\r\nLinux' 'version' '2.6.21' '(perry@perry-pc)' '(gcc' 'version' '3.3.6)' '#47' '
Thu' 'Mar' '4' '16:17:18' 'CST' '2010\r
```

matasano SECURITY

wuntee vs debug pins round 2 - JP1 point wuntee

matasano SECURITY

wuntee vs debug pins

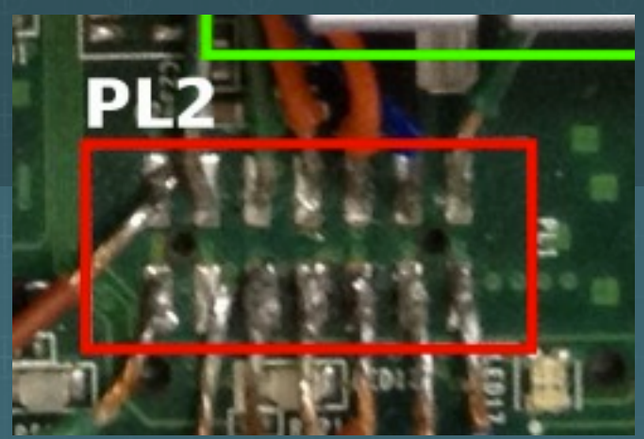
round 3: JP2, JP5, JP6

wuntee vs debug pins round 3: JP2, JP5, JP6 draw.... no show

matasano SECURITY

wuntee vs debug pins

round 4 - PL2

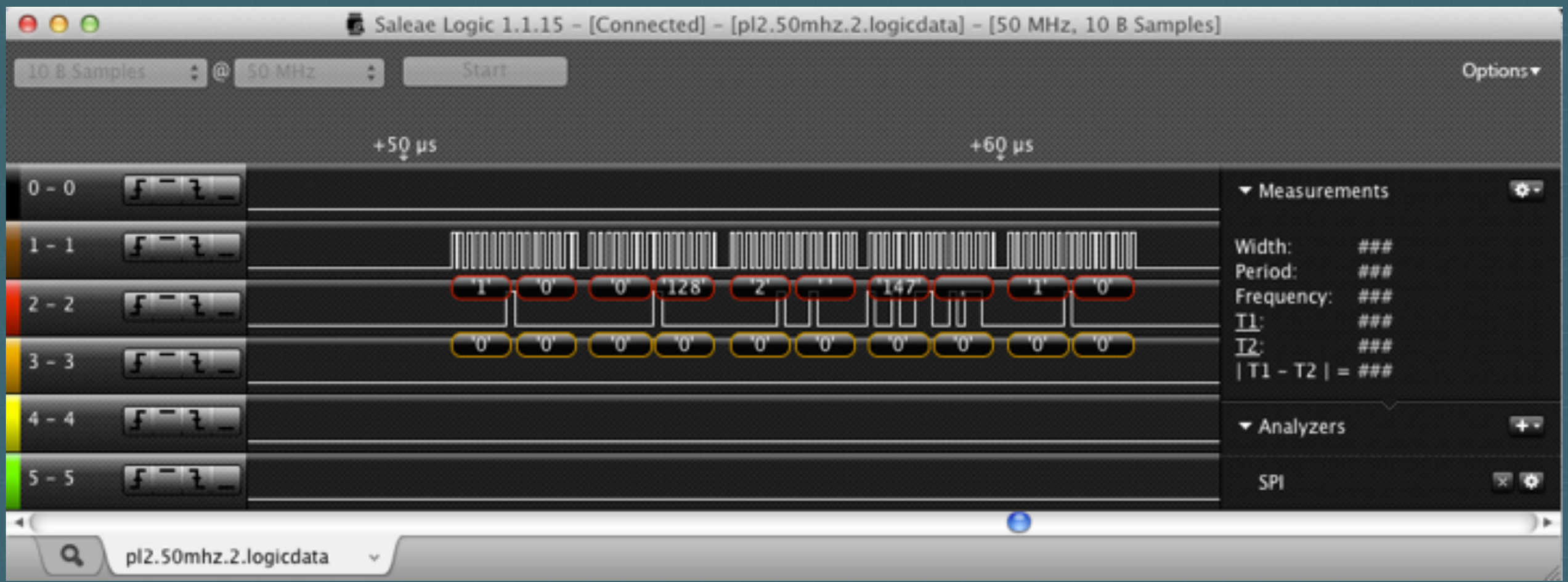


Something different... 3 pins of data?

matasano SECURITY

SPI

- Up to 100MHz - must increase sample rate
- Master/slave with multiple slaves
- Four lines
  - MOSI – Output
  - MISO – Input
  - Enable/Slave Select – Determine which slave the master is talking to
  - Clock – Not like your typical metronome clock, but will be explained in the next point
- The clock operates in one of two modes, called CHPA, where the data on one of the lines (MOSI, or MISO) is "read" when the clock is changing from low to high, or high to low. So, if it's set up on low to high, when you see the line on the clock go from bottom to top, that is when the MOSI and MISO lines are read.

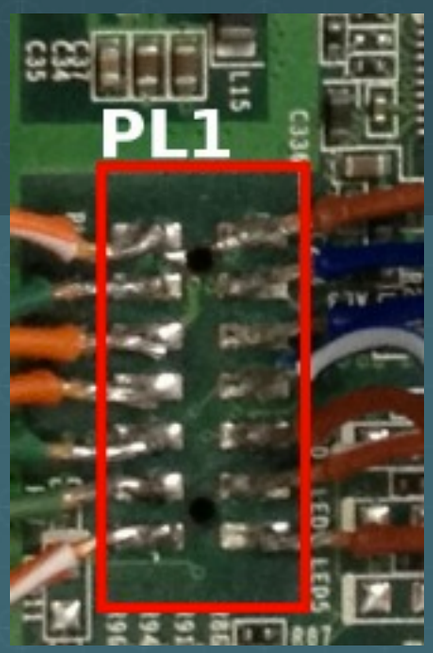


matasano SECURITY

wuntee vs debug pins round 4 - PL2 point debug pins

wuntee vs debug pins

round 5 - PL1



matasano SECURITY

PL1

- No data seen with logic analyzer
- However, these 7x2 pins “scream JTAG”

what is jtag?

- Allowed me to dump and update firmware on SurfBoard modems?
- Standard for hardware developers the ability do debug chips that have already been placed on a board.

jtag

- JTAG pins, on their own, do not send any data. AKA – you will not see anything if you only have a logic analyzer connected
- The cable provides the clock signal to the board (presumably that's why there is no data on the pins on their own)
- There are 5 pins that must be connected in order to communicate with a device (VREF, TMS, TCK, TDO, TDI)
- Multiple chips can be "daisy chained" together. Meaning one JTAG plug/pin-out can communicate with multiple chips on a board
- Each chip that is connected in a JTAG chain is called a TAP

matasano SECURITY

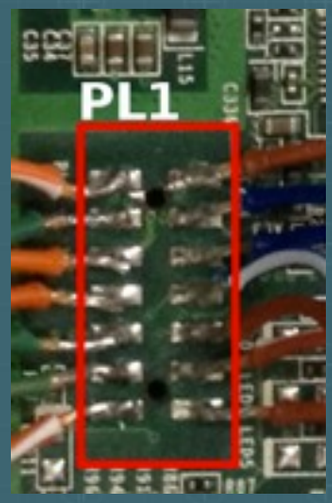
first... hardware/software

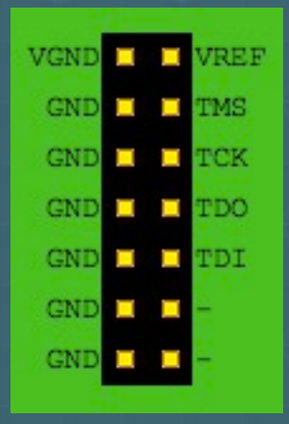
- Olimex ARM-USB-OCD-H
  - Docs have JTAG pinout
- OpenOCD
  - Open source
  - Supports Olimex cable
  - Ability to auto-discover TAPs
    - Can reliably find TAP ID
    - Unreliably find IRLEN

matasano SECURITY

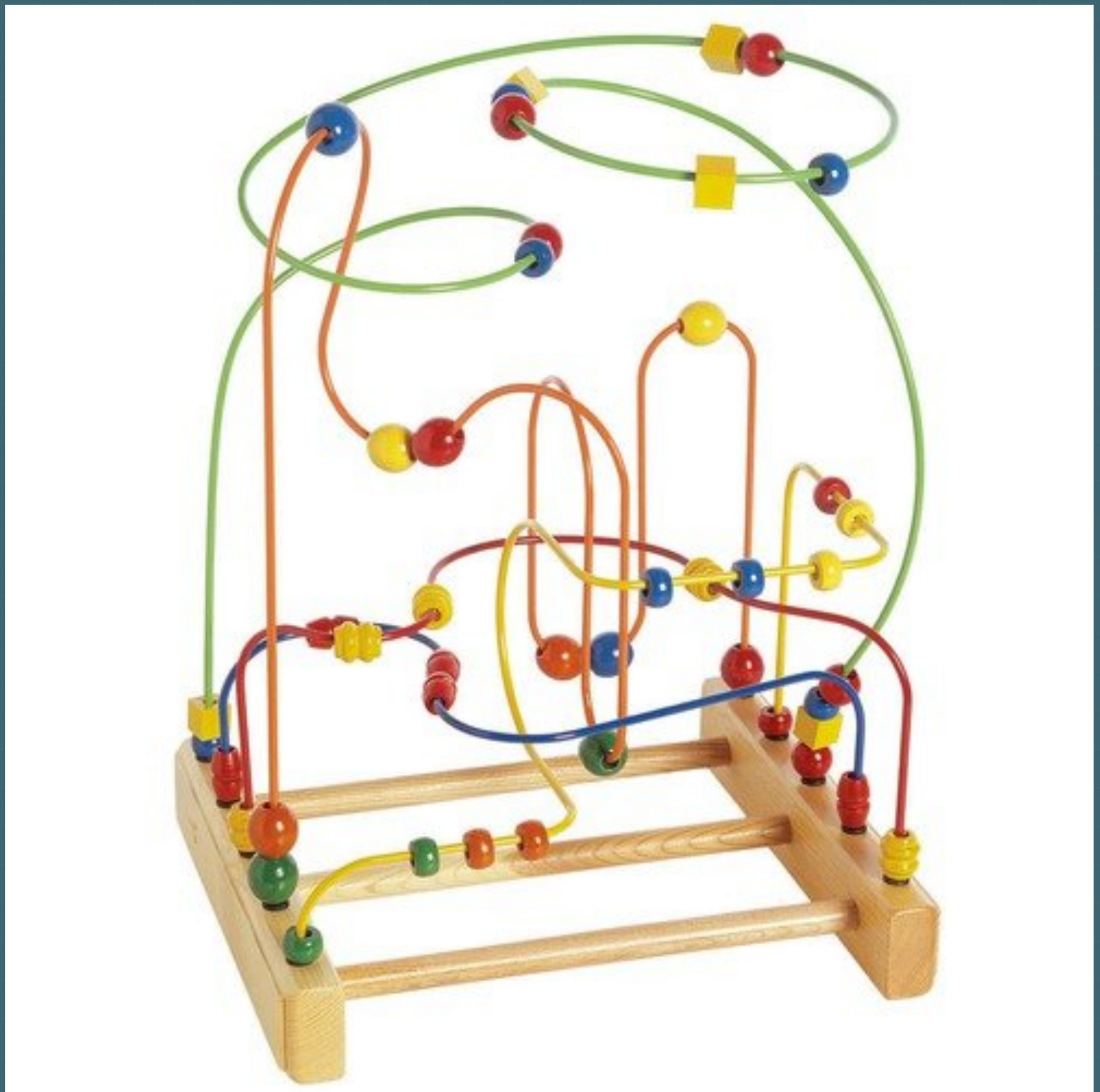
then... pinout discovery workflow

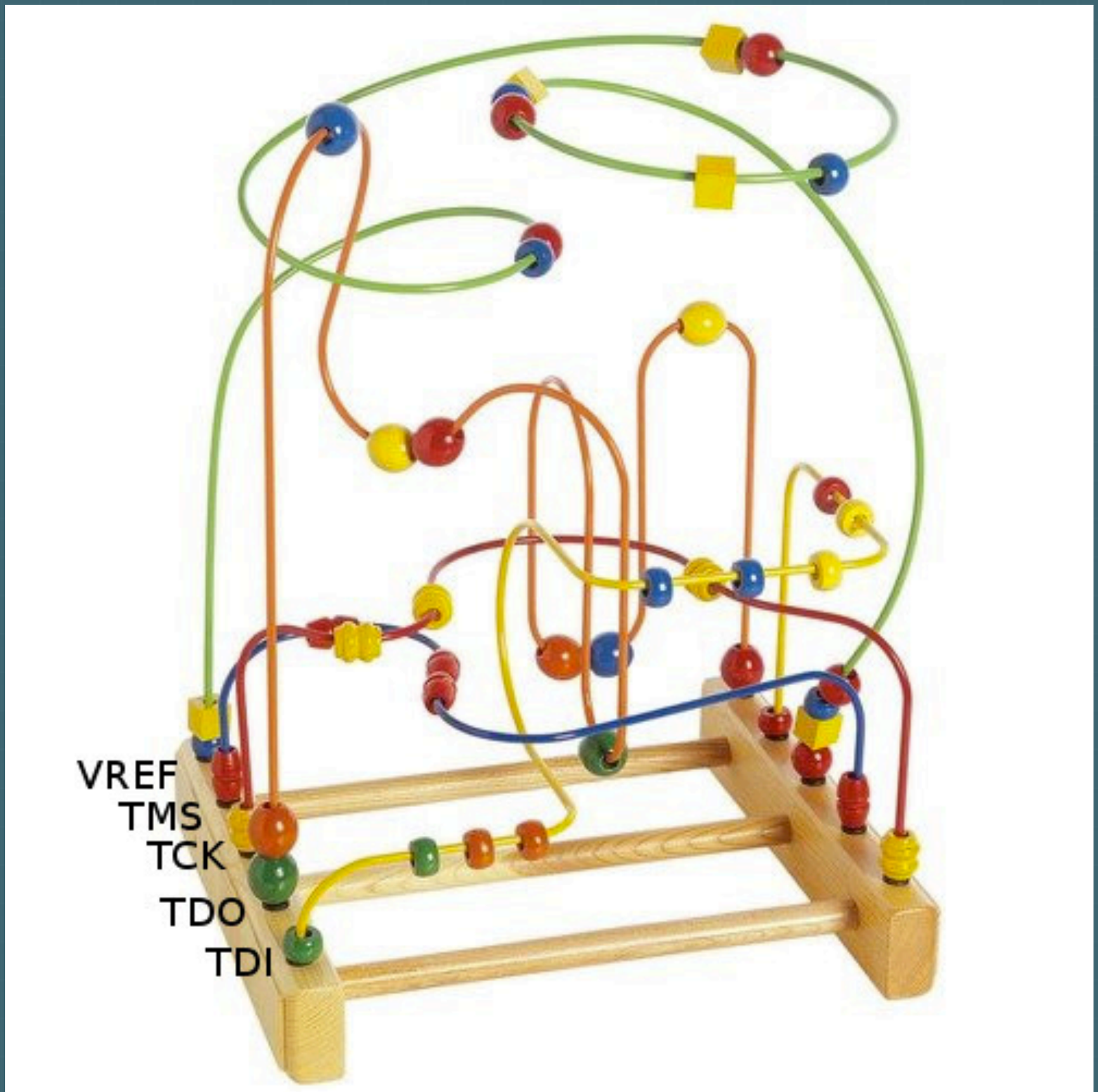
1. If there is data on the pins, then its not JTAG
2. If there is a known configuration for the pins, plug the JTAG up accordingly (as well as the 180 degree flip version as we do not know which is PIN0)
3. Power the device
4. Start OpenOCD software. If it can discover TAPs, then you have a JTAG port





matasano SECURITY




VREF
TMS
TCK
TDO
TDI

```
$ sudo ./openocd -f wuntee.cfg
Open On-Chip Debugger 0.5.0 (2012-07-02-13:56)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
3000 kHz
trst_and_srst separate srst_gates_jtag trst_push_pull
srst_open_drain
RCLK - adaptive
Info : device: 6 "2232H"
Info : deviceID: 364511275
Info : SerialNumber: OLUTHMH9A
Info : Description: Olimex OpenOCD JTAG ARM-USB-OCD-H A
Info : max TCK change to: 30000 kHz
Info : RCLK (adaptive clock speed)
Warn : There are no enabled taps.  AUTO PROBING MIGHT NOT
WORK!!
Warn : AUTO auto0.tap - use "jtag newtap auto0 tap -expected-id
0x02220093 ..."
Warn : AUTO auto0.tap - use "... -irlen 2"
Error: IR capture error at bit 2, saw 0x3FFFFFFFFFFFFFF5 not
0x...3
Warn : Bypassing JTAG setup events due to errors
Warn : gdb services need one or more targets defined
```

next step... configure TAP

- Googling the expected-id reveals this is the Xilinx chip
- OpenOCD TAP configuration needs:
  - expected-id
  - irlen
  - ircapture
  - irmask
- BSDL
  - Configuration file defining how to communicate via JTAG to a specific chip
  - Xilinx provides for each chip version

```
...
attribute INSTRUCTION_LENGTH of XC3S400_BARE : entity is
6;
...
attribute INSTRUCTION_CAPTURE of XC3S400_BARE : entity
is
-- Bit 5 is 1 when DONE is released (part of startup
sequence)
-- Bit 4 is 1 if house-cleaning is complete
-- Bit 3 is ISC_Enabled
-- Bit 2 is ISC_Done
        "XXXX01";
...
```

```
$ sudo openocd -f probe.cfg
Open On-Chip Debugger 0.6.0-dev-00603-g43863b6
(2012-07-10-12:01)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
RCLK - adaptive
3000 kHz
trst_and_srst separate srst_gates_jtag trst_push_pull
srst_open_drain
Info : clock speed 3000 kHz
Info : JTAG tap: unk1.tap tap/device found: 0x02220093 (mfg:
0x049, part: 0x2220, ver: 0x0)
Warn : gdb services need one or more targets defined
```

```
> jtag init
Info : JTAG tap: unk1.tap tap/device found:
0x02220093 (mfg: 0x049, part: 0x2220, ver: 0x0)
```

ok, now what?

- We can communicate with the Xilinx chip via JTAG, however that doesn't really give me much of anything...
- No flash
- No OS
- We can now, maybe, program the FPGA

wuntee vs debug pins round 5 - PL1 point debug pins

wuntee vs software
round 1 - uboot

wuntee vs software
round 1: uboot

- Remember JP1? (Linux boot prompt)
- 3v3 USB FTDI cable to pins for two way communication
- Goes through to a “login:” prompt
- Initially I was starting the terminal session after the device started booting so I was not seeing the UBoot procedure
- After a while I noticed a pause in the UBoot procedure...

```
==========================================
Ralink UBoot Version: 3.7.1
------------------------------------------
ASIC 2150_MP2 (MAC to GigaMAC Mode)
DRAM COMPONENT: 128Mbits
DRAM BUS: 16BIT
Total memory: 16 MBytes
Date:Jan  7 2009  Time:12:26:56
==========================================
icache: sets:256, ways:4, linesz:32 ,total:32768
dcache: sets:128, ways:4, linesz:32 ,total:16384

 ##### The CPU freq = 384 MHZ ####

SDRAM bus set to 16 bit
 SDRAM size =16 Mbytes

Please choose the operation:
   1: Load system code to SDRAM via TFTP.
   2: Load system code then write to Flash via TFTP.
   3: Boot system code via Flash (default).
   4: Entr boot command line interface.
   9: Load Boot Loader code then write to Flash via TFTP.

<PAUSE>
```

uboot

- Press 4, and you're in the uboot prompt
- HELP - List available commands
  - Read UBoot manual...
- Only option was:
  - MD - Memory Display
  - MD + Screen(for logging) + Ruby = Flash Dump

Thank you ExploitWorkshop.org

Ralink

The full 4MiB File:MX.raw.bz2 dump.

```
0xA0000000 - 0xC0000000         kseg1 unmapped, uncached
    0xA0A81000                  phy_tx_ring, size: 16 bytes
    0xA0A82000                  phy_rx_ring, size: 16 bytes
    0xB0000000 - 0xB0200000     Ralink Registers
        0xB0000500              serial8250: ttyS0 (irq = 37) is a 16550A
        0xB0000C00              serial8250: ttyS1 (irq = 12) is a 16550A
    0xBF000000 - 0xBF400000     MX flash device: 4MiB Flash (See 0xBFC00000)
    0xBF400000 - 0xBF800000     MX flash device: 4MiB Flash (See 0xBFC00000)
    0xBF800000 - 0xBFC00000     MX flash device: 4MiB Flash (See 0xBFC00000)
    0xBFC00000 - 0xC0000000     4MiB Flash (MX.raw.bz2)
        0xBFC00000 - 0xBFC20000 "Bootloader"
            0xBFC00000 - 0xBFC1EF67     U-Boot
                0xBFC1E2E4 - 0xBFC1E5B1 U-Boot default config
        0xBFC20000 - 0xBFC30000 "Config"
            0xBFC20000 - 0xBFC2031B     4d 3b ac 50 62 6f (U-Boot config)
            0xBFC24000 - 0xBFC29923     03 92 da de 89 01 (PICO config?)
        0xBFC30000 - 0xBFC40000 "Config2"
            0xBFC30000 - 0xBFC34E63     03 92 da de 88 01 (PICO config?)
            0xBFC3BF2C - 0xBFC3BFFF     68 b4 00 00 10 00 (short switch info?)
        0xBFC40000 - 0xBFE20000 "Kernel"
            0xBFC40000 - 0xBFDDCA76     27 05 19 56 c0 36 (U-Boot image: "Linux Kernel Image")
            0xBFDE0000 - 0xBFDF8A8b     ea 26 a2 8d fb d4 (Kernel.extra1 - unidentified)
            0xBFE00000 - 0xBFE16D6A     8e 98 bc 1b ae 6a (Kernel.extra2 - unidentified)
        0xBFE20000 - 0xC0000000 "Kernel2"
            0xBFE20000 - 0xBFFBCA76     27 05 19 56 c0 36 (U-Boot image: "Linux Kernel Image")
            0xBFFC0000 - 0xBFFFFED9     38 d6 cd 35 b2 9a (Kernel2.extra1 - possibly lzma compressed squashfs)
```

matasano SECURITY

```
RT2150 # md bfc00000 1000000
bfc00000: 100000ff 00000000 100000fd 00000000    ................
bfc00010: 10000219 00000000 10000217 00000000    ................
bfc00020: 10000215 00000000 10000213 00000000    ................
…
bffffff0: ffffffff ffffffff ffffffff ffffffff    ................
```

```
$ ruby memToBin.rb microcell.hex microcell.bin bfc00000 bffffff0

$ file microcell.bin
microcell.bin: data
```

matasano SECURITY

wuntee vs software
round 2 - firmware

flash/firmware analysis tools

- strings
- file
- binwalk - steps each byte of a file and basically performs ‘file’ as if the file started there (configurable magic file)

microcell.bin (full 4mb flash)

```
$ file microcell.bin
microcell.bin: data
```

```
$ strings -n 10 microcell.bin
[nada]
```

```
DECIMAL       HEX             DESCRIPTION
--------------------------------------------------------------------------------------------------
38193         0x9531          LZMA compressed data, properties: 0x80, dictionary size: 807469056 bytes, uncompressed size: 941686944 bytes
53113         0xCF79          LZMA compressed data, properties: 0x90, dictionary size: 46923776 bytes, uncompressed size: 36738 bytes
262144        0x40000         uImage header, header size: 64 bytes, header CRC: 0xC0361020, created: Thu Mar  4 03:17:29 2010, image size: 1690167 bytes, Data Address:
 0x80000000, Entry Point: 0x802A0000, data CRC: 0x70DC4C09, OS: Linux, CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image name: Linux Kernel Image
262208        0x40040         LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 3681740 bytes
2228224       0x220000        uImage header, header size: 64 bytes, header CRC: 0xC0361020, created: Thu Mar  4 03:17:29 2010, image size: 1690167 bytes, Data Address:
 0x80000000, Entry Point: 0x802A0000, data CRC: 0x70DC4C09, OS: Linux, CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image name: Linux Kernel Image
2228288       0x220040        LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes, uncompressed size: 3681740 bytes
```

dd

- Allows you to dump contents of a file from point x to end
- LZMA will decompress regardless if the ending of the file is legitimate
- Of course, the only one that worked was the last one

lzma4

```
$ file lzma4
lzma4: data

$ strings -n 10 lzma4 | head
__remove_pages
TERM=linux
<4>Parameter %s is obsolete, ignored
<3>Unknown boot option `%s': ignoring
Too many boot env vars at `%s'
Too many boot init vars at `%s'
<4>Malformed early option '%s'
early options
<5>Kernel command line: %s
Booting kernel
```

binwalk

```
DECIMAL       HEX             DESCRIPTION
--------------------------------------------------------------------------------
43879         0xAB67          LZMA compressed data, properties: 0x99, dictionary size: 604110848 bytes, uncompressed size: 134228624 bytes
43991         0xABD7          LZMA compressed data, properties: 0xA1, dictionary size: 604110848 bytes, uncompressed size: 272695199 bytes
44055         0xAC17          LZMA compressed data, properties: 0xB9, dictionary size: 604110848 bytes, uncompressed size: 134228727 bytes
48415         0xBD1F          LZMA compressed data, properties: 0x8D, dictionary size: 604176384 bytes, uncompressed size: 285409297 bytes
211609        0x33A99         LZMA compressed data, properties: 0x98, dictionary size: 2883584 bytes, uncompressed size: 270602336 bytes
665349        0xA2705         LZMA compressed data, properties: 0xB0, dictionary size: 4456448 bytes, uncompressed size: 539037760 bytes
747221        0xB66D5         LZMA compressed data, properties: 0x90, dictionary size: 3538944 bytes, uncompressed size: 5376 bytes
1230829       0x12C7ED        LZMA compressed data, properties: 0x90, dictionary size: 1507328 bytes, uncompressed size: 270602368 bytes
1321181       0x1428DD        LZMA compressed data, properties: 0x98, dictionary size: 4718592 bytes, uncompressed size: 404820514 bytes
1345045       0x148615        LZMA compressed data, properties: 0x98, dictionary size: 4718592 bytes, uncompressed size: 404820514 bytes
1361573       0x14C6A5        LZMA compressed data, properties: 0x98, dictionary size: 262144 bytes, uncompressed size: 4800 bytes
2280645       0x22CCC5        LZMA compressed data, properties: 0x80, dictionary size: 2686976 bytes, uncompressed size: 404821056 bytes
2352340       0x23E4D4        Squashfs filesystem, big endian, lzma compression,  version 10281.2560, size: 7304680684267074304 bytes, 1835097973 inodes, blocksize: 16
60944384 bytes, created: Sat May 10 19:23:31 2031
2525762       0x268A42        Squashfs filesystem, big endian, lzma compression,  version 4598.1432, size: 969631157860134376 bytes, -737483663 inodes, blocksize: -175
4829336 bytes, created: Fri Aug 22 18:03:59 2031
2625208       0x280EB8        Squashfs filesystem, big endian, lzma compression,  version 28001.24422, size: 7956861085501714028 bytes, 1835097973 inodes, blocksize: 7
695986 bytes, created: Fri Jan 23 08:15:17 2026
2625220       0x280EC4        Squashfs filesystem, big endian, lzma compression,  version 29556.25970, size: 7956861085501124449 bytes, 1835097961 inodes, blocksize: 1
852601088 bytes, created: Mon Mar 30 20:46:26 1970
2625232       0x280ED0        Squashfs filesystem, big endian, lzma compression,  version 30062.29285, size: 8286623314368819041 bytes, 1835097958 inodes, blocksize: 1
852601088 bytes, created: Thu Sep 14 23:24:48 2028
2650332       0x2870DC        LZMA compressed data, properties: 0x80, dictionary size: 536870912 bytes, uncompressed size: 8393935 bytes
2690621       0x290E3D        LZMA compressed data, properties: 0x00, dictionary size: 8388608 bytes, uncompressed size: 2129937 bytes
2754996       0x2A09B4        Linux rev 0.0 ext2 filesystem data (mounted or unclean), UUID=1800bd27-e8ff-bd27-1000-bfaff9cc0a0c (huge files)
2890204       0x2C19DC        LZMA compressed data, properties: 0x98, dictionary size: 805306368 bytes, uncompressed size: 1 bytes
2890224       0x2C19F0        LZMA compressed data, properties: 0x98, dictionary size: 805306368 bytes, uncompressed size: 1 bytes
2899968       0x2C4000        LZMA compressed data, properties: 0x5D, dictionary size: 1048576 bytes, uncompressed size: 2862592 bytes
```

matasano SECURITY

lzma4.18

```
$ file lzma4.18
lzma4.18: ASCII cpio archive (SVR4 with no CRC)

$ strings -n 10 lzma4.18 | head
070701000002D10000A1FF000003E8000003E80000000014B8F6C8A0000000C0000000
bin/busybox
070701000002D2000041ED000003E8000003E80000000024B8F6C8A0000000000000000
070701000002D3000041ED000003E8000003E80000000024B8F6C8A0000000000000000
070701000002D4000041ED000003E8000003E80000000024B8F6C8A0000000000000000
070701000002D5000041ED000003E8000003E80000000024B8F6C8A0000000000000000
070701000002D60000A1FF000003E8000003E80000000014B8F6C8A0000001200000000
setlogcons
../../bin/busybox
070701000002D7000081ED000003E8000003E80000000014B8F6C8500001DF80000000
ipc_client
/lib/ld-uClibc.so.0
```

```
DECIMAL       HEX             DESCRIPTION
--------------------------------------------------------------------------------
880           0x370           ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV)
6951          0x1B27          LZMA compressed data, properties: 0x80, dictionary size: 4456448 bytes, uncompressed size: 4194304 bytes
8688          0x21F0          ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV)
9520          0x2530          LZMA compressed data, properties: 0xCA, dictionary size: 855638016 bytes, uncompressed size: 184549376 bytes
9620          0x2594          LZMA compressed data, properties: 0xB9, dictionary size: 2130706432 bytes, uncompressed size: 335544320 bytes
9680          0x25D0          LZMA compressed data, properties: 0xC8, dictionary size: 1275068416 bytes, uncompressed size: 973078528 bytes
9716          0x25F4          LZMA compressed data, properties: 0xC1, dictionary size: 201326592 bytes, uncompressed size: 1023410176 bytes
180800        0x2C240         ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV)
198743        0x30857         LZMA compressed data, properties: 0x80, dictionary size: 4456448 bytes, uncompressed size: 4194304 bytes
201236        0x31214         LZMA compressed data, properties: 0x84, dictionary size: 16777216 bytes, uncompressed size: 50331648 bytes
201864        0x31488         ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV)
212319        0x33D5F         LZMA compressed data, properties: 0x80, dictionary size: 4456448 bytes, uncompressed size: 4194304 bytes
214756        0x346E4         ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV)
234811        0x3953B         LZMA compressed data, properties: 0x80, dictionary size: 4456448 bytes, uncompressed size: 4194304 bytes
239340        0x3A6EC         ELF 32-bit LSB executable, MIPS, MIPS-II version 1 (SYSV)
244008        0x3B928         LZMA compressed data, properties: 0xA4, dictionary size: 4587520 bytes, uncompressed size: 4506360 bytes
262691        0x40223         LZMA compressed data, properties: 0x80, dictionary size: 4456448 bytes, uncompressed size: 4194304 bytes
265924        0x40EC4         ELF 32-bit LSB shared object, MIPS, MIPS-I version 1 (SYSV)
279964        0x4459C         ELF 32-bit LSB shared object, MIPS, MIPS-I version 1 (SYSV)
298104        0x48C78         ELF 32-bit LSB shared object, MIPS, MIPS-II version 1 (SYSV)
361516        0x5842C         LZMA compressed data, properties: 0xC0, dictionary size: 486539264 bytes, uncompressed size: 520093696 bytes
365320        0x59308         ELF 32-bit LSB shared object, MIPS, MIPS-I version 1 (SYSV)
371444        0x5AAF4         ELF 32-bit LSB shared object, MIPS, MIPS-I version 1 (SYSV)
381256        0x5D148         ELF 32-bit LSB shared object, MIPS, MIPS-I version 1 (SYSV)
391120        0x5F7D0         ELF 32-bit LSB shared object, MIPS, MIPS-I version 1 (SYSV)
391252        0x5F854         LZMA compressed data, properties: 0xCC, dictionary size: 872415232 bytes, uncompressed size: 100663297 bytes
417868        0x6604C         ELF 32-bit LSB shared object, MIPS, MIPS-I version 1 (SYSV)
418992        0x664B0         LZMA compressed data, properties: 0x5D, dictionary size: 1325400064 bytes, uncompressed size: 520093696 bytes
471440        0x73190         LZMA compressed data, properties: 0xC0, dictionary size: 1342177280 bytes, uncompressed size: 167772167 bytes
471504        0x731D0         LZMA compressed data, properties: 0xE0, dictionary size: 1342177280 bytes, uncompressed size: 100663303 bytes
471536        0x731F0         LZMA compressed data, properties: 0x90, dictionary size: 1392508928 bytes, uncompressed size: 989855751 bytes
471568        0x73210         LZMA compressed data, properties: 0xD0, dictionary size: 1358954496 bytes, uncompressed size: 285212679 bytes
471696        0x73290         LZMA compressed data, properties: 0xC8, dictionary size: 1358954496 bytes, uncompressed size: 218103815 bytes
471824        0x73310         LZMA compressed data, properties: 0xD8, dictionary size: 1375731712 bytes, uncompressed size: 385875975 bytes
```

googling strings -> cpio archive. however.....

googling strings -> cpio archive. however.....

```
$ file lzma4.18
lzma4.18: ASCII cpio archive (SVR4 with no CRC)
```

cpio hell

```
$ cpio -it -F lzma4.18
/init
/var
/proc
/usr
/usr/sbin
/usr/sbin/setlogcons
/usr/sbin/ipc_client
/usr/sbin/config_server
/usr/sbin/cs_client
/usr/sbin/telnetd
/usr/sbin/udhcpd
/usr/sbin/rmm_client
/usr/sbin/chpasswd
/usr/sbin/ipc_server
/usr/bin
…
```

matasano SECURITY

CPIO limitation...

-r (All modes.) Rename files interactively.

```
$ cpio -it -F lzma4.18 | wc -l
5591 blocks
     208
```

- Yes, there are other ways to do this... I was just too excited

matasano SECURITY

wuntee vs software round 3 - operating system

matasano SECURITY

Reversing

- Focus
  - sbin/*.sh
  - boot procedure
  - binaries using ‘_eval’

matasano SECURITY

PICO_CONFIG

- tamper_proof – this seems to be the configuration of the ‘tamper’ pins on the front and back of the board. One of the applications actually allows you to set the device in ‘learn mode,’ which presumably writes the current pin configuration.
- There looks like firmware images on a 192.168.157.186 host
- There is a firewall node that resembles what is being seen at boot

PICO_CONFIG + IPtables boot

```
[ firewall ]--[ pf ]--[ enable ]--[ 1 ]
                      [ num ]--[ 3 ]
                      [ 0 ]--[ proto ]--[ tcp ]
                             [ port ]--[ 80 ]
                             [ dstip ]--[ 192.168.157.186 ]
                      [ 1 ]--[ proto ]--[ tcp ]
                             [ port ]--[ 22 ]
                             [ dstip ]--[ 192.168.157.186 ]
                      [ 2 ]--[ proto ]--[ tcp ]
                             [ port ]--[ 8080 ]
                             [ dstip ]--[ 192.168.157.186 ]
                    [ 3 ]--[ proto ]--[ tcp ]
                             [ port ]--[ 20000 ]
                             [ dstip ]--[ 192.168.157.186 ]
                [ enable ]--[ 1 ]
                [ snat ]--[ enable ]--[ 0 ]
                          [ num ]--[ 0 ]
```

```
[FW] [C]iptables -t nat -A PREROUTING -p tcp --dst 0.0.0.0 --dport 80 -j DNAT --to
192.168.157.186:80
[FW] [C]iptables -t nat -A PREROUTING -p tcp --dst 0.0.0.0 --dport 22 -j DNAT --to
192.168.157.186:22
[FW] [C]iptables -t nat -A PREROUTING -p tcp --dst 0.0.0.0 --dport 8080 -j DNAT --to
192.168.157.186:8080
```

matasano SECURITY

```
loc_412328:
la      $a0, 0x460000
nop
addiu   $a0, (dword_465A20 - 0x460000)
lw      $a0, (dword_465A20 - 0x465A20)($a0)
la      $a1, 0x420000
nop
addiu   $a1, (aIptablesTNat_2 - 0x420000)  # "iptables -t nat -A PREROUTING -p %s --d"...
move    $a3, $s5
sw      $s1, 0x48+var_34($sp)
sw      $s0, 0x48+var_30($sp)
sw      $s0, 0x48+var_38($sp)
la      $t9, cs_execute_id
nop
jalr    $t9 ; cs_execute_id
nop
lw      $gp, 0x48+var_28($sp)
b       loc_412254
addiu   $s2, 1
```

```
la      $t9, cs_log
nop
jalr    $t9 ; cs_log
nop
lw      $gp, 0x440+var_430($sp)
move    $a0, $s0
la      $a1, 0x420000
nop
addiu   $a1, (aDevNull21 - 0x420000)  # ">/dev/null 2>&1"
la      $t9, strcat
nop
jalr    $t9 ; strcat
nop
lw      $gp, 0x440+var_430($sp)
nop
la      $v1, 0x420000
nop
addiu   $v1, (aC - 0x420000)  # "-c"
addiu   $a0, $sp, 0x440+var_28
move    $a1, $zero
move    $a2, $zero
move    $a3, $zero
la      $v0, 0x420000
nop
addiu   $v0, (aSh - 0x420000)  # "sh"
sw      $s0, 0x440+var_20($sp)
sw      $v0, 0x440+var_28($sp)
sw      $v1, 0x440+var_24($sp)
sw      $zero, 0x440+var_1C($sp)
la      $t9, _eval
nop
jalr    $t9 ; _eval
nop
lw      $gp, 0x440+var_430($sp)
lw      $ra, 0x440+var_8($sp)
lw      $s2, 0x440+var_10($sp)
lw      $s1, 0x440+var_14($sp)
lw      $s0, 0x440+var_18($sp)
move    $v0, $zero
jr      $ra
addiu   $sp, 0x440
 # End of function cs_execute_id
```

_eval("sh -c [IPTABLES STRING]")

matasano SECURITY

back to uboot

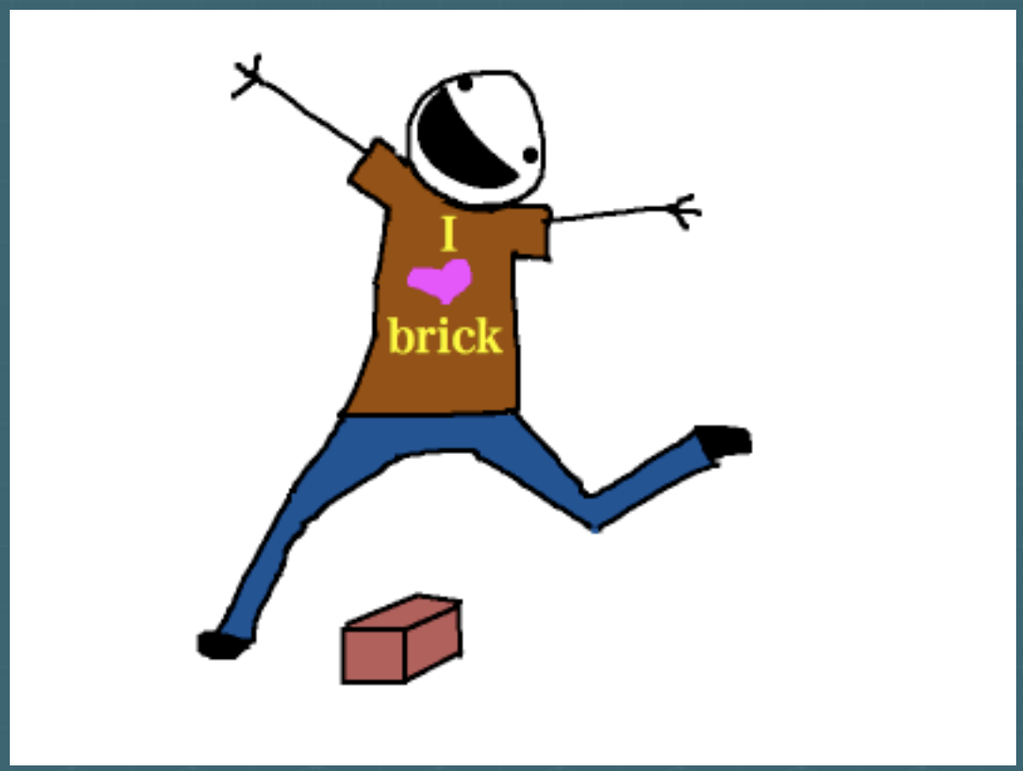
- PICO_CONFIG lives in flash
- MD - ability to display memory...
- MW - ability to write memory
- FAIL... Nothing was working
  - Protect?

uboot memory protect...

```
RT2150 # printenv
bootcmd=tftp
bootdelay=3
...
flash_self=run ramargs addip addmisc;bootm $(kernel_addr) $
(ramdisk_addr)
kernel_addr=BFC40000
u-boot=u-boot.bin
load=tftp 8A100000 $(u-boot)
u_b=protect off 1:0-1;era 1:0-1;cp.b 8A100000 BC400000 $
(filesize)
loadfs=tftp 8A100000 root.cramfs
u_fs=era bc540000 bc83ffff;cp.b 8A100000 BC540000 $(filesize)
...
stdout=serial
stderr=serial
ethact=Eth0 (10/100-M)

Environment size: 829/65532 bytes
```

u_b = brick #2



matasano SECURITY

wuntee vs software
round 3 - linux kernel image

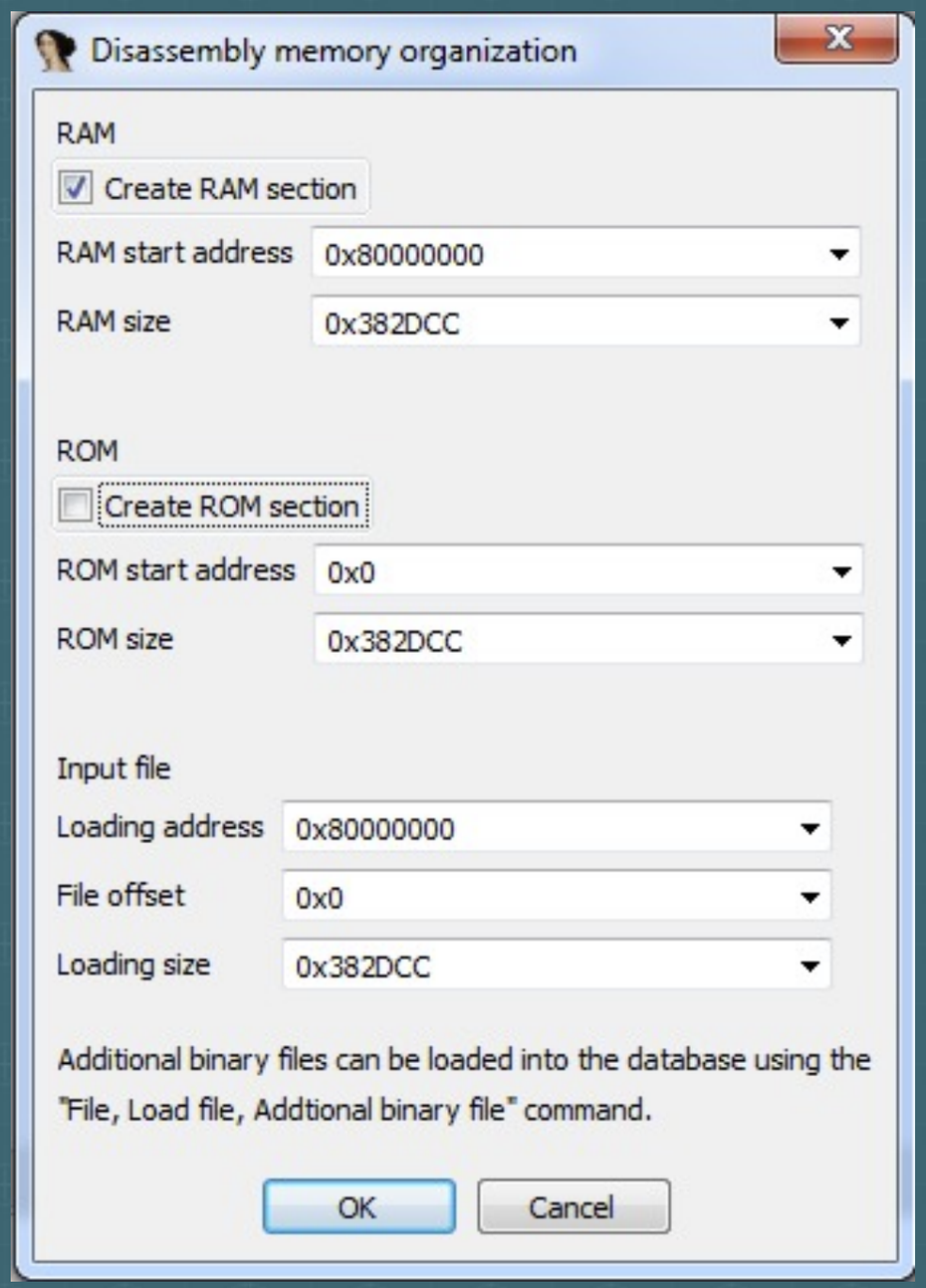
Wile waiting for a new microcell...

- /etc/passwd - John The Ripper
  - In some weird format I have never seen before (13 characters)
  - No faith

- Loading the kernel image

```
DECIMAL          HEX              DESCRIPTION
…
2228224          0x220000         uImage header, created: Thu Mar
4 03:17:29 2010, image size: 1690167 bytes, Data Address:
0x80000000, Entry Point: 0x802A0000, CRC: 0x70DC4C09, OS: Linux,
CPU: MIPS, image type: OS Kernel Image, compression type: lzma,
image name: Linux Kernel Image
```

IDA



matasano SECURITY

no function names, but funciton strings?

Functions window | Output window

Function name	Segment	Start	Length	R
sub_8007A68C	RAM	8007A68C	00000018	R
sub_8007A6A4	RAM	8007A6A4	000000E8	R
sub_8007A78C	RAM	8007A78C	00000034	R
sub_8007A854	RAM	8007A854	00000050	R
sub_8007A8A4	RAM	8007A8A4	000000AC	R
sub_8007A950	RAM	8007A950	0000013C	R
sub_8007AA8C	RAM	8007AA8C	000002FC	R
sub_8007AD88	RAM	8007AD88	000002F8	R
sub_8007B0D4	RAM	8007B0D4	00000080	R
sub_8007B154	RAM	8007B154	000000A0	R
sub_8007B1F4	RAM	8007B1F4	00000088	R
sub_8007B27C	RAM	8007B27C	0000007C	R
sub_8007B2F8	RAM	8007B2F8	00000120	R
sub_8007B418	RAM	8007B418	00000164	R
sub_8007B57C	RAM	8007B57C	00000440	R
sub_8007B9BC	RAM	8007B9BC	00000034	R
sub_8007B9F0	RAM	8007B9F0	0000008C	R
sub_8007BA7C	RAM	8007BA7C	0000011C	R
sub_8007BB98	RAM	8007BB98	00000088	R
sub_8007BC90	RAM	8007BC90	00000110	R

"..." Strings window

Address	Length	Type	String
"..." RAM:8023...	0000000F	C	blk_do_ordered
"..." RAM:8028...	00000012	C	blk_dump_rq_flags
"..." RAM:8028...	00000010	C	blk_end_sync_rq
"..." RAM:8028...	0000000F	C	blk_execute_rq
"..." RAM:8023...	00000016	C	blk_execute_rq_nowait
"..." RAM:8028...	00000016	C	blk_execute_rq_nowait
"..." RAM:8028...	0000000E	C	blk_free_tags
"..." RAM:8028...	00000019	C	blk_get_backing_dev_info
"..." RAM:8028...	0000000E	C	blk_get_queue
"..." RAM:8028...	00000010	C	blk_get_request
"..." RAM:8028...	0000000F	C	blk_init_queue
"..." RAM:8028...	00000014	C	blk_init_queue_node
"..." RAM:8028...	0000000E	C	blk_init_tags
"..." RAM:8028...	00000013	C	blk_insert_request
"..." RAM:8028...	00000010	C	blk_max_low_pfn
"..." RAM:8028...	0000000C	C	blk_max_pfn
"..." RAM:8028...	00000010	C	blk_plug_device
"..." RAM:8023...	00000010	C	blk_plug_device
"..." RAM:8028...	0000000E	C	blk_put_queue
"..." RAM:8028...	00000010	C	blk_put_request
"..." RAM:8027...	00000011	C	blk_queue_bounce

matasano SECURITY

memcpy

- String at 0x8027CDE8
- Does address exist anywhere else? (using search -> sequence of bytes in ida)

```
RAM:8027CDE8                    .byte 0x6D  # m
RAM:8027CDE9                    .byte 0x65  # e
RAM:8027CDEA                    .byte 0x6D  # m
RAM:8027CDEB                    .byte 0x63  # c
RAM:8027CDEC                    .byte 0x70  # p
RAM:8027CDED                    .byte 0x79  # y
RAM:8027CDEE                    .byte    0
RAM:8027CDEF                    .byte    0
```

```
RAM:80276288                    .byte 0xC0  # +
RAM:80276289                    .byte 0xC8  # +
RAM:8027628A                    .byte  0xF
RAM:8027628B                    .byte 0x80  # Ç
RAM:8027628C                    .byte 0xE8  # F
RAM:8027628D                    .byte 0xCD  # -
RAM:8027628E                    .byte 0x27  # '
RAM:8027628F                    .byte 0x80  # Ç
RAM:80276290                    .byte 0x60  # `
RAM:80276291                    .byte 0xCB  # -
RAM:80276292                    .byte  0xF
RAM:80276293                    .byte 0x80  # Ç
```

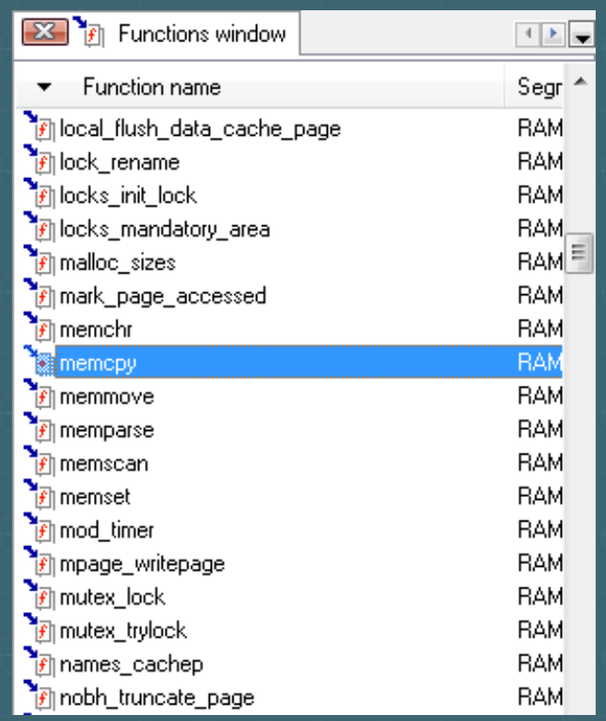
function strings

- Linked list of : [function_name_pointer][funciton_pointer]
- Ruby to strip and create IDA script...

```
#include <idc.idc>

static main() {

MakeName(0x8028afc8, "init_mm");
MakeName(0x80383000, "init_task");
MakeName(0x80383008, "system_state");
MakeName(0x8028b4d4, "reset_devices");
MakeName(0x80002c1c, "loops_per_jiffy");
MakeName(0x80004920, "init_uts_ns");
MakeName(0x80384000, "get_surfboard_sysclk");
MakeName(0x80004a2c, "allocate_irqno");
MakeName(0x8038502c, "free_irqno");
MakeName(0x80006390, "pm_power_off");
MakeName(0x8022a380, "__up");
MakeName(0x8022a4d8, "__down");
MakeName(0x80384040, "__down_interruptible");
MakeName(0x80384274, "cpu_data");
```



matasano SECURITY

wuntee vs software round 4 - gpl

GPL

Where specific free/open source license terms (such as the GNU Lesser/General Public License) entitle you to the source code of such software, that source code will be available to you at cost from [COMPANY] for at least three years from the purchase date of your product. If you would like a copy on a CD of such open source code, upon written request and receipt of payment of $9.99 (to cover shipping and handling costs), [COMPANY] will mail to you a copy. Please send your written request and check payment (payable to [COMPANY]), together with your name, mailing address, email address and phone number to:

email and wait...

- Johnathan the ripper had success... after 7 days
  - root/sshd = 7 character, lowercase a-z

GPL

- DPH151_V1.0.25-5.tar.gz – This is the full build chain for the device that will allow you to build an image file for the device on Ubuntu OS. It contains a configuration file that allows full control of what applications are included in the final image.
- ip.access-AP-IPA1.0-3.zip – This seems to be source code for another (PICO) processor on the board. It does not contain a full build chain. It is just the source code for specific packages and patches, as well as the licenses for the associated packages.

RALink Internals

- Architecture - GPIO to boot pico, DHCPD 192.168.157.185/30
- IPTables - NATs 80, 22, 8080 to .186(pico)
- ipcserver - Router/PICO IPC mechanism
- wizard - Remote commands via multicast
- cfg_flash - backdoor
  - cfg_flash -s -n backdoor -v 1 = bind telnet to 0.0.0.0

wizard

- http://fail0verflow.com - Remote commands via multicast/BackdoorPacketCmdLine_Req
  - That was intended developer functionality
  - If devs were smart, they would remove that function... however

BackdoorPacketLoadSerialNum_Ack

```
loc_402978:
la      $a0, 0x400000
nop
addiu   $a0, (aUsrSbinCs_cl_7 - 0x400000)  # "/usr/sbin/cs_client set sys/serial_num "...
la      $t9, evalsh
nop
jalr    $t9 ; evalsh
nop
lw      $gp, 0x30+var_18($sp)
nop
la      $a0, 0x400000
nop
addiu   $a0, (aUsrSbinCs_cl_3 - 0x400000)  # "/usr/sbin/cs_client commit"
la      $t9, evalsh
nop
jalr    $t9 ; evalsh
nop
```

```
aUsrSbinCs_cl_7:.ascii "/usr/sbin/cs_client set sys/ser"
.ascii "ial_num %s"<0>
```

wuntee vs microcell

winner wuntee!

matasano SECURITY

how many bricks did it take?

matasano SECURITY

thanks

- rajendra umadas - intrepidus group
- jeremy allen - intrepidus group
- cory benninger - intrepidus group
- kurt rosenfeld
- andrew zonenberg - rensselaer polytechnic institute
- travis goodspeed

matasano SECURITY

questions?

mathew rowley

mathew@matasano.com

@wuntee

http://www.matasano.com/research/

http://67.219.122.21/blackhat2012/