whois donb?
whatis iSEC Partners?
Technology is The Great Equalizer
As Technology Increases, Control Decreases
BRUCE WAYNE!

☆%&!! WIKILEAKS
libya
algeria
bahrain
egypt
egypt
support the people's
revolution
Aboard The Lulz Boat
Examples of Emerging Technology?
GlowCaps™
light and sound
remind you to take your prescriptions every day
No, really.

• Cellular enabled pill bottles
• Track pill usage remotely
• Email alerts when
  ▫ Pill count is low
  ▫ Pills haven’t been taken
  ▫ When its time to take your pill
Wait. That sounds bad.
But, it’s helping people.

- Alzheimer’s patients
- Children with severe diseases
- Physically disabled patients
- Overworked security consultants
Wait. That sounds good.
Hey, I'm being followed by monkeys!
Everything will be a computer
Examples?

- Medical devices (personal, industrial)
- Industrial monitoring
- Automated Teller Machines
- Industrial/Commercial Alarm Systems
- Home Alarm Systems
- ...Car security systems
Common M2M Example from Microchip

M2M PICtail™ Daughter Board
Find Architectural Commonalities

• Baseband
  ▫ modules must be approved
  ▫ The approved list is public
  ▫ few features
  ▫ can’t drive Application Logic

• Microcontrollers
  ▫ Small RAM
  ▫ Small Code Space (flash)
  ▫ Minimal security surface (if any)
Find Architectural Commonalities

• Communication
  ▫ Network Comm = Baseband
  ▫ Peripheral Comm = uC
  ▫ Comm between Baseband & uC = UART

• Cryptographic Capability
  ▫ Only *some* Basebands provide HTTPS/SSL
    • Usually only Java VM capable
  ▫ uC is usually baked (or non-existent)
Easiest Way to Attack?

- **Sniff**
  - USART
  - SPI
  - I2C
- **Debug ports**
  - JTAG
  - SWIM
  - DebugWire
  - etc
The GoodFET
Open Source JTAG Adapter (and more)

- SPI
- I2c
- JTAG
- AVR
- Glitching
- SmartCard
- NordicRF
- PIC
Architecture

• Simple hardware architecture
  ▫ Few components
  ▫ Open Source

• Simple software architecture
  ▫ Python based
  ▫ Open Source
AVR Port

- Simple hardware architecture
  - Few components
  - Open Source

- Simple software architecture
  - Python based
  - Open Source
AVR GoodFET Requirements

- Simple board design
- Boot loader needed
- No soldering!
- Portable to almost any Atmel AVR
- Cheap!!
- Components must be easily accessible
  - world wide
AVR GoodFET Hardware

- ATmega1284P
- One pull-up resistor (1K Ohm)
- One 0.1uF and one 1uF capacitor
- 20MHz external clock (Abracon ACHL-20MHz)
- FTDI Cable
AVR Boot Loader

- 20MHz
- 0.5M USART baud rate
- Flash from file
- Flash from web
- Peek
- Signature
- Fuse bytes
- Page Size
boot loader found, entering command mode.
donbL> pagesz
donbL: retrieving page size
pagesize: 100
donbL> peek 0x0
donbL: peeking address
peek 0x0: 940c
donbL> peek 0x02
donbL: peeking address
peek 0x02: 0050
donbL> fuse
donbL: retrieving avr fuse and lock bytes
fuse: ff ff 9c e0
donbL> signature
donbL: retrieving avr signature
signature: 1e 97 05
rc calibration: 51
donbL>
AVR Boot Loader

- Shouldn’t have to know Chip
  - Requirement of Travis’
- From web & signature = solution
  - Request sig (1E9705)
  - Download per-sig image 1E9705.hex
  - Flash image
- Fuses can be validated per signature
  - Each chip has slightly different fuses
boot loader found, entering command mode.

donbl> flash goodfet.hex
donbl> flashing image

pagesz: 100

flashing pages: ............................................................
donbl> reset

boot loader found, entering command mode.
donbl> signature
donbl> retrieving avr signature
signature: 1e 97 05

rc calibration: 51
donbl>
Boot Loader Bugs

- A section can’t exceed one file
- Can’t use .data, .bss
- Word address versus Byte address
- Vectors are /required/
- IVTs must get naked (ISR -> BL_ISR)
- WatchDog spinlock
- Pgm_read_byte_far() is buggy
- Undocumented bits in –P models (SIGRD)
AVR Port Code

- Build library files
- Integrated “donbfet” support
- Adjusted for silly AVRnesses
- Go!
GoodFET with O000 MCU
Clocked at 0x0000
Build Date: 2011-10-10 23:31
Firmware apps:

Monitor
The monitor app handles basic operations on the MSP430 such as peeking and poking memory, calling functions and managing the baud rate.

SPI
The SPI app handles the SPI bus protocol, turning your GoodFET into a USB-to-SPI adapter.

AVR
The AVR app adds support for debugging AVR based devices.

JSCAN
The JScan app adds support for JTAG brute-force scanning.

Identifies as Atmel O9705, lock=ff
JTAG Scanning
What is JTAG?

- Standard for debugging/monitoring chips
- Originally used to test manufactured equipment
- Used to test/debug embedded devices
- Simple state machine protocol
- Daisy chain-able
- Field updates!
What is JTAG?

- **5 Pins**
  - TCK – Clock
  - TMS – Mode Select
  - TDI – Data In
  - TDO – Data Out
  - TRST – Reset

- **TRST is “optional”**
  - Not always (AVR)
JScan Application

• 646 Lines of C (firmware)
• 143 Lines of Python (client)
• Dynamic Pin definition
• Control endianness
• Select delay (pin state sync)
• Store/retrieve results
• Core is based on
  ▫ Hunz’s slides
  ▫ ArduiNull (LeKernel)
How Do We “See” JTAG?

- 11111b is Always a state machine Reset
- Then
  - 0: Run Test Idle
  - 1: Select DR
  - 1: Select IR
  - 0: Capture IR
  - 0: Shift IR
- Shift IR activates TDO
- Shift in via TDI, monitor TDO
Hunz’s Method

- Only 4 pins are required
- Yes, still NRST
- Still N! operations
- Approximately 120 tests per minute
```
donb@localhost $/lib/src/goodfet/c/goodfet/trunk/client $ ./jscan.sh
scan verb: 7f
scan count: 01
addpin returned ID 80
scan verb: 7f
scan count: 01
addpin returned ID 81
scan verb: 7f
scan count: 01
addpin returned ID 82
scan verb: 7f
scan count: 01
addpin returned ID 83
scan verb: 7f
scan count: 01
addpin returned ID 84
Endian is 0x01
Endian set 0x01
Delay is 0x01
Delay set 0x01
donb@localhost $/lib/src/goodfet/c/goodfet/trunk/client $ platform=donfbet ./goodfet.jscan
scan verb: 7f
donb@localhost $/lib/src/goodfet/c/goodfet/trunk/client $ platform=donfbet ./goodfet.jscan results
scan verb: 7f
scan count: 05
tck=80, tms=82, tdi=83, tdo=81, nrst=84
```
Results

• ~0.55% FP rate
  ▫ 5 pins
  ▫ 6 pins
  ▫ 7 pins
  ▫ 8 pins
• @20MHz, 120 tests per minute
• Pull-ups are required
• False positives are easy to detect
• Output arrays should feed other Apps
Issues

• False positives often drive invalid states
  ▫ Logic gate w/ power control
• Delays should be adjusted when $R = 0$
• 220 – 330 Ohm resistors Must be used
• Output -> App requires dynamic Pin control
• Can only fit $\sim$100 results in response
  ▫ Limited by GoodFET protocol
Future Requirements

- Select “Profile” mode (i.e. AVR, ARM, etc)
- Fingerprint JTAG subtleties
- Automated target power control ala JTagger
- Apps should interleave
- Protocol scanning should be generic
  - Pattern based
- Language should define pattern
Demo
Summary?

- Need More Tools like GoodFET and UberTooth
- Opening up GoodFET’s arch further will help
- JTAG scanning is easy
- Integrating it is hard
- Other protocols are needed
Thanks to...

- iSEC Partners
- Travis Goodspeed
- Mike Kershaw
- Mike Ossmann
- Nick DePetrillo
- hunz@hunz.org
- LeKernel.net
“Pull up the people. Pull up the poor.”
- M.I.A.