Methodologies For Hacking Embedded Security Appliances

(How we break really expensive boxes)

About The Speakers

- Mark Carey is the Chief Scientist for Peak Security.
  - Over 22 years of experience in engineering and security.
  - Keeper of secrets, bringer of ideas.
- Rob Bathurst if the VP of Engineering.
  - Has over 12 years in security design and large-scale engineering projects.
  - Herder of cats, wrangler of interns.

Peak Security, Inc.

- The Company
  - Peak is a veteran owned small business specializing in innovation and creative solutions to security problems.
  - If you have a security problem, that no one else can help with, you might be able to hire the Ghostbusters; failing that, there's us.
    - Just kidding
  - http://www.peaksec.com

Before We Get Started

- We'd like to thank some of the people who's work helped us make this possible
  - Joe Grand (You should all know him from TV)
  - Emerson Tan (The wonders of methylene chloride)
  - Flylogic's silicon device attacks (http://www.flylogic.net/blog/)
  - Everyone who's ever screwed with an Arduino, AVR, ARM, MIPS, etc.
  - Finally to our families who gave us support as we littered our houses with archaic electronics

Methodologies

- Problems with the industry
  - The security industry has very few public methods for hardware evaluation
  - Many consider it to be “voodoo”
  - We have organizations that create methods (SANS, OWASP, etc.), but none that have a good one for hardware

Defining A Good Method

- Must be repeatable by a competent hardware/software engineer
- Must not require unreasonable equipment
- Must not require huge $$$
- With a little effort and desire can actually be completed

Addressing The Gap

- The series of steps we'll present will help you evaluate a device for possible security vulnerabilities
- Show a series of repeatable tasks that will allow you to begin the hardware and software evaluation
- We'll be standing up a site open to the community to be used to share information on devices that people are working on cracking

Getting Down To Business

- You'll need a few key pieces of physical equipment and software tools
- Most of these items are not expensive and can be acquired for $500-$1000
- A few of the big items (we'll cover these in detail)
  - Your Brain
  - A Voltmeter
  - Surface Mount Soldering/Hot Air Rework Station
  - Soldering Stuffs
  - Magnifying Glass
  - Microscope
  - Bus Pirate
  - Spare Parts
  - Debugging Interfaces
  - IDA Pro

Your Brain

- Remember SAFETY SAFETY SAFETY
- Electricity can kill and maim and kill
- Always be aware of your surroundings when soldering
- Wear safety glasses
- Don’t die

A Voltmeter

- Absolutely needed to do circuit probing
- Used to test various parts for electrical resistance
- Needed to test the circuit voltage so you don’t destroy your Bus Pirate
- Check diode conductivity
- USB Volt Meters are great for recording directly to your computer
- Volt Meters can go from very cheap all the way into the high hundreds (Fluke)
- We’ll be using a $60 USB model from SparkFun Electronics

PEAK SECURITY

Surface Mount Soldering/Hot Air Rework

- Can be bought on Amazon for ~$160 for a decent model
- Extremely good for removing surface mount components without destroying your board

PEAK SECURITY

Soldering Elements

- We'll need some of these items to add and remove components to the board
- Solder wick (used to remove solder)
- Insulated tweezers or micro-forceps
- Solder
- Flux
- Chip puller

PEAK SECURITY

Magnifying Glass

- Go to amazon, they can be found cheaply there
- We're using a rather expensive one (~$180), but they can be super cheap and effective
- The higher power, the better
- Make sure it's got a light
- Pro-tip: Get some Rain-x or anti-fog spray so you won't fog up your glass when you end up breathing on it

USB Microscope

- Used for all kinds of micro examination
- Some things we'll use it for
  - Examining contacts for broken solder
  - Chip numbers (very important)
  - Board traces
  - Anything else we find on our desk when we get board

PEAK SECURITY

A Bus Pirate

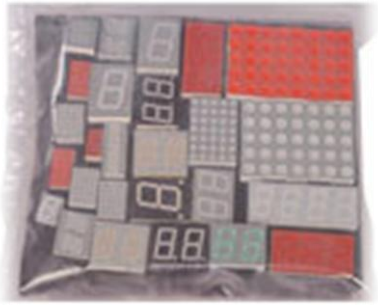
- No, not that kind of pirate
- Used to read and wire to almost every raw “bus protocol” (almost)
- Very gentle learning curve
- Not the best for scripting for things we like to do such as dumping SPI flash or I2C in an automated way.
- Very active community
- Go get one (https://www.sparkfun.com/products/9544)

PEAK SECURITY

Spare Parts

- eBay is great for spare parts. Search for “sample packs” that contain resistors, capacitors, inductors, diodes, LEDs, etc.
- Spare parts are needed to replace blown one. You will at some point wreck one (or more) of these on a board
- An example of the way we’ll use these
  - Using a resistor to tie a pin to ground or VCC to change signals on chips

PARTS!



Debugging Interfaces

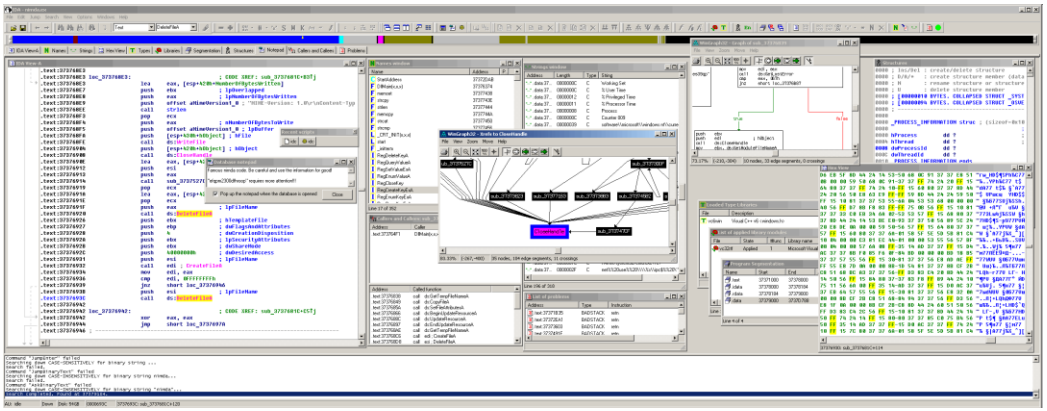
- If the device you're testing does not have a JTAG (Joint Test Action Group) interface, we can put one on
- A good debugging set will contain
  - JTAGs
  - A BDM (Background Debugger Mode Interface)
  - ISP (In-System Programming) Device
- We will use all these in various ways to access the device software/firmware

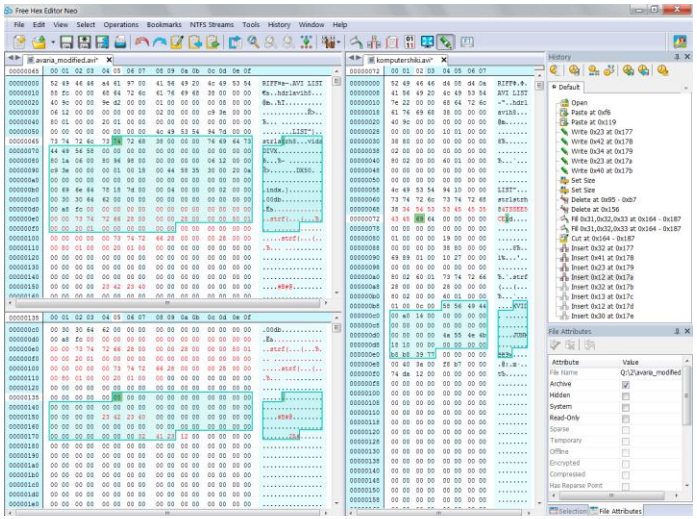
IDA Pro

- Totally worth it, single best way to disassemble and analyze software
- Turns the compiled code into “C like” code for analysis
- Downside, bit on the expensive side
- Hex-Rays also have x86 32bit and ARM decompilers.
- IDA (https://www.hex-rays.com/products/ida/index.shtml)

PEAK SECURITY

IDA Easy / Hex Editor Hard





PEAK SECURITY

Define The Device

- What is its marketing name?
- Is it a third party device (i.e. cable modem)?
- Does it have a non-obvious name?
- Does it have a part number from the manufacture?
- Does it have an FCC ID?

Marketing Name

- The OEM Process introduces a lot of hardware out there that's actually commodity.
  - A large number of PC based "secure appliances" are rebranded Intel servers.
    - EMC devices
    - Sun's V20 and V40 lines
    - RSA Appliances

PEAK SECURITY

Third-Party Devices

- Did someone give it to you or your company?

PEAK SECURITY

Non-Obvious Names and Knockoffs

- Wonderful Chinese knock offs like:
  - The HiPhone or the APhone A6 (Running Android Jelly Bean)

PEAK SECURITY

Manufactures Part Numbers

Board Markings

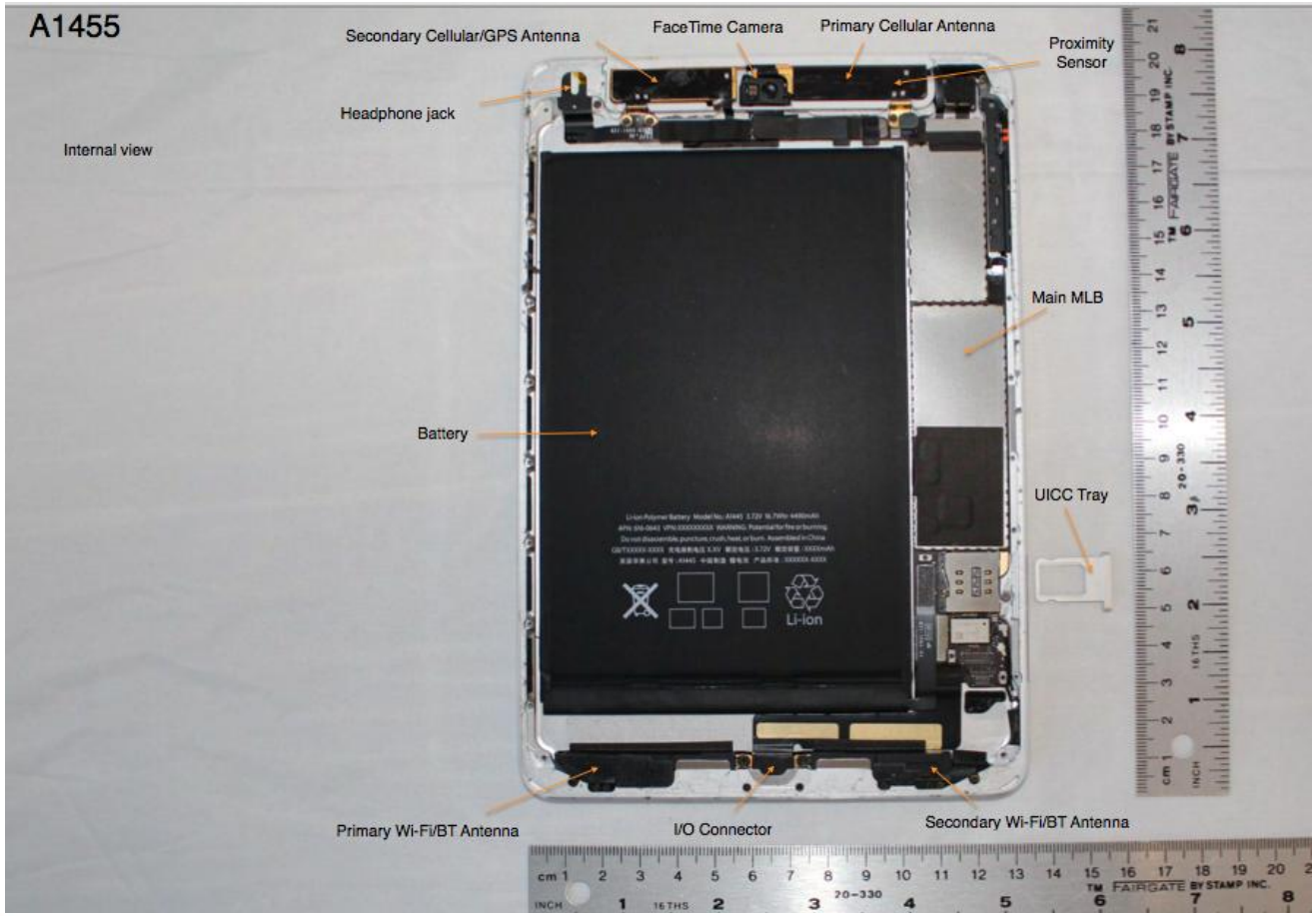
FCC IDs

- FCC IDs are required for almost all modern electronic devices
- This is caused by the RF emission they “might” have during operation
- They will always have them if they have a radio transmitter

FCC IDs Cont'd

- We can find the ID in a nice database (http://transition.fcc.gov/oet/ea/fccid/)
- FCC IDs have 2 parts
  - The Grantee Code (first three letters)
  - Product Code (the rest)
- Example: iPad mini with FCC ID BCGA1455
  - If we search for it we get 26 results
  - Click the detail of the top entry and get all the filing documents
  - Chose Internal 2 (we just like it)
  - See the pretty insides
  - We can get all kinds of stuff in these documents like user manuals and more diagrams/drawings

FCC IDs Part 15… Err 3.



No Board Numbers!?!

- Remember that chips don't lie (normally)
- Catalog the chips on the board (take lots of photos)
- Lots of "custom" chips come from excess and end up on eBay or chip wholesalers
- An example would be a device labeled "Spansion" has a high chance of being a NAND flash EEPROM device

How To Get Chip Information

- Most chip manufactures will “private market” parts that are very close to their public market chips
  - Marvell, for example, makes a large number of silicon devices for Seagate, Western Digital, and Samsung
  - These “private chips” are usually embedded ARM processors with additional parts like memory IO peripherals (Marvell uses an 88i prefix code for these a lot of these chips)

How To Get Chip Information Cont'd

- You can often derive information about a given chip from similar chips by the manufacture
  - Remember all chip manufactures have a NRE cost to their chips, so the more they can reuse designs the better
  - You can get things like the location of JTAG or BDM ports, pins to apply voltage to, addressing, data, bus connections, etc.
  - We can also make some educated guesses about whether a chip might be a SPI Flash or I2C EEPROM device

How To Get Chip Information Cont'd

- You can get all kinds of information from public sources
  - Google (who doesn't use Google?)
  - Mouser
  - DigiKey
  - Manufactures website
  - Call the manufacture, sales people love to talk

PEAK SECURITY

How To Get Chip Information Cont'd

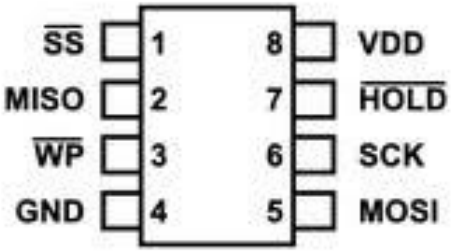
- Ways to identify a component
  - Look for logos (used to save space)
  - Lookup visual chip directories
    - http://how-to.wikia.com/wiki/Howto_identify_integrated_circuit_(chip)_manufacturers_by_their_logos/all_logos
    - *http://www.advanced-tech.com/ic_logos/ic_logos.htm*

How To Get Chip Information Cont'd

- Still can't find anything?
  - You're screwed (just kidding, mostly)
- We can still succeed
  - Examine all the board components
  - Look for the power feed, trace it to the PMIC (power regulation components)
  - Identify the ground plane, this will help identify chips by pin out

Common Layout Components

- SPI chips are laid out like this:

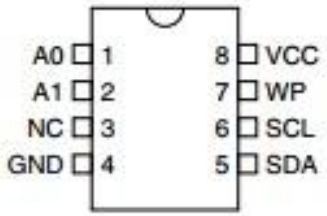


- I2C chips are laid out like this:

Pin Configurations

Pin Name	Function
A0–A1	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect

8-lead PDIP



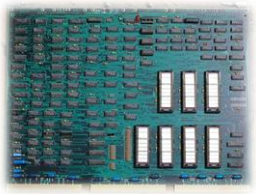
Notice anything odd about them?

- SPI and I2C can be identified by the position of the VSS (ground) and VCC/VDD (positive) in most cases
- Pin 4 and 8 are Ground and VCC.
- If the chip has a write protection tied to the Vcc through a 4.7k Ohm resistor it's most likely a SPI flash ROM
  - Why?
  - Because it keeps the chip writable by the software for firmware updates

PEAK SECURITY

How To Get Chip Information Fin

- You should now have enough info to estimate if the device is:
  - A power conservation system
  - A fully functional computer
  - A IO sub-processor from a VAX
  - Mystery meat



- Also remember to take the date into consideration
  - When did the chip arrive on market
  - Is it out of production now
  - Does it have known weaknesses (clock glitching, power glitching, differential current draw analysis, other side-channel attacks, etc.)

PEAK SECURITY

Bring On The Attack - Physical

- Epoxy Removal 101 (I hate that stuff)
  - Epoxy is:
    - An adhesive, plastic, paint, or other stuff made from a class of synthetic thermosetting polymers containing epoxide groups
  - What does this mean for us:
    - This stuff is a pain to remove after it dries
    - It could contain some dangerous chemical
    - We use polyfunctional amines, acids, acid anhydidres, phenols, alcohals, and thiols to remove various epoxies. DON'T TRY THIS AT HOME.
  - SAFETY TIPS:
    - Do this ONLY in a well-ventilated area.
    - Do this ONLY someplace fireproof.
    - Do this ONLY on a TEST DEVICE, not the one you need to get the info from
    - Do this with a buddy
    - Use a respirator (hard to see, but good not to die)
    - Be aware of any DMCA violations you may be running afoul of

Bring On The Attack - Physical

- Heat Removal
  - The simplest way to remove epoxy
  - Heat removal relies on two principles
    - Thermal differential to cause micro-fracturing between the board the epoxy
    - Most bonding agents will relax their homopolymerisation bonds between 200-500 degrees Centigrade, which will allow us to slice them away
  - The heat technique can be used on metal-impregnated epoxies, as well, but may require a much higher temperature, and destroy what you're working to get at.

Bring On The Attack - Physical

- Heat removal prerequisites
  - More ventilation
  - A hot air source (Hot air rework station)
  - A buddy (to pull you off a burning board)
  - A very sharp Xacto knife with a heat resistant handle and a small blade
  - A very sharp Xacto knife with a heat resistant handle and a large blade
    - We use several to allow the hot ones to cool as we keep cutting
  - A magnification station (read what we were talking about earlier)

Bring On The Attack - Physical

- Heat Removal Demo Video
- The Venue is a little funny about chemical experiments on their property.

Bring On The Attack - Physical

- What if the device coms encapsulated in an unusual form factor?
- Cards
  - Circuit cards and Laminate Layer Removal
    - This technique was shown by Emerson Tan and Co.
  - Methylene Chloride Card Facing Technique
    - This technique is appropriate for plastic coated cards, which use a flexible layer circuit material to put traces on, but coat it with a laminated layer of plastic over that circuit “board” material.
    - The technique works by way of dissolving the bonds between the organic molecules in the plastic.
    - It will cause the card’s outer plastic layers to slough off, hopefully leaving the insides in-tact.

Bring On The Attack - Physical

- Obtaining Concentrated Methylene Chloride
  - Methylene Chloride is a compound found in a large number of household compounds in trace amounts.
    - Things like floor refinishing gel are a good bet for low density Methylene Chloride.
    - How you distill it is beyond the scope of this discussion, and due to liability I'm not going to cover it.
  - We will say, if you figure out how, do it outside and be very careful. Talk to a chemist, not the Internet.
  - You can also order it online.
  - Safety Precautions
    - Be sure to use Nitrile gloves to the elbow, and a metal pan!
    - Nitrile is non-reactive with Methylene Chloride.
    - Metal is non-reactive with Methylene Chloride. (Non-alkali metals, like stainless steel, or aluminum really.)
    - DON'T DO THIS AT HOME!

Bring On The Attack - Physical

- Back to the fun!
- Removing the card's outer layers:
  - In order to remove the cards outer layers, steep the card in the gel.
  - Depending on how concentrated the methylene chloride is, and what the ambient temperature is, it will take between 5 and 30 minutes to dissolve the outer layers on the card.

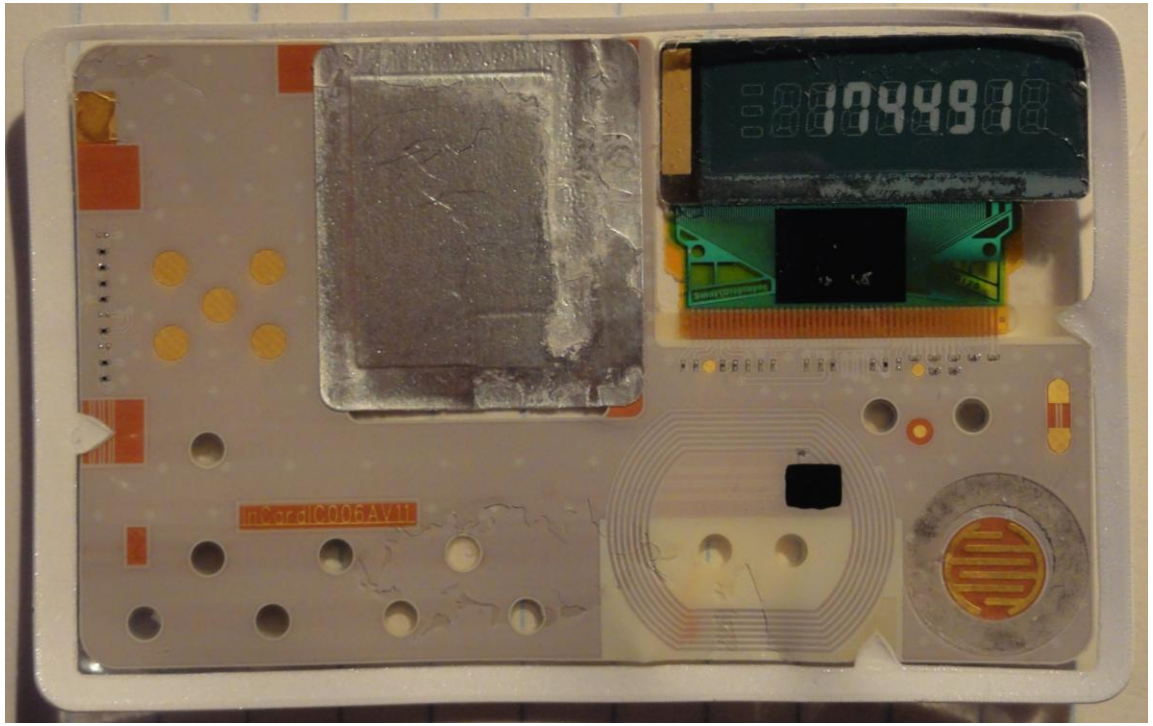
Bring On The Attack - Physical

- Our victim before:
  - PayPal OTP (One Time Password) Card
  - Still has a case on it
  - We don't like that

PEAK SECURITY

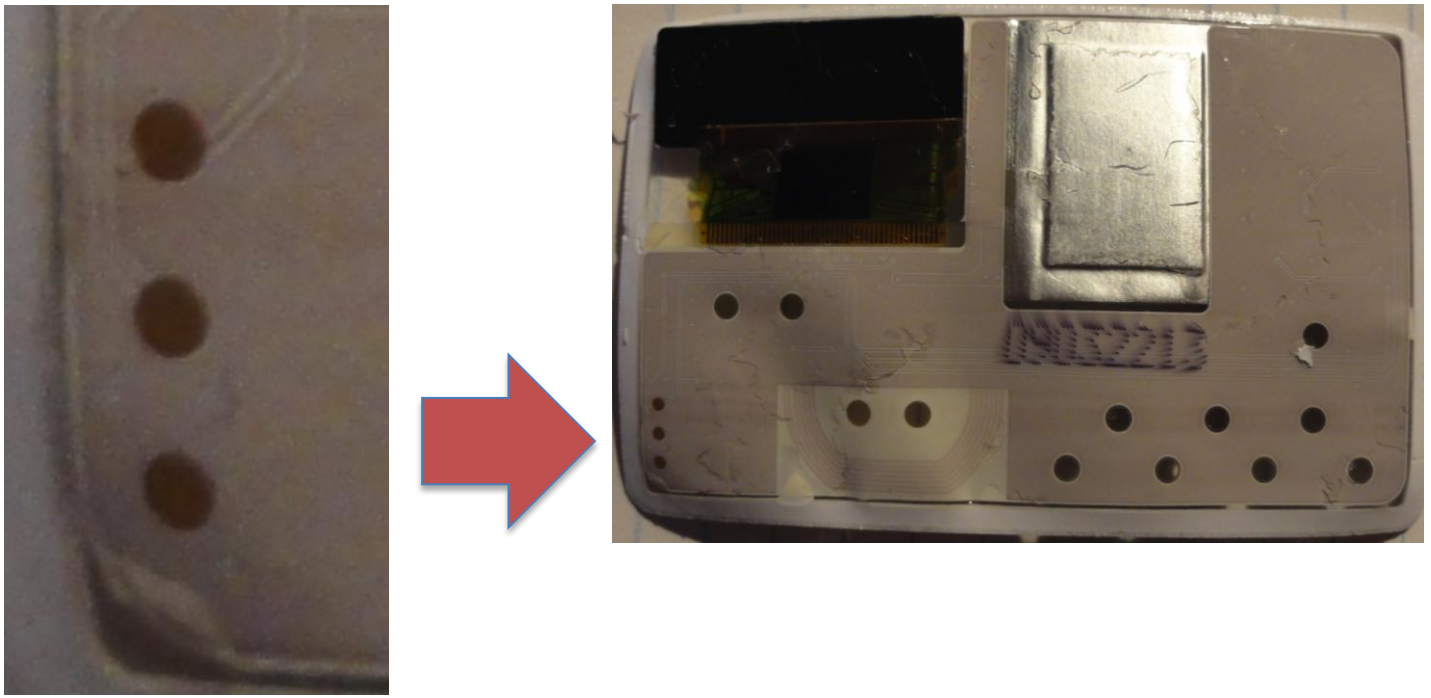
Bring On The Attack - Physical

- Our victim after the bath:
  - Notice the difference
  - Notice the board design
  - Notice the connectors
  - The Silver Foil patch is the battery.



Bring On The Attack - Physical

- Now let's look at the back.
- Notice the three little contacts? Ground, TX, and RX of some sort!



Bring On The Attack - Physical

- Additional physical protections:
  - Welded devices
    - Acid etching
    - Grinder
    - Drilling (precision drill)
  - Security Screws
    - You can get about any bit you need from eBay!
    - Usually cheap and good to have around
  - Evil Plastic Latches
    - Sometimes access deterrents can hurt you
    - Mostly your own fault, very easy to slice yourself opening these
    - Builder tip: ABS high surface area-interlocks are a pain to open
    - These latches can be strong enough that you could damage the board if you're not careful

Bring On The Attack - Physical

- Electrical Intrusion Sensors
  - Carefully examine the case
  - Look for points to drill to expose the sensor wires
  - Can the wires be hook and dragged outside the case?
  - Do you know if the intrusion sensor is passive or active?
  - Could there be magnetic sensors? How to tell safely?
    - Coiled wire hooked to a volt meter is one way. Not a good way, but a way.
    - A Gauss meter (Magnometer) is a good way.
    - iPhones have one called a “Hall Effect Sensor”. It’s the Compass!

Could the device be booby-trapped?

- XRAY and light sensors
- Nitrogen filled cases
- Case contact wires to detect opening
- These are usually a military grade problem

Bring On The Attack - Physical

- Other Device Access Avenues
  - Does the device have a front panel (maybe an LCD) that has access to an internal bus?
    - I2C or SPI for example?
    - Could be attached to the same bus as the startup flash!
  - MiTM on devices that authenticate or download configs from a standard network
  - Hold down button on boot or mode change to access special debug features

Bring On The Attack - Physical

- Some final thoughts on physical access:
  - Physical access always wins
  - All physical protections will fall in time
  - Be aware that when you're evaluating you'll most likely need a couple "donor" devices that will be destroyed.
    - Most "secure" devices get destroyed ☺

Bring On The Attack – Provisioning

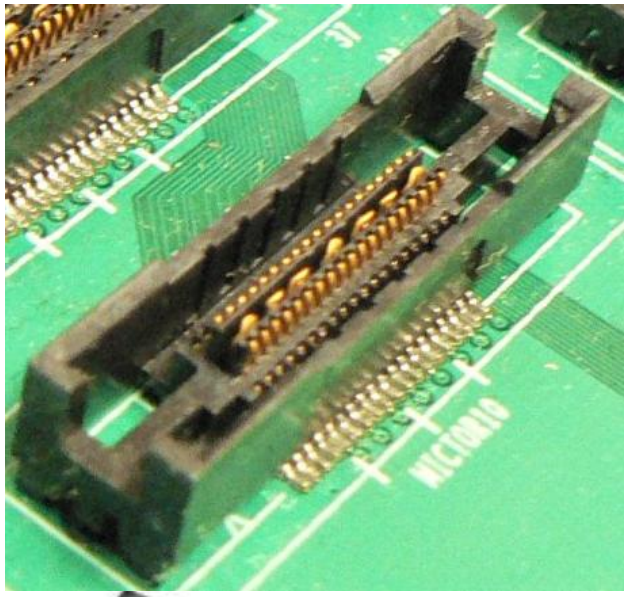
- Things we'll cover under the provisioning section:
  - How provisioning works
  - How devices are provisioned and managed
  - Connectors
  - Debugging Ports

Bring On The Attack – Provisioning

- What is provisioning:
  - Provisioning is the process by which a raw (un-programmed) device is made ready for operation
  - Most devices need to be provisioned, because the device and software are built as separate units
- Every consumer device usually has a provisioning mechanism
- Most devices are a Factory or a Field provisioned device
  - Important to understand the difference, because their methods are different

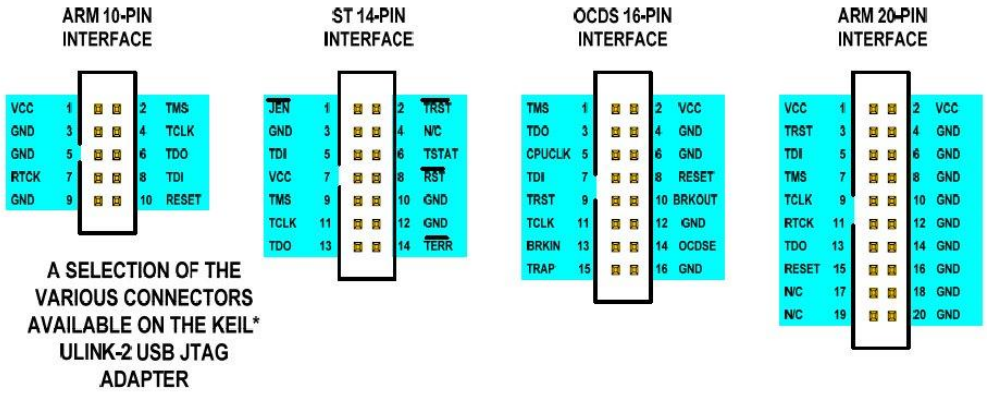
Bring On The Attack – Provisioning

- Identifying common connectors for provisioning:
  - The Mictor from Agilent
    - Pic
    - Why you will hate this connector:
      - $$$
      - Tiny pins, hard to solder
      - Designed for large shops
    - Why you will love it:
      - Impedance matched with ground plane
      - One connector to rule them all



Bring On The Attack – Provisioning

- The ARM standard JTAG



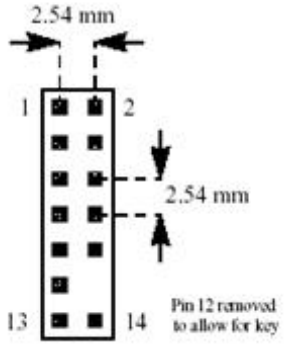
A SELECTION OF THE VARIOUS CONNECTORS AVAILABLE ON THE KEIL* ULINK-2 USB JTAG ADAPTER

Bring On The Attack – Provisioning

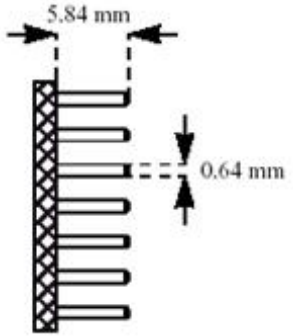
- The MIPS eJTAG

Signal	Pin		Pin	Signal
TRST-	1		2	GND
TDI	3		4	GND
TDO	5		6	GND
TMS	7		8	GND
TCK	9		10	GND
RST-	11		12	Key
DINT	13		14	VIO (Reference Voltage)

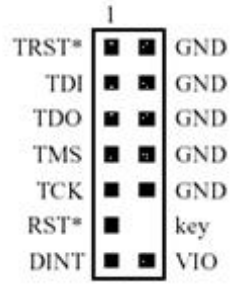
Top view on PCB



Side view on PCB



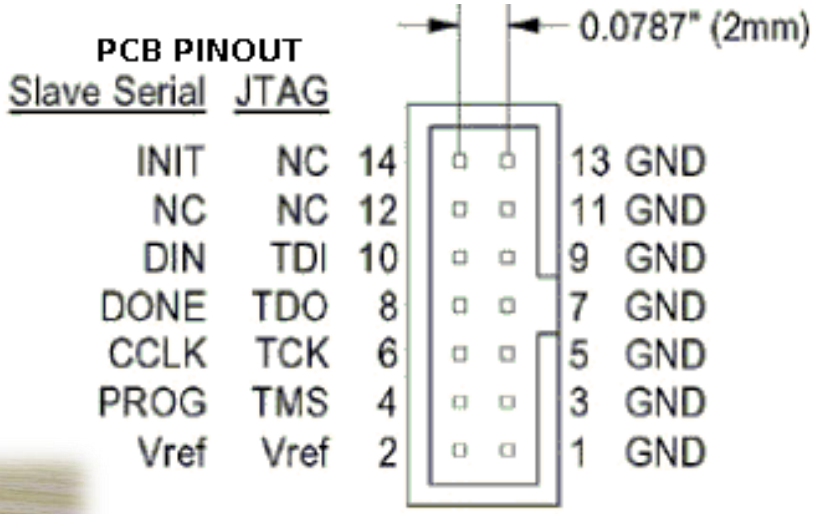
Signal Positions



PEAK SECURITY

Bring On The Attack – Provisioning

- The Xilinx JTAG



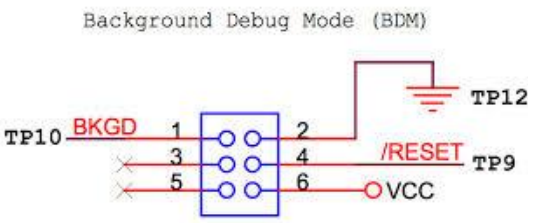
Bring On The Attack – Provisioning

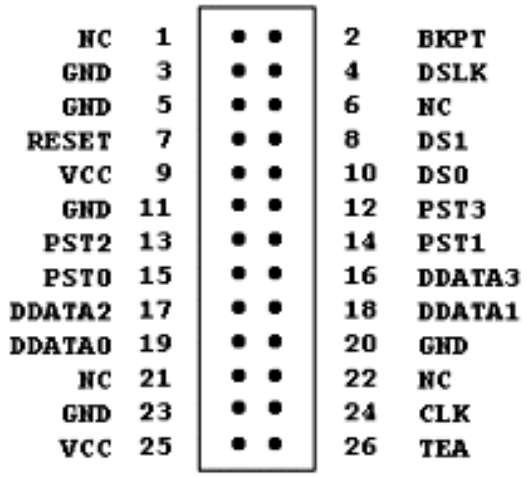
- MSP430 JTAG

PEAK SECURITY

Bring On The Attack – Provisioning

- Motorola / Freescale BDM (Background Debugging Module)
  - 6 pin
  - 26 pin

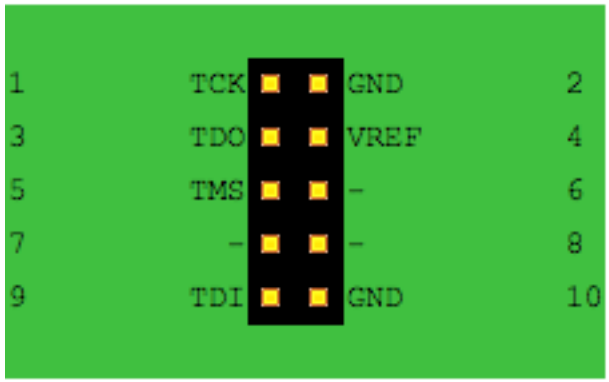




Pin	Signal	Type	Description
1	5V-Supply	-	Target power supply
2	BKPT	Output	Breakpoint
4	DSCLK	Output	Development Serial Clock
6	NC	-	Developer reserved
7	RESET	I/O	Reset
8	DSI	Output	Development Serial Input
9	+5V	-	Voltage supply
10	DSO	Input	Development Serial Output
12	PST3	Input	Processor Status
13	PST2	Input	Processor Status
14	PST1	Input	Processor Status
15	PST0	Input	Processor Status
16	DDATA3	Input	Debug Data
17	DDATA2	Input	Debug Data
18	DDATA1	Input	Debug Data
19	DDATA0	Input	Debug Data
21	NC	-	Motorola reserved
22	NC	-	Motorola reserved
24	CLK_CPU	Input	CPU Clock
25	VCC_CPU	-	CPU Voltage
26	TEA	Output	

Bring On The Attack – Provisioning

- The “common” 10 pin JTAG
- How common is common?
- A very subjective question and it depends on what you’re asked to analyze.



PEAK SECURITY

Bring On The Attack – Provisioning

- The Motorola PPC JTAG
  - Extremely common in a lot of embedded devices

Bring On The Attack – Provisioning

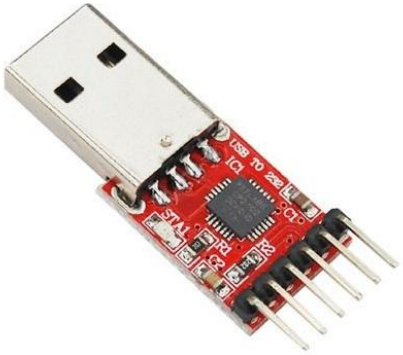
- Others we'll not specifically cover:
  - Lattice ISPDOWNLOAD (JTAG and ISP) 8 and 10 pin
  - IBM RISCWatch 16-pin
  - Motorola "ONCE" On Chip Emulation 14 pin (JTAG)
  - Philips MIPS JTAG 20-pin
  - ST FlashLink 14 pin
  - Xilinx 9 pin (Serial Slave and JTAG)
  - Check out thist site for some decent information onall of these.

http://www.jtagtest.com/pinouts/

Bring On The Attack – Provisioning

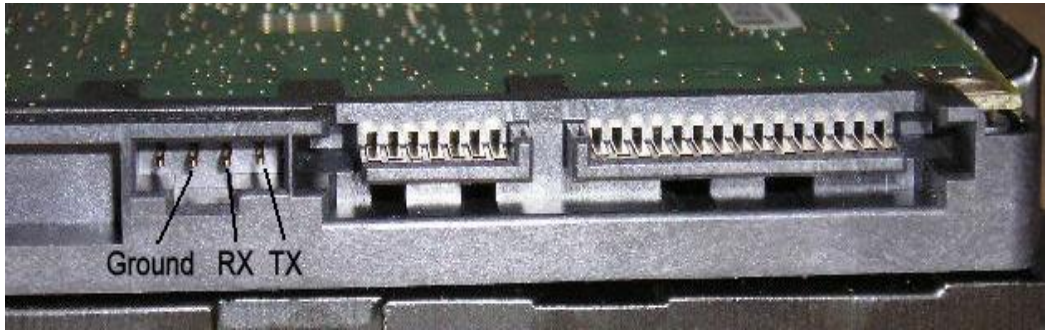
- Our all-time favorite: TTL Serial
  - Three pins, often tied to on chip boot loaders, debuggers, and things the manufacture would hate you to get ahold of
  - Pro Tips:
    - TTL serial mean Transistor-to-Transistor Level Serial
    - This operates at between 3.3 and 5.0 volts (0-VCC technically) for logic signaling in most cases
    - This is not the serial port on a computer (That's RS-232 serial port)
    - This will destroy the chip if you attempt to attach it to the RS-232
    - Buy an adaptor on Amazon, search for FTDI or SILABS Chip, which have manufactures reference drivers available
    - Don't install the drive that comes with the board unless it's signed
    - "Cavet Emptor"

Examples of TTL Serial



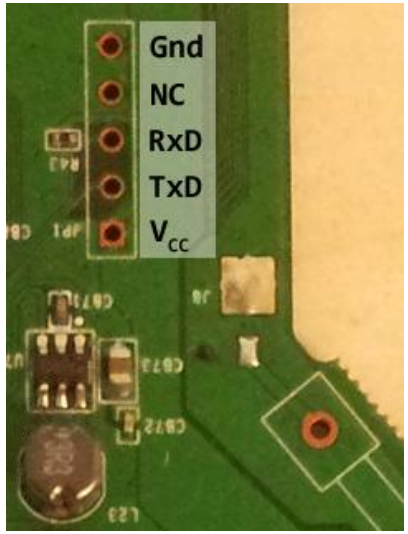
This unit supports Transmit (TX), Receive (RX), Reset (RST), 5 Volts+ (VCC5), 3.3 Volts+ (VCC33), and Ground (GND).

These are easy to find on eBay or Amazon and are extremely useuseful to have around!



Yes, that's a hard drive!

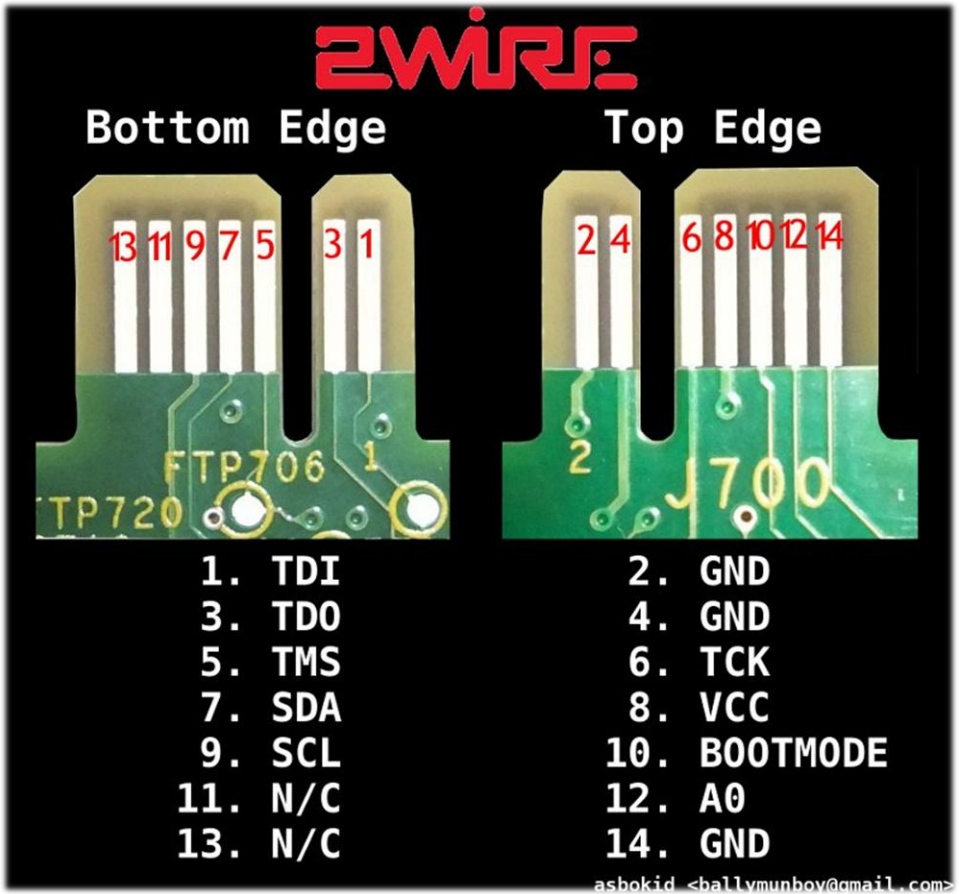
Examples of TTL Serial (Wireless Access Points have it ☺)



Bring On The Attack – Provisioning

- Edge Card Connectors
  - Some manufactures will use a card edge technique to provision
  - This is dependent on where it's being provisioned in a lot of cases.
    - The edge card connector makes it possible for unskilled labor to provision these at a domestic location.
    - Sometimes done if software isn't complete by the time hardware's to manufacture.
    - Sometimes no good reason at all …

Close-up of Edge Card Provisioning Adapter.



- Credit:
  - *Credit for this to "asbokid".*
  - *http://hackingbtbusinesshub.wordpress.com/2012/01/16/discovering-2wire-card-edge-pinout-for-jtag-i2c/*

PEAK SECURITY

Bring On The Attack – Provisioning

- A quick demo of using TTL serial for communication to a Bluetooth Board or Hard drive.
  - Live Demo Goes Here ☺

Debugging Ports

- Remember that chip inventory we were talking about?
- Let's look at an example, which type of debugging port do you see?
  - JTAG
  - BDM
  - ISP
  - SPI
  - I2C
  - CAN
  - NAND flash

Debugging Ports

- JTAG Port Identification
  - What's JTAG:
    - JTAG is the "Joint Test Action Group" standard for circuit level debugging.
  - What can JTAG do:
    - Manipulate individual pins on components
    - Change component state
    - Alter flash memory
    - Has access to many debug utilities

Debugging Ports

- JTAG allows control of all devices on the JTAG bus
  - Usually seen attached to the SPI bus
  - Pic
- JTAG and access to Flash
  - JTAG controls individual chip lines on devices.
  - If there is a flash chip attached to a chip with JTAG (like a NAND Flash) chip, you CAN get the contents of that chip
  - There's tools that make it easy
    - http://www.topjtag.com/flash-programmer/
- JTAG the easy way
  - TopJTAG Probe
    - http://www.topjtag.com/probe/
    - Pic
  - What can we do with it?

Debugging Ports

- Get out that Voltmeter! We're going to identify the TSC (Test Clock) and TMS (Test Mode Select)
  - They're both attached to all chips with JTAG on them and do not get buffered in most cases
  - Look for public data sheets which should allow you to easily identify the TDO and TDI lines
  - Probe the TDI/TDO pin on each device, and run through the provisioning connector to see where you get a BEEEEEEPPPPPP!

Debugging Ports

- 2-wire Unit Demo identifying SPI or I2C provisioning and JTAG connector.

Debugging Ports

- Other Debugging Interfaces
  - Almost all the debugging interfaces we've shown are still in use
  - BDM is extremely common in Freescale based devices
    - Like Joe Grand's DEF CON badges
    - Also show up in SONNET gear
      - Sonnet controller
  - ISP
    - People love to pick on ISP
    - ATmel used it on all it's 8-bit Micro AVR line (Arduino too)
    - Demo
  - Tons more debug interfaces that we don't have time to cover ☺

Debugging Ports

- A note on Side Channel Attacks
  - Are any of the components in the device susceptible to the side-channel attacks we've mentioned?
    - Power consumption analysis
    - Protocol weaknesses
    - Poorly initialized random number generator
    - Timing Analysis
    - Differential Fault Analysis
    - Data Remnant (hat tip to Adam Laurie's work on the AT91SAM7XC)

Debugging Ports

- Beating the Hardware Provisioning Interface Drum with Joe Grand.
- In another talk here:
  - Introduced/will Introduce a device called JTAGulator, makes it much easier to locate JTAG ports
  - Joe has advocated securing JTAG ports for years
  - People should listen to this guy
  - Full props to him on all his hardware work

Software Attacks

- Software is usually the weakest link
  - “If civil engineers built bridges like software engineers build programs, they’d all fall down” – Unknown
  - If you’re going after crypto, always go after the implementation, not the cryptography itself.
- Software can be defeated
  - The software has ultimate control over the hardware
  - Software can be analyzed to determine the exact purpose and functions of the hardware you’re analyzing
  - Our best opportunity for compromise is where software and hardware meet
  - Most software is test under “ideal” conditions
  - The designer can never truly test their own design

Software Attacks

- An example of state dependency for operation
  - This device was used to meter the consumption of a commodity.
  - The device used infrared to communicate with the metering authority's data collection device.
  - By analyzing the protocol, it became evident that certain commands were being issued post authentication, while others weren't.
  - The structure was fairly simple (packetized bytes with a counter, type of packet, flags, and the payload).
  - One of the flags was set only post authentication.
  - By changing the packet type to something different and setting the "authenticate bit" , the security challenge response was defeated!
  - How silly is that?!?!
  - This is a common occurrence in embedded devices. Why?
    - No one bothers to look
    - Software is often rushed

Software Attacks

- Locating the software the controls a device by physical analysis:
  - Examine the traces around the components of the device
  - Are any of the components connected in a way that directly indicates what processor it is (ARM, PIC, Atmega, etc.), such as a direct connection to small form factor flash memory (like an SOIC8)
  - Do the chip numbers indicate that some the components are SPI flash (Dead giveaway)
  - Are some of the components socketed?
    - Careful. Some devices will drop the contents of SRAM if the chip is unsocketed

PEAK SECURITY

Software Attacks

- What if there are no externally visible methods for storing software or configuration data?
  - Some devices hold all the needed software on the main processing component itself
  - This means that there is a way to get the software through a debug connector
  - Some devices have a mask ROM boot loader that allows upload of program code over serial
    - Terrible for security

Software Attacks

- What is a fuse?
  - A fuse is a piece of the hardware that is burnt in order to prevent download of the program code from the chip
- What if I find a “fuse” on the device?
- There are ways around it, as usual
  - The most common is voltage glitching. This involves timing a drop in voltage precisely such that the chip believes it reads a 0 instead of a 1
  - Another way to get around lockouts and fuses is to “erase” the device, but power it off immediately after the command is issued (milliseconds)
    - This can reveal the contents of the flash memory if done correctly
    - You can do this with an Arduino in a lot of cases with a little custom software
  - Partial powering of the device
    - Some devices will draw power from several pins (like large QFP or BGA components)
    - By severing or regulating some of those power leads, you can cause chips to misbehave
    - Sometimes it’s possible to confuse the chip, but make it responsive to the debugging interface
  - Bottom line, chips are fragile to power fluctuations, but there have been huge improvements over the past few years.

Software Attacks

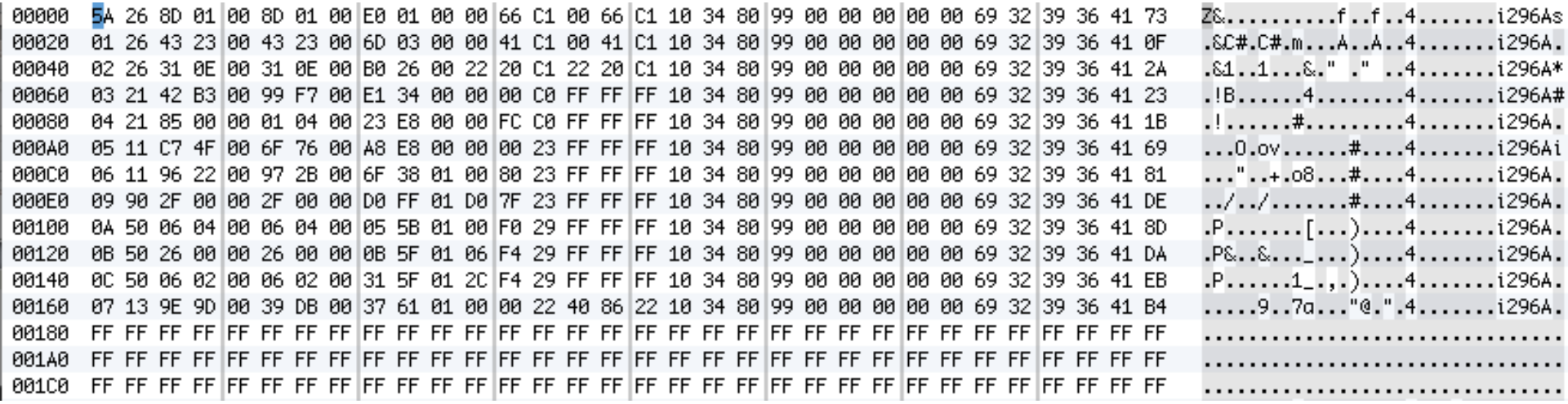
- Can the device's software be updated?
  - Check the website
  - If there's a software update, download it and see if you can deconstruct it (lots of info in here)
  - If you find a binary package (like a CAB) you may have found the firmware
  - Some vendors like to "encrypt" their updates. Most of the time this a VERY weak system like XOR or XOR +/-1
  - In most cases unless it's a "secure" device, more companies are concerned with IP leak than the security of the device

Software Attack

- Identifying the firmware
- Each device has it’s own way to structure software
- Every Device is Unique, just like all the others ☺!

Software Attack
ARM:

- Loader tables



```
00000  5A 26 8D 01 00 8D 01 00 E0 01 00 00 66 C1 00 66 C1 10 34 80 99 00 00 00 00 00 69 32 39 36 41 73  Z&..........f..f..4.......i296As
00020  01 26 43 23 00 43 23 00 6D 03 00 00 41 C1 00 41 C1 10 34 80 99 00 00 00 00 00 69 32 39 36 41 0F  .&C#.C#.m...A..A..4.......i296A.
00040  02 26 31 0E 00 31 0E 00 B0 26 00 22 20 C1 22 20 C1 10 34 80 99 00 00 00 00 00 69 32 39 36 41 2A  .&1..1...&." ." ..4.......i296A*
00060  03 21 42 B3 00 99 F7 00 E1 34 00 00 00 C0 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 23  .!B......4........4.......i296A#
00080  04 21 85 00 00 01 04 00 23 E8 00 00 FC C0 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 1B  .!......#........4.......i296A.
000A0  05 11 C7 4F 00 6F 76 00 A8 E8 00 00 00 23 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 69  ...O.ov......#....4.......i296Ai
000C0  06 11 96 22 00 97 2B 00 6F 38 01 00 80 23 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 81  ..."..+.o8...#....4.......i296A.
000E0  09 90 2F 00 00 2F 00 00 D0 FF 01 D0 7F 23 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 DE  ../../.......#....4.......i296A.
00100  0A 50 06 04 00 06 04 00 05 5B 01 00 F0 29 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 8D  .P.......[...)....4.......i296A.
00120  0B 50 26 00 00 26 00 00 0B 5F 01 06 F4 29 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 DA  .P&..&..._...)....4.......i296A.
00140  0C 50 06 02 00 06 02 00 31 5F 01 2C F4 29 FF FF FF 10 34 80 99 00 00 00 00 00 69 32 39 36 41 EB  .P......1_.,.)....4.......i296A.
00160  07 13 9E 9D 00 39 DB 00 37 61 01 00 00 22 40 86 22 10 34 80 99 00 00 00 00 00 69 32 39 36 41 B4  .....9..7a..."@."4.......i296A.
00180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................................
001A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................................
001C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................................
```

- These represent a 20 byte loader row that tells the program on the MPU how to load an overlay into memory on the chip since the main chip doesn't have enough memory to hold it all at once. (Hint, the addresses are three bytes long!)

Software Attack

- MSP430
  - Notice the vector table at the 0x0000 addresses
  - seg000:0000FFE0 .short 0F852h
  - seg000:0000FFE2 .short 0F852h
  - seg000:0000FFE4 .short 0F852h
  - seg000:0000FFE6 .short 0F852h
  - seg000:0000FFE8 .short 0F852h
  - seg000:0000FFEA .short 0F852h
  - seg000:0000FFEC .short 0F852h
  - seg000:0000FFEE .short 0F852h
  - seg000:0000FFF0 .short 0F852h
  - seg000:0000FFF2 .short 0F852h
  - seg000:0000FFF4 .short 0F956h
  - seg000:0000FFF6 .short 0F852h
  - seg000:0000FFF8 .short 0F852h
  - seg000:0000FFFA .short 0F852h
  - seg000:0000FFFC .short 0F852h
  - seg000:0000FFFE .short 0F800h

Software Attack

- AVR
  - Again, a vector table to define what happens to the component upon events like restart, timer interrupt, etc.
  - ROM:0000 TWSI__0: ; CODE XREF: TWSI_j
  - ROM:0000 jmp __RESET ; External Pin, Power-on Reset, Brown-out Reset and Watchdog Reset
  - ROM:0000 ; End of function TWSI__0
  - ROM:0000
  - ROM:0002 ; ---
  - ROM:0002 .org 2
  - ROM:0002 jmp TWSI_ ; 2-wire Serial Interface
  - ROM:0004 ; ---
  - ROM:0004 .org 4
  - ROM:0004 jmp TWSI_ ; 2-wire Serial Interface
  - ROM:0006 ; ---
  - ROM:0006 .org 6
  - ROM:0006 jmp TWSI_ ; 2-wire Serial Interface
  - ROM:0008 ; ---
  - ROM:0008 .org 8
  - ROM:0008 jmp TWSI_ ; 2-wire Serial Interface
  - ROM:000A ; ---
  - ROM:000A .org 0xA
  - ROM:000A jmp TWSI_ ; 2-wire Serial Interface

Software Attack

- Xilinx FPGA
  - Xilinx bit stream

```
----------Bitstream Header----------
Design Name: Unknown
Part Name: 2vp50
Date: Unknown
Time: Unknown
-------------End Header-------------
----------Begin Bitstream-----------
ffffffff - Dummy Word
aa995566 - Sync Word
30008001 - CMD
00000007 - CMD: Reset CRC
30016001 - FLR
000000e1
30012001 - COR
00043fe5
3001c001 - IDCODE
0129e093
3000c001 - MASK Type I
00000000
30008001 - CMD
00000009 - CMD: Switch CCLK Frequency
30002001 - FAR
00000000 - MJA: 0 MNA: 0
30008001 - CMD
00000001 - CMD: Write Configuration
30004000 - FDRI Type II
500910ea - Length: 594154 words (2629 frames)
00800000 - BA: 0 MJA: 0 MNA: 0 Word: 0 (GCLK)
00000140 - BA: 0 MJA: 0 MNA: 0 Word: 1 (GCLK)
```

  - Much Thanks to Casey Morford and is work on dynamic reconfiguration of Xilinix chips

Software Attack

- Determining other chips structures
  - Most have different version of embedded compilers and RTOS (Real Time Operating Systems)
  - Each one has its own “markers” in the binary image
  - The Code Sorcery (now Mentor Graphics) compilers and linkers address overlaying code in a particular way
  - Keil Software’s compilers also have a number of markers in their loader and vector handling code
  - Many devs will start with manufacture provided code samples (we do), which makes it easier to identify
  - Look at the order of addresses, segment numbers, and flags

Software Attack

- Each device has its own mechanism for updates
  - It may be possible to “man in the middle” a firmware update and obtain the decrypted firmware.
  - Some firmware may be decrypted on-component.
  - If you find a significant area of a file that is a sequence of repeating characters (say 4 bytes), there’s a good bet that the “encryption” your dealing with is simple XOR and that you’re looking at the key.
  - In the same sense, if you find a sizable area where the values are incrementing or decrementing, you may have found a XOR +n/-n key.
    - If you work the math backwards, you can figure out the key, the repeat interval, and the progression.
    - Determine the unit of advancement, the value at the observed locations, and the offset in those units and work it backwards to get the key to start with.
    - Starting Key = (((known offset) – (start offset)) / (size of key)) * (Key at known offset)
    - Roughly. The multiplication is confined to the key size. Or you can use a “for” loop. ☺

Software Attack

- Each component type (family) has its own machine language
- *ARM is not AVR is not Z80 is not x86 is not PPC.*
- *The operational codes (OP codes) are different.*
- *The Peripheral regions (Input / Output regions) are different.*
  - *Some families may have the same general regions, but specific additions to deal with additional I/O needs (like an ARM chip on a hard disk controller).*
- *Recognizing these will greatly assist you in analyzing the security of the device in cases where you do not know the type of component you are dealing with that runs the software.*

Software Attack

- Taking Apart the Software
  - Weapon of choice
    - IDA Pro is most reverse engineer's weapon of choice.
    - It will not always unless you know something about what you're dealing with.
    - Knowing the chip's layout and I/O peripheral regions is important.
    - IDA Pro will ask you about these if it knows the processor in question.
    - Demo
  - You may need to write software to extract portions of the firmware before you run it through IDA Pro.
    - We're dealing with bare metal here folks!
    - Some hardware developers are quite inventive.
      - They will write their own loader/linker format and no-one else in the world (or their right mind) uses it.
    - This is where your own reverse engineering skill level comes in to play.

Software Attack

- Methodical analysis of the hardware by debug interface
  - Use a JTAG (or other interface) to enumerate interconnections on devices that are otherwise impossible (BGA/SMT with through holes under the components).
  - JTAG will allow the switching of individual IO lines on and off
  - Build a document about the relevant interconnections
    - What's relevant?
    - Connections from a main processor unit (MPU) to a flash chip.
    - Connections form the MPU to a solid state or switching relay
    - Connections from the MPU to any sub-processors.
    - Connections from the sub-processors to their controlled devices.
    - Bus interconnections for SPI/I2C/CAN/Single-Wire devices.

Software Attack

- Identifying major modes of operation
  - Usually are the most basic
    - Powered On
    - Standby
    - Powered Off

Software Attack

- Identify major functional software groups
  - Hardware I/O routines
  - Main program logic
  - User interaction routines.
  - Configuration routines.
  - Encryption routines.

Software Attack

- Enumerate the device's major functional states
  - Each type of device will be a bit different
  - An encryption device might have:
    - Not Imprinted / Blank
    - Provisioned by corporate owner, ready for personalization.
    - Personal Key generated, device in "secure mode", device Locked, not operating.
    - Personal Key generated, device in "secure mode", device Unlocked and operating.
    - Device Tampered
    - Device needs provisioning due to Certificate expiration.
    - Device in Debug Mode
  - A simple device like a "fan on, fan off" device might have:
    - Get packet, turn fan on or off

Theory

- Theory of Operation
  - Now that we have a great deal of information we can analyze it.
    - We know how the device operates.
    - We know how it gets input and output.
    - We have a rough idea of how it will respond to various stimuli.
  - Put this into a usable form.
    - State transition and flow diagrams are helpful to visualize possible points of weakness.
  - A summary listing of software and hardware functions.
    - List of software methods and description beside each.
    - OpenSSL
    - List of hardware “transaction” on a SPI Flash ROM.
    - Microchip Diagram

Theory

- Test the Theory of Operation
  - Setup tests to validate your theory
    - Test for confirmation of function.
    - Be creative, but exhaustive!
    - If you see something unusual, that's a good place to start looking for a weakness.

Theory

- These tests can be direct tests of tampering with information as it flows in an out of the device
  - Altering I/O states on devices
  - Changing the contents of flash device or EEPROM.
  - Causing power state changes (brown out) to reset fuses or cause some portion of the device to reset, while leaving the other running (i.e. a MPU reset and a encryption sub-processor that's already authenticated).
  - Feeding bad network information into the device
  - Using the TTL serial port to halt the processor and cause timeout states on I/O peripheral devices.
  - Changing device programming to do abnormal behaviors under some set of conditions.
  - Attempt to extract keying material from the device.
  - So many variants

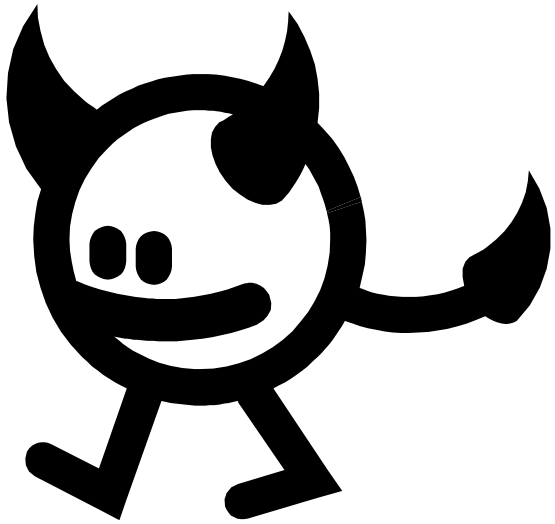
Theory

- Record the Results
  - Taking apart hardware and software is complex on a good day.
  - Detailed recording of details as they are noticed will make final analysis much easier.
    - Record:
      - The parts on the device
      - The interconnection of major components
      - The methods that the software uses to use these components
      - A general theory of operation about the device
    - Issue the Final Report
      - If this was done to evaluate a device professionally.
        - » Put in an executive summary
        - » A list of tests preformed
        - » A list of vulnerabilities located
        - » Appendices with all the other relevant detail.
      - If not, go have a beer! You've earned it!

Methodology Overview

- Gather open source information about the device from a meta-level.
- Open the device up and catalog it.
- Gather information about the components.
- Examine component circuit net interconnections
- Extract the software and reverse engineer, if possible.
- Form theory of operation.
- Test theory of operation by asserting conditions and observing results.
- Document and Report findings!
- Beer.

Questions?



PEAK SECURITY