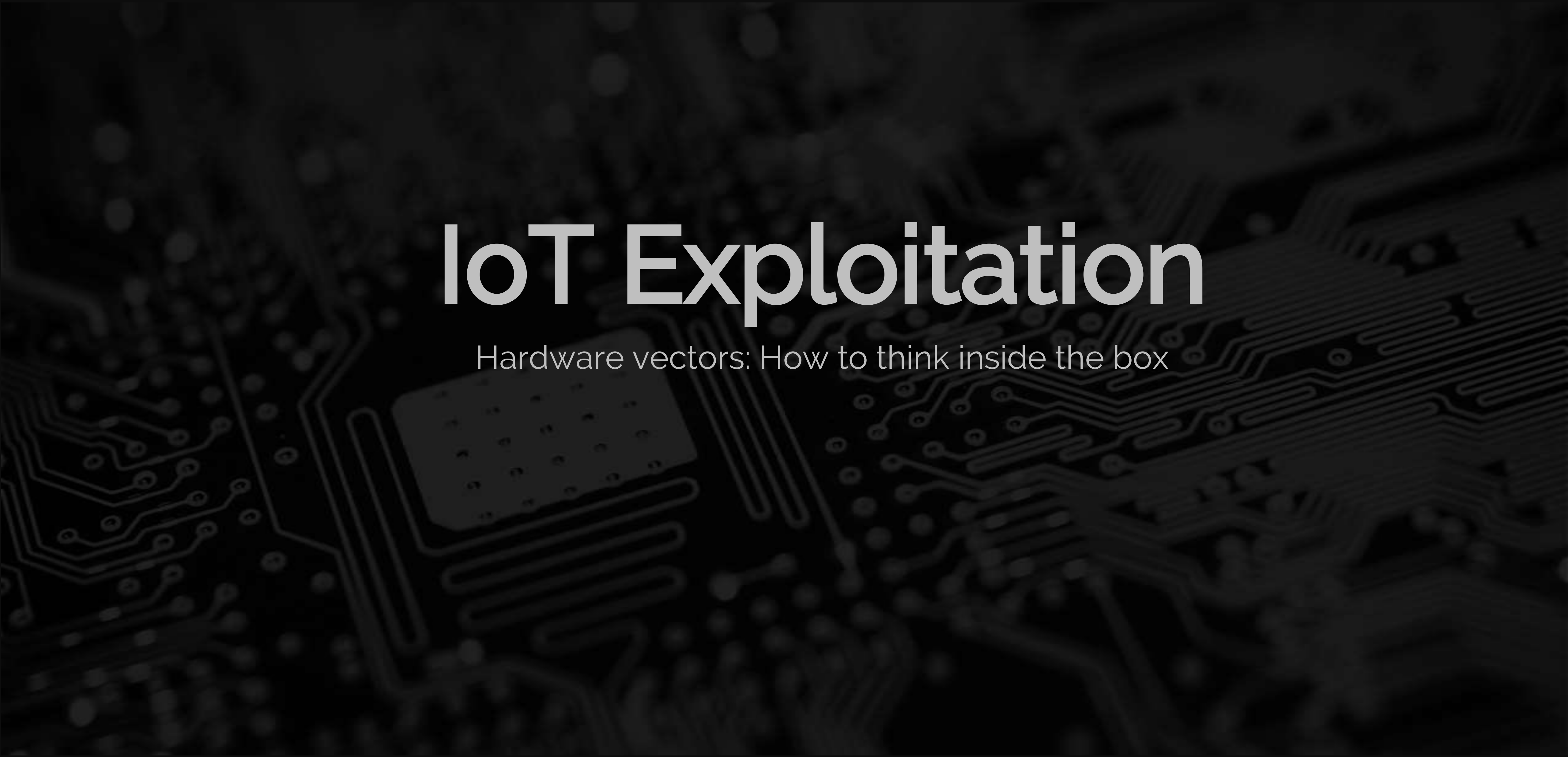


IoT Exploitation

Hardware vectors: How to think inside the box

What’s with IoT?

The internet of things (IoT) is the network of physical devices, vehicles, buildings and other items—embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data.

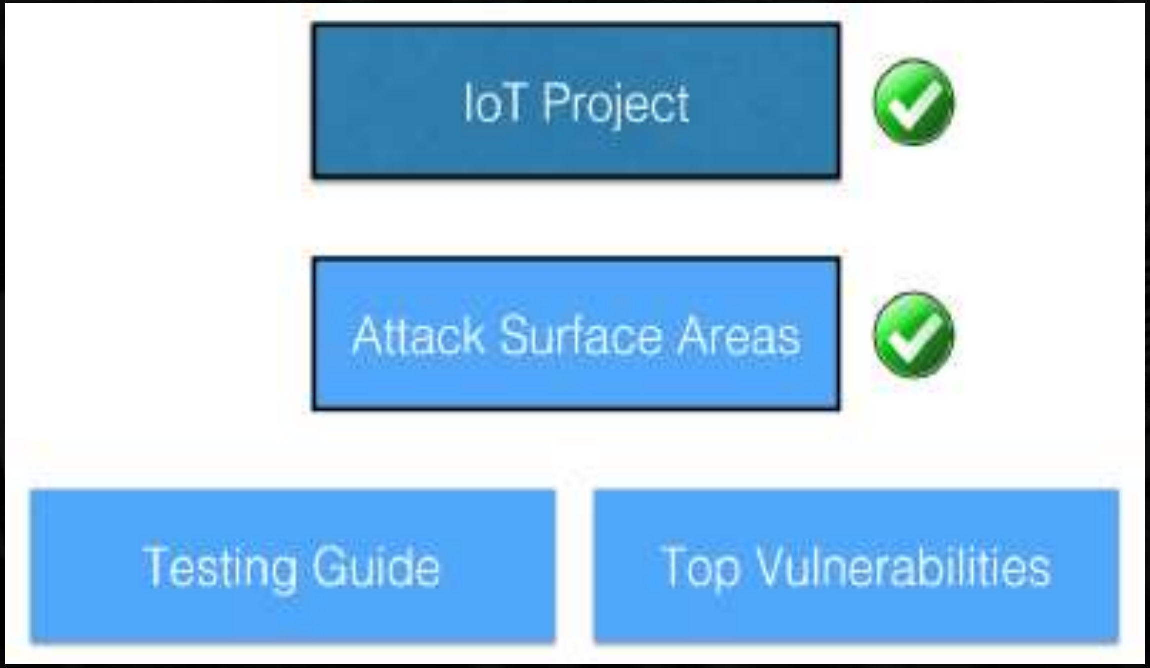
(Wikipedia)

Devices like..

- Networking devices(routers, firewalls, IDS)
- Set-top boxes
- Medical devices(Health monitors)
- Mobile phones.
- Home security systems
- Vehicles, in-flight entertainment.
- Thermostats, metering systems, consumer electronics Displays

So what do we attack?

a.k.a. attack surfaces



IoT Project ✓

Attack Surface Areas ✓

Testing Guide

Top Vulnerabilities

[OWASP IoT project]

Attack Surfaces

- Ecosystem
- Device Memory
- Physical interfaces
- Firmware
- Networks/Communications
- Mobile Applications/ Web / Cloud.

etc

OWASP IoT project

Let’s talk hardware..

Starting point..

- Scapy(Python) + some fuzzing + TP-LINK == Crash!
- No way to understand crash from outside the box.

Starting point..

How about thinking inside the box?

WR740N - v4.23

Debug Interfaces!

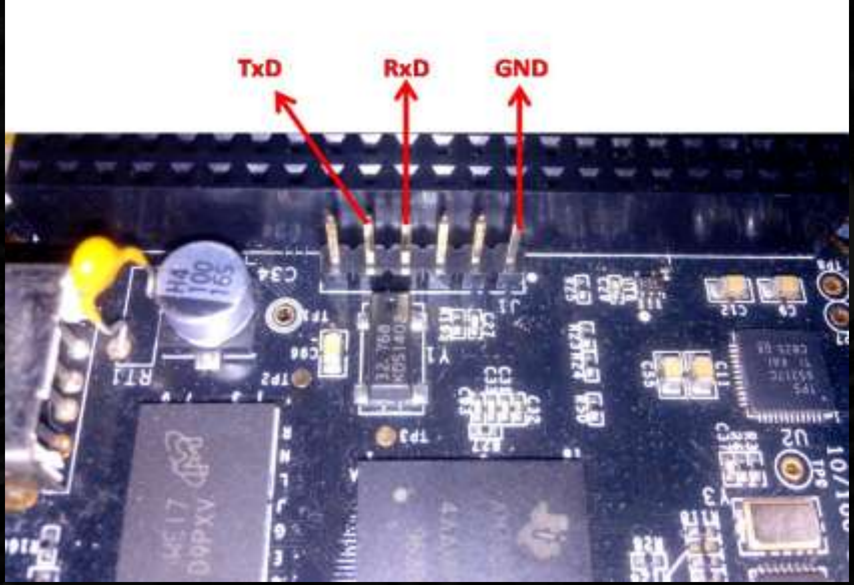
- So wait.. there are debug interfaces that are serial.
- UART, I2C, SPI, JTAG.
- We are talking hardware but this isn’t engineering major, this stuff is simple.
- Yes, we are talking about complete physical access but for vulnerability research

(U)ART thou mad?

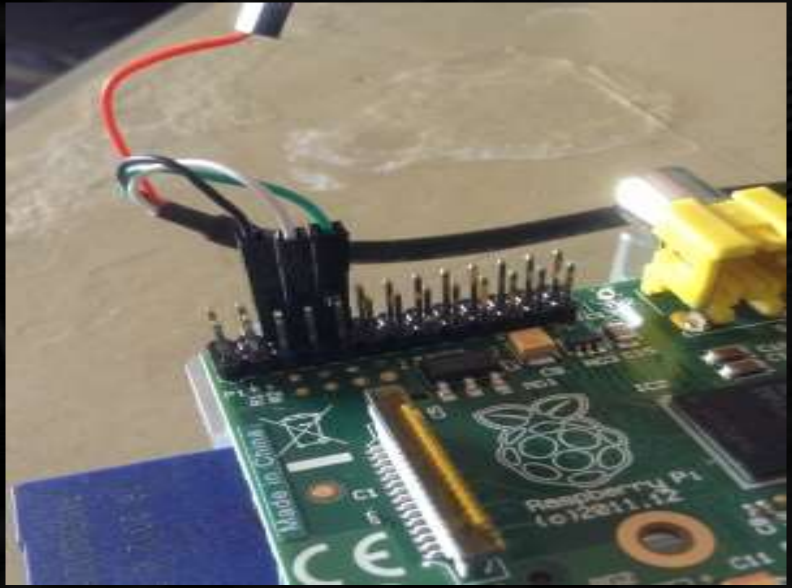
- Universal Asynchronous Receiver/Transmitter (UART) interfaces are by far the most common interface for diagnostic and debug output from embedded devices.
- Simple & inexpensive.
- Available on almost every embedded devices.
- Gets you a console or shell access to the device(sometimes root shell without authentication).

Identifying UART

- Simple UART has little as three or four connecting wires. (Transmission, Receiving, Ground, Power).



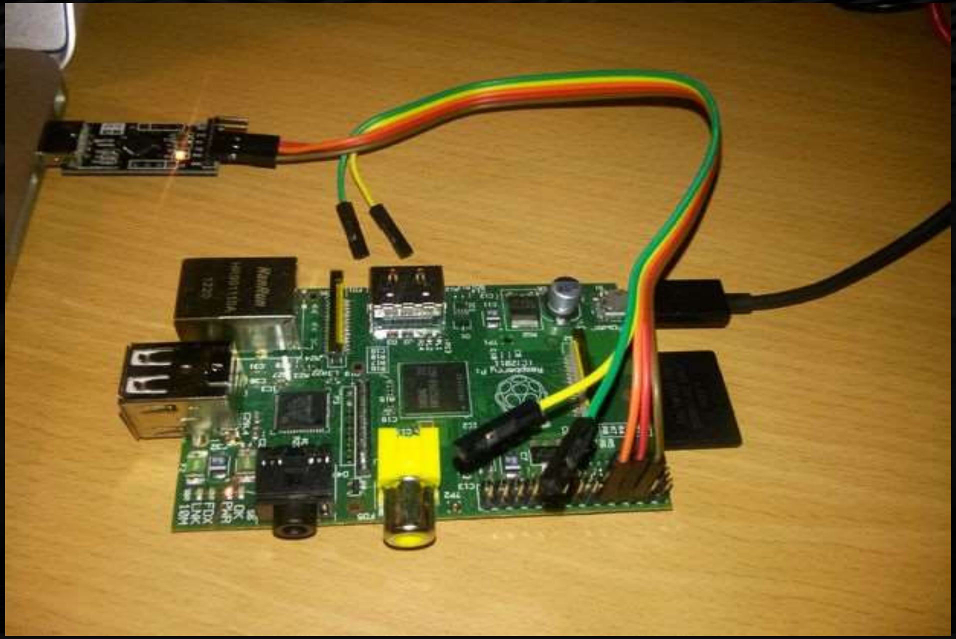
Beagle board UART



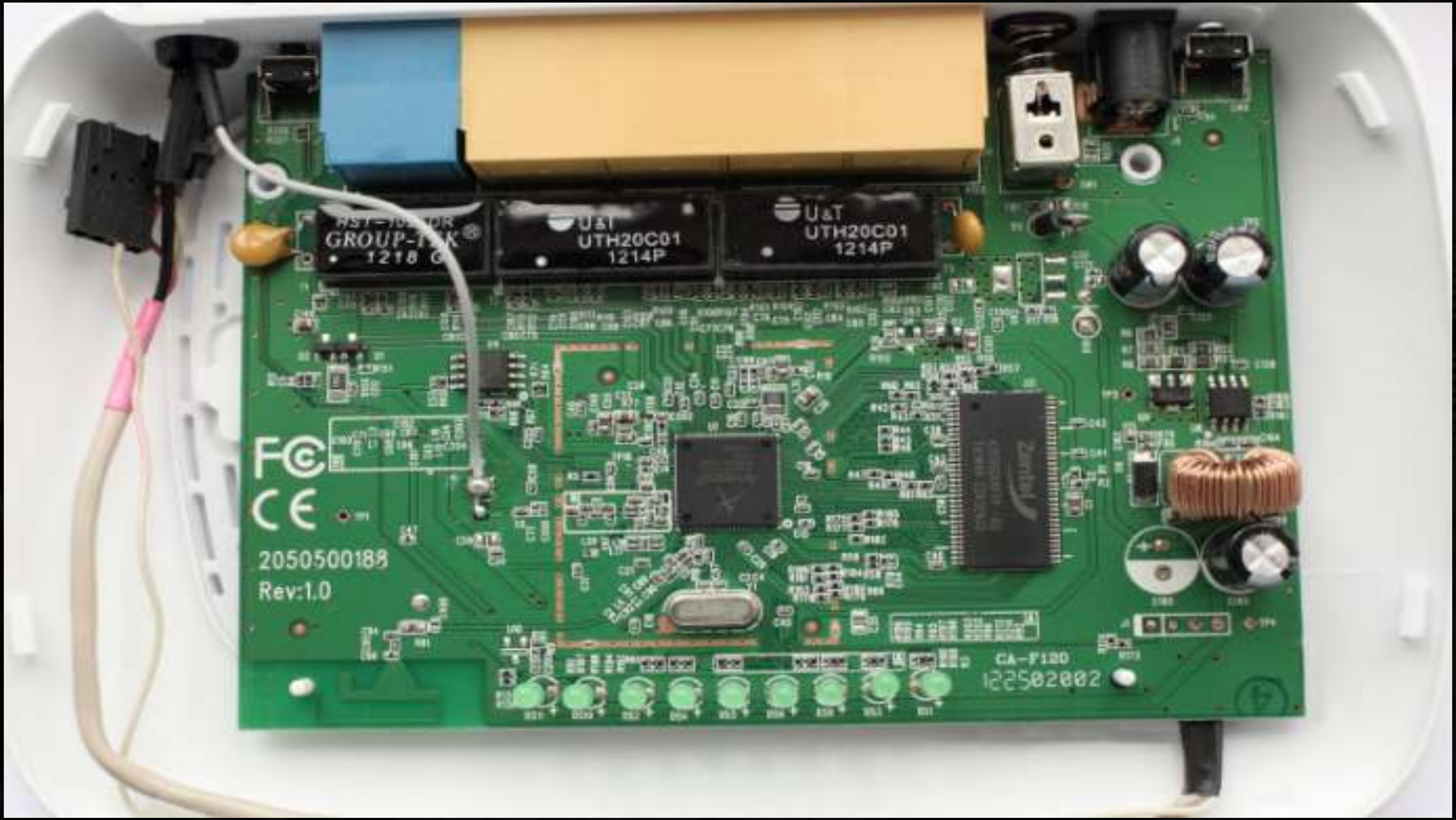
Raspberry 2 UART

Identifying UART

- Use a USB to TTL cable to connect to the device from computer.
- Use software like minicom to get to the console.
- You'll get debug information, access to boot loader, shell(maybe root access, filesystems).

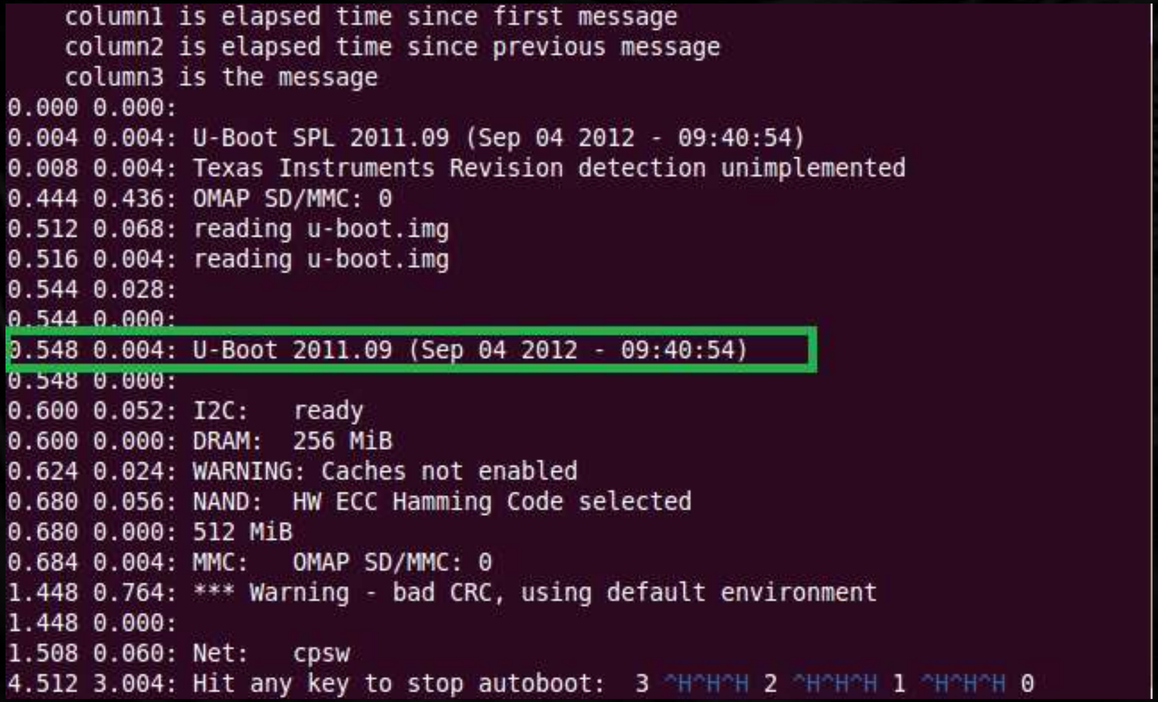


Can you find the UART?



After connecting on UART..

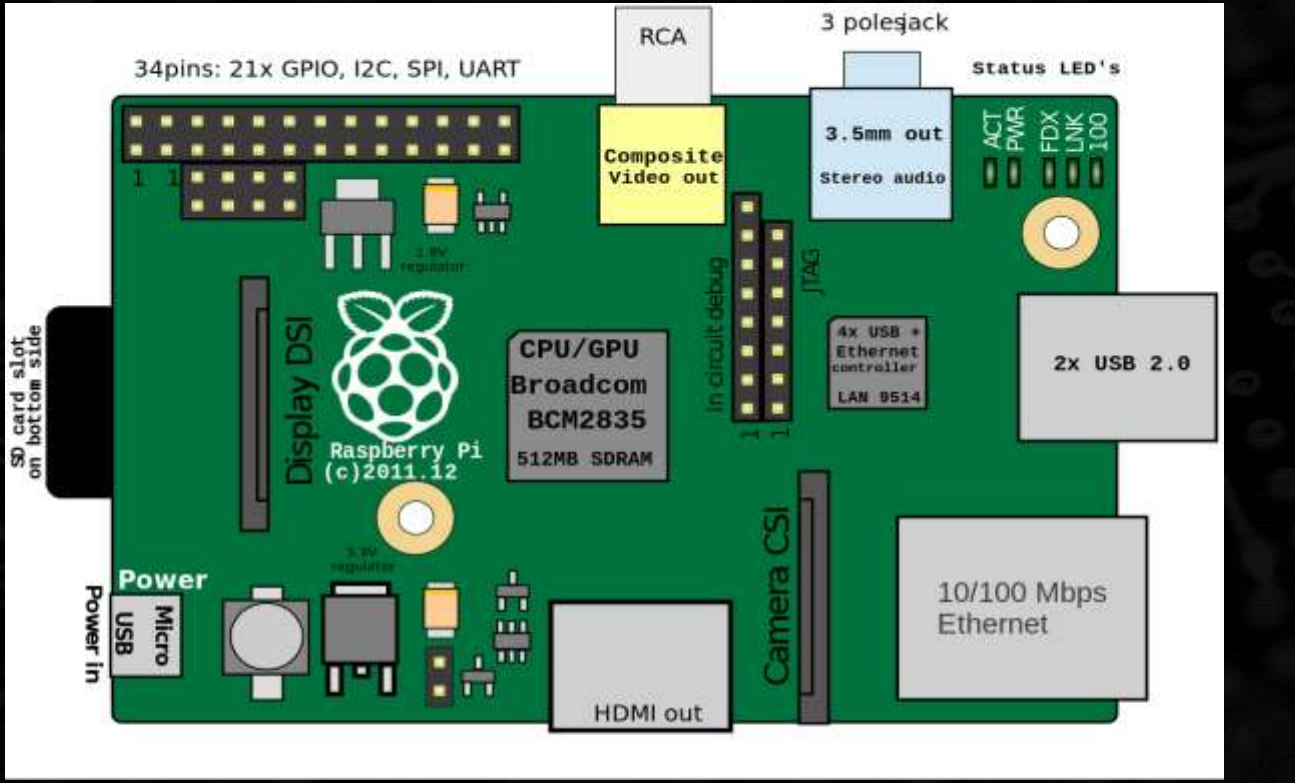
- Using minicom like software you get access to debug information, bootloader, bootloader menu, shell etc.
- **Crash + debug information + binary → Software exploitation.**

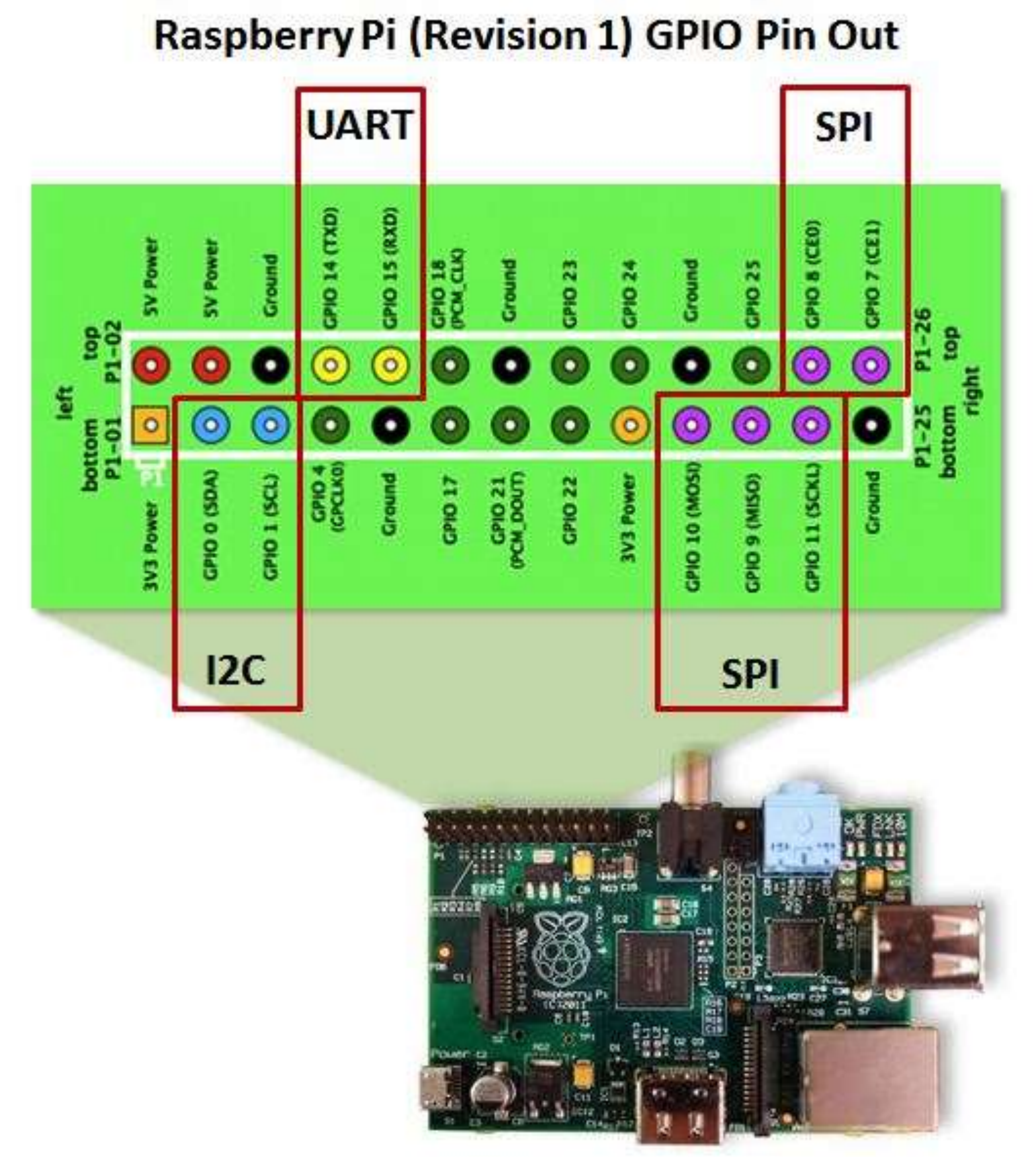


```
    column1 is elapsed time since first message
    column2 is elapsed time since previous message
    column3 is the message
0.000 0.000:
0.004 0.004: U-Boot SPL 2011.09 (Sep 04 2012 - 09:40:54)
0.008 0.004: Texas Instruments Revision detection unimplemented
0.444 0.436: OMAP SD/MMC: 0
0.512 0.068: reading u-boot.img
0.516 0.004: reading u-boot.img
0.544 0.028:
0.544 0.000:
0.548 0.004: U-Boot 2011.09 (Sep 04 2012 - 09:40:54)
0.548 0.000:
0.600 0.052: I2C:   ready
0.600 0.000: DRAM:  256 MiB
0.624 0.024: WARNING: Caches not enabled
0.680 0.056: NAND:  HW ECC Hamming Code selected
0.680 0.000: 512 MiB
0.684 0.004: MMC:   OMAP SD/MMC: 0
1.448 0.764: *** Warning - bad CRC, using default environment
1.448 0.000:
1.508 0.060: Net:   cpsw
4.512 3.004: Hit any key to stop autoboot:  3 ^H^H^H 2 ^H^H^H 1 ^H^H^H 0
```

Finding UART Pins

Find the manufacturer schematics





Finding UART Pins

- Use multimeter to find voltage (Helps you avoid frying your boards)

  3.3v data
  5v is power
  0v is ground

- Connect oscilloscope to find square wave(indicates digital signals)

- Better, use logical analyzer for detecting digital signals.

- Use things like JTAGulator

- If you found your pins.. start memory analysis, dumping, reversing etc

I2C, SPI

- UART is for humans to interact with the PCB components.
- How about components talking to each other??
- I2C, SPI enables IC’s to talk to each other(like a LAN connection between components).
- Extremely simple(Sometimes just one pin).
- Possibility of spying.

I2C, SPI

- I2C, SPI are available on virtually every embedded device.
- HDMI, VGA etc.
- SDcards use SPI for data transfer.
- **I2C is what laptops use to update the remaining battery status.**

- GeoHot’s first Apple jailbreak involved I2C spying.
- Charlie Miller research on MACBook power management involved I2C spying(presented at BlackHat 2012)

JTAG

- Very powerful debug interface to debug a chip from a separate computer.
- UART relies on dedicated code execution on the device(a shell, an interactive boot loader etc) but JTAG helps to debug the device at any point. (You can even get a GDB server for debugging and can get full memory dumps).
- Software debugging is just part of JTAG specification and is not strictly standardized so it depends on the vendor.
- **Single stepping, breakpoints, power resets, watch-points, register viewing, and boundary scanning are part of JTAG implementations.**

Identifying JTAG

- JTAG standard defines five standard pins for communication.
  - TDO: Test Data Out
  - TDI: Test Data In
  - TMS: Test Mode Select
  - TCK: Test Clock
  - TRST: Test Reset

Identifying JTAG

- JTAG standard defines five standard pins for communication.
- The way communication happens over JTAG pins is vendor specific.(Which serial protocol to use etc)
- Implementation differences between each device can take a lot of time and effort to figure out.
- Identifying JTAG and it's implementations in the wild is a lot of effort.

JTAG Adapters

- JTAG adapters are like translators and they understand hundreds or thousands of JTAG implementations.
- Segger J-Link is relatively inexpensive, supports long list of devices, USB powered, acts as GDB server.

JTAG Adapters

- OpenOCD – just software, supports vast number of implementations but it has to be used along with some hardware like Bus Pirate that comes with no software.
- Using OCD is heavy lifting, you need to be aware of pins, configurations and wiring etc.

Logic Analyzers

- These devices just show you what is happening on a pin.
- If there is data being transmitted on a pin it shows you the square wave of that data and even attempts to decode it for you using a number of different filters.

Finding JTAG Pins

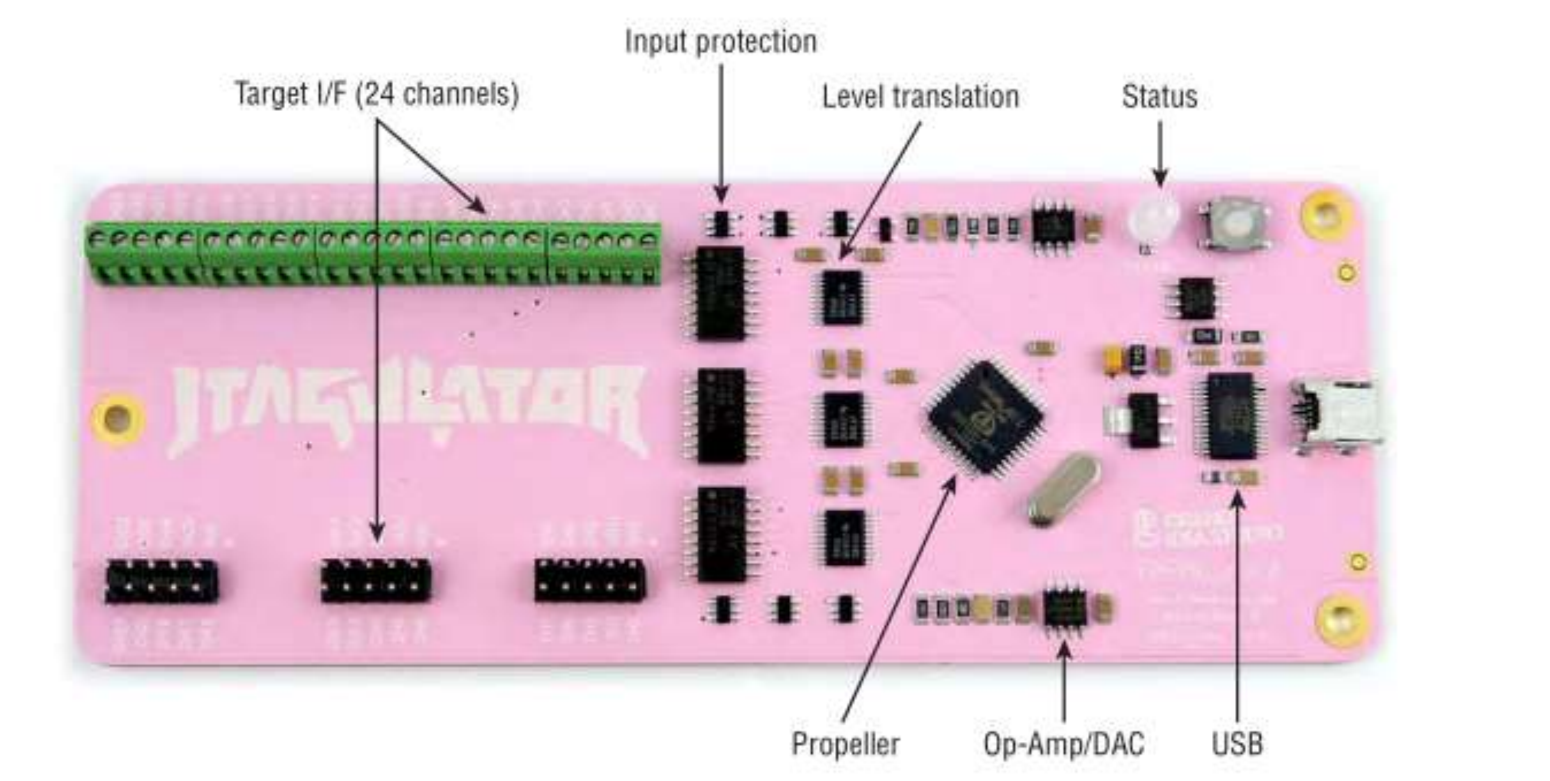
- Finding JTAG pins and configurations is time consuming and tedious.
- It all depends on the manufacturer and if they try to hide JTAG adapters, it gets even worse.
- Good news: We care about only 5 pins
  - TDO: Test Data Out
  - TDI: Test Data In
  - TMS: Test Mode Select
  - TCK: Test Clock
  - TRST: Test Reset

(TRST is just for reset so we just need 4 pins)

JTAGulator

- Open source hardware device developed by Joe Grand.

(You can find everything about schematics, firmware online)

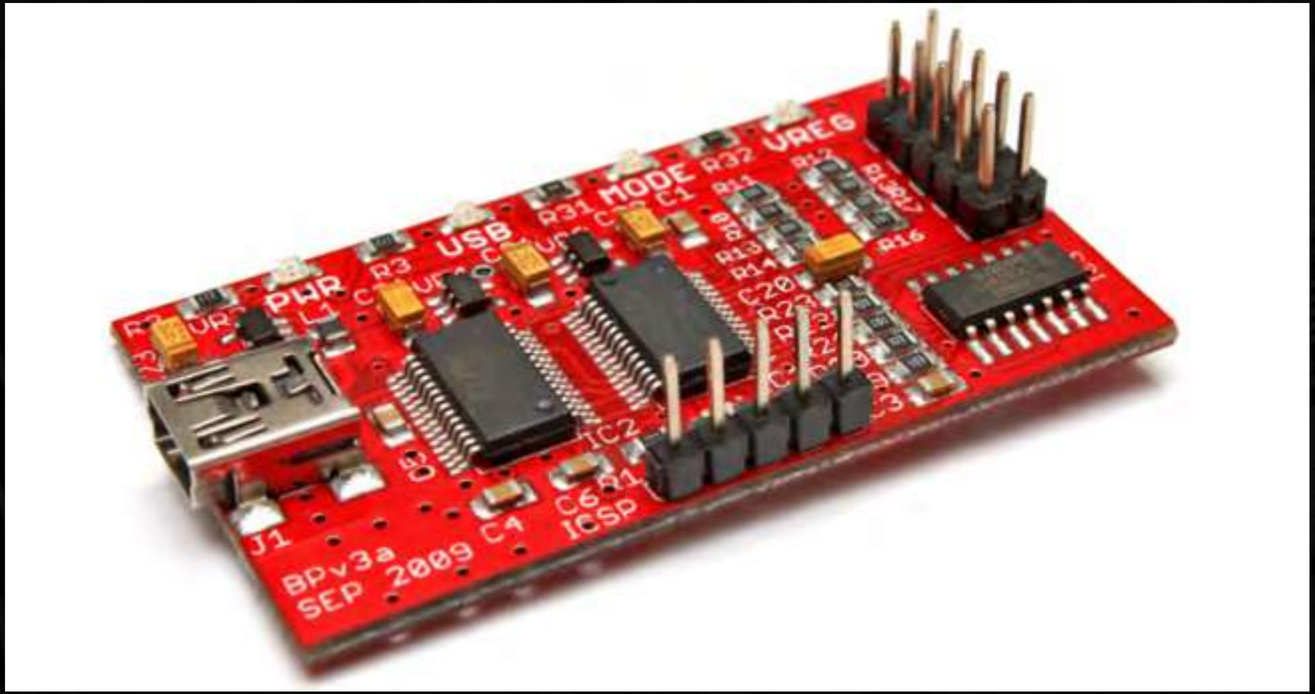


JTAGulator

- Brute-forces JTAG pinouts blindly.
- As simple as connecting all the suspicious pins to ***JTAGulator*** and use Putty or Minicom.
- http://www.grandideastudio.com/portfolio/jtagulator

Talking simple serial protocols

- Bus pirate has support for a variety of protocols.



Talking simple serial protocols

- Bus pirate has support for a variety of protocols.
- Similar to JTAGulator. Connect with USB and interact using Minicom or Putty.
- Once connected bus pirate makes it easy to interact with device over serial protocols.
- Extensive number of macros and scripts are available from the community.
- You can do everything you do with bus pirate using a USB-TTL cable but Bus pirate makes things easier

Let’s talk software exploitation

Firmware Analysis

- Firmwalker
- Firmware Modification Kit
- Angr binary analysis framework
- Binwalk firmware analysis tool
- Binary Analysis Tool
- Firmadyne
- Flashrom for capturing file systems

Damn Vulnerable Router Firmware for practice

Firmware Analysis

- Firmware analysis tools are not effective all the time that’s when it’s time for binary reverse engineering.
- IDA, Hopper, Radare 2.
- IDA FLIRT is useful for firmware analysis.

Architecture

- Architecture: ARM | MIPS | SPARC
- Operating Systems: Linux based OS, RTOS etc.

- Embedded system are built on RISC architectures opposing to CISC like X86.
- Exploitation principals are still the same, just a little different.
- ROP attacks are convoluted but works.
- Exploit mitigations(like NX) exist in ARM.

Architecture

- ARM is RISC so you have lesser instructions and more registers so attacks like ROP can get more convoluted and tedious.
- Hovav Shacham in his paper on ROP explains that it’s possible to build turing complete machines using ROP gadgets. (Especially on X86 due to it’s dense instruction set)

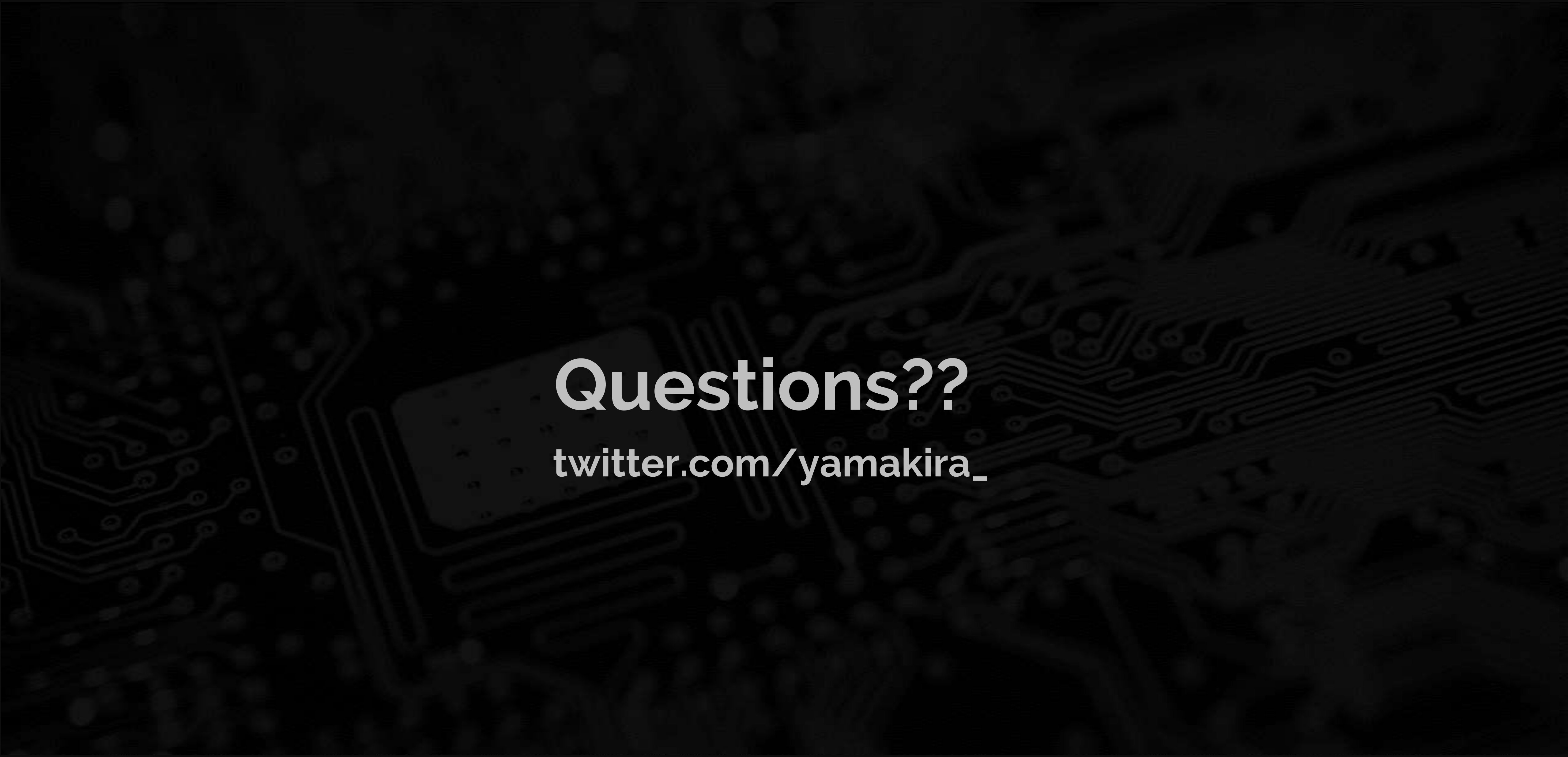
How to attack a million devices?

- SHODAN – search engine for devices.
- It’s trivial to write scripts to scan for IoT devices online.

Getting started

- Use **qemu** to emulate the ARM architecture
    - Try to understand ARM, the edge cases
    - Get comfortable with debugging, disassembly
    - Gera insecure programs, Exploit exercises

- Get some real hardware
    - Raspberry
    - Beagle Board
    - ARMini
    - CuBox
    - Gumstix



Questions??

twitter.com/yamakira_