The Smarts Behind Hacking Dumb Devices

Maddie Stone
@maddiestone

OffensiveCon 2018

Who am I? - Maddie Stone

- Security Engineer on the Android Security Team at Google
- ~5 years experience reversing embedded devices -- lots and lots of microcontrollers
- Creator of IDAPython Embedded Toolkit
- I like to break things, do improv comedy, and travel.

The process & mindset behind hacking non-IoT embedded devices

Target Devices: “Dumb” Embedded Devices

- Microcontroller that senses and/or controls physical world
- No wireless connectivity (WiFi, Bluetooth)
- No operating system (Linux, RTOS, etc)

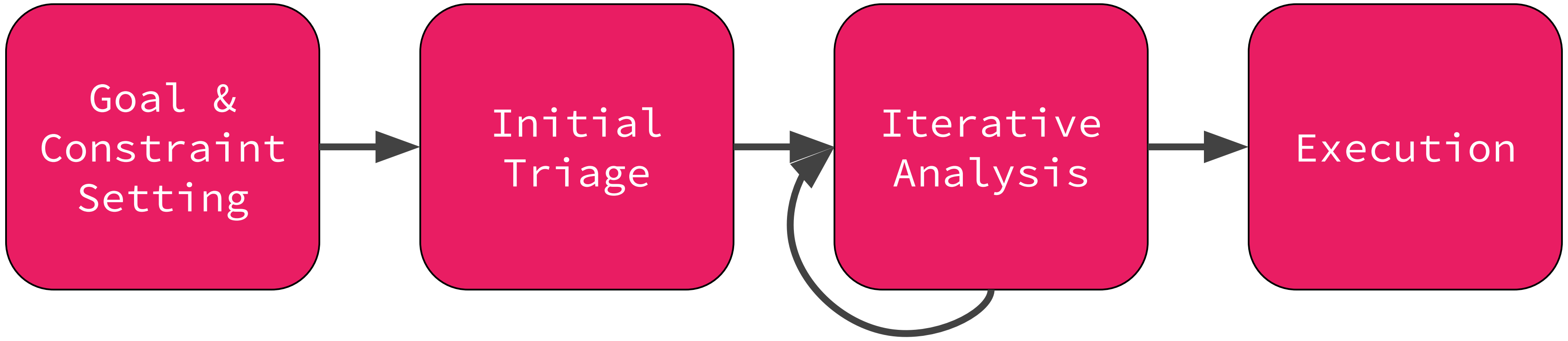
Why these devices?

- They’re everywhere
- Control many critical functions
- Intersection between hardware & software
- Oh so much fun!!

What does it mean to “attack” these devices?

- Cause the device to act in unintended ways
- Examples:
    - Exfil data
    - Modify the control algorithms
    - Add features

The Process


Goal & Constraint Setting
Initial Triage
Iterative Analysis
Execution

Initial Goal & Constraint Setting

- What is the goal?
  - Ex. Destroy it, communicate with it, add extra features
- What do I need to access?
  - Do you need the FW/netlist? Comms traffic? Device in whole system?
- How destructive can I be?
  - Does the device have to work? Can I purchase more? Am I worried about anti-tamper?
- How many resources am I willing to put in?

Initial Triage

- Capture behavior and traffic
- Open source review
  - Datasheets, schematics, programming methods
- Hardware teardown
  - Component identification
  - Hidden interfaces - debugging or comms
  - Access to power
- Obtain/access firmware
  - Firmware extraction: SW methods (BSPs, serial console, patch), internet, JTAG/ICE/debugging, chip reader, SEM
- Instrumentation

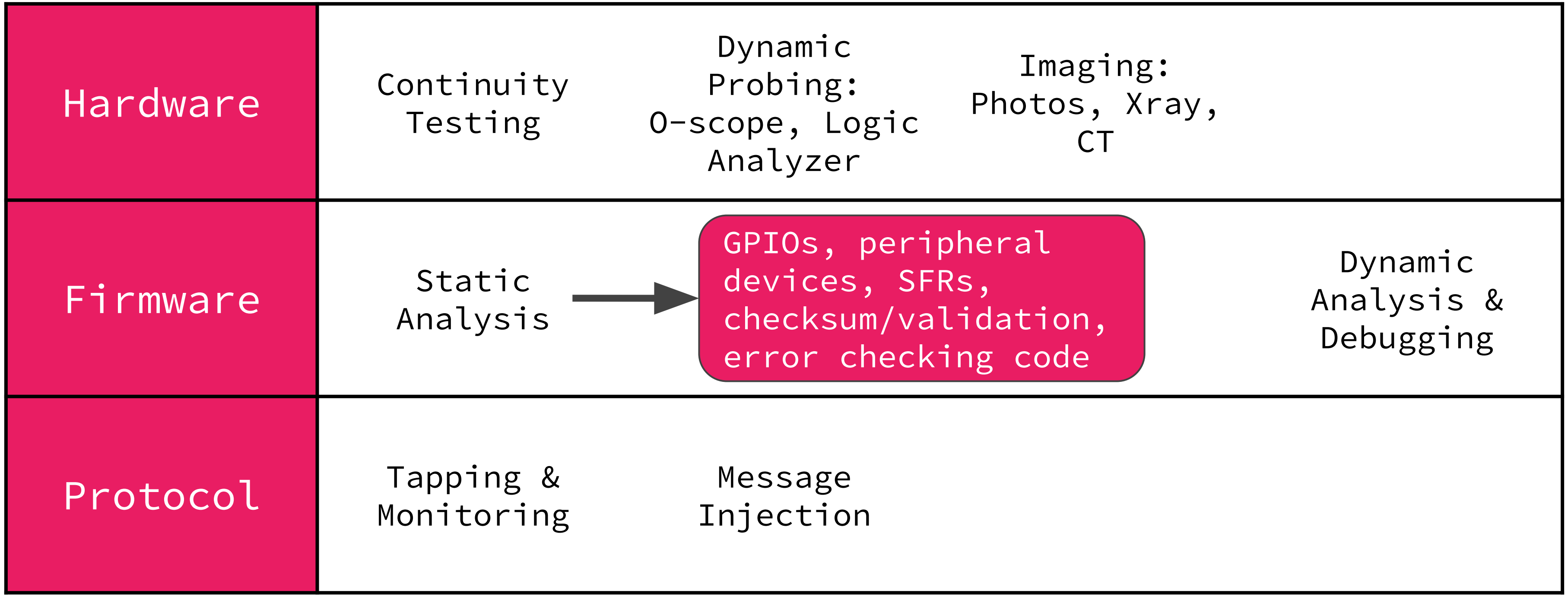
Can I access the firmware?
Is the firmware writable?

Is power accessible?
Are the critical vias/traces accessible?

Identifying HOW to Achieve Attack

- Physical hardware you want to affect or alter
- Control algorithms
- View or modify information the device stores
- Communication buses
- Debugging interfaces

Iterative Analysis



Hardware	Continuity Testing	Dynamic Probing: O-scope, Logic Analyzer	Imaging: Photos, Xray, CT	
Firmware	Static Analysis →	GPIOs, peripheral devices, SFRs, checksum/validation, error checking code		Dynamic Analysis & Debugging
Protocol	Tapping & Monitoring	Message Injection		

Developing Patch/Implant/Exploit

- Hardware
  - Prototyping - space, power, attachment points, heating
  - Replacing components
- Firmware
  - Mitigate checksum
  - Identifying location in firmware
  - Hooking
  - Update/ firmware writing process
  - Writing your patch (C, asm, machine code)
- Protocol
  - Crafting specific communication messages
- Debugging
  - Toggling GPIO

Thank You!

github.com/maddiestone
@maddiestone