Proving the wild jungle jump

Master Systems Network Engineering
University of Amsterdam
Research Project 2 (#48)

Supervisors:
Niek Timmers
Albert Spruyt
Lukasz Chmielewski

Student:
James Gratchoff
james.gratchoff@os3.nl

What is a wild jungle jump?

What is a wild jungle jump?

The effect of corrupting the program counter of the processor in such a way that it points the attacker to a controlled address

Purpose

- Run arbitrary code on a secure device

Why?

- Riscure saw this behaviour happening while attacking systems implementing secure boot

Outline

I. Introduction
II. Scope
III. Research question
IV. Related work
V. Target overview
VI. Approach
VII. Set up
VIII. Assumptions
IX. Results
X. Conclusions and future work

Introduction

Research performed at Riscure in Delft

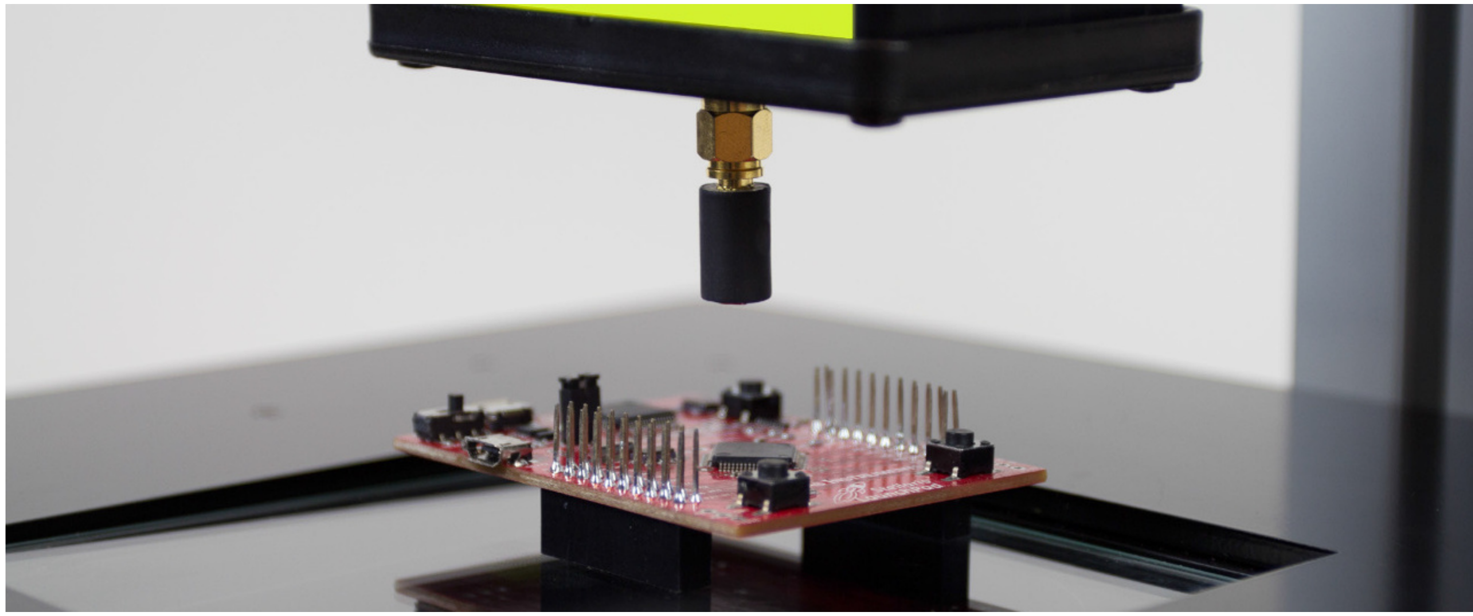
- Specialised in side channel analysis and fault injection

FI is a successful and cheap way to attack systems:

- Cryptographic systems (AES, RSA)
- Smartcards

Fault injection

- Clock
- Temperature
- Optical (Light)
- Electromagnetic radiation
- Power



Electromagnetic FI

Scope

riscure

Power fault injection

- Insert an impulse or drop of power in the system to change the behaviour of the processor without interupting its process

Targeting one kind of architecture

- ARM

ARM

Research questions

What is the feasibility of a wild jungle jump?

- How can the PC be corrupted?
- What is the likelihood of a glitch corrupting the PC?
- What are the repercussions of a wild jungle jump?

Related work

- No research perfomed around PC corruption with FI
- 2012
  Barenghi et al: Fault injection attacks on cryptographic devices?
  - Memory instructions are the only instructions prone to power FI.
- 2014
  Thessalonikefs: EMFI on a Wandboard
  - Skip instructions

Target

Wandboard

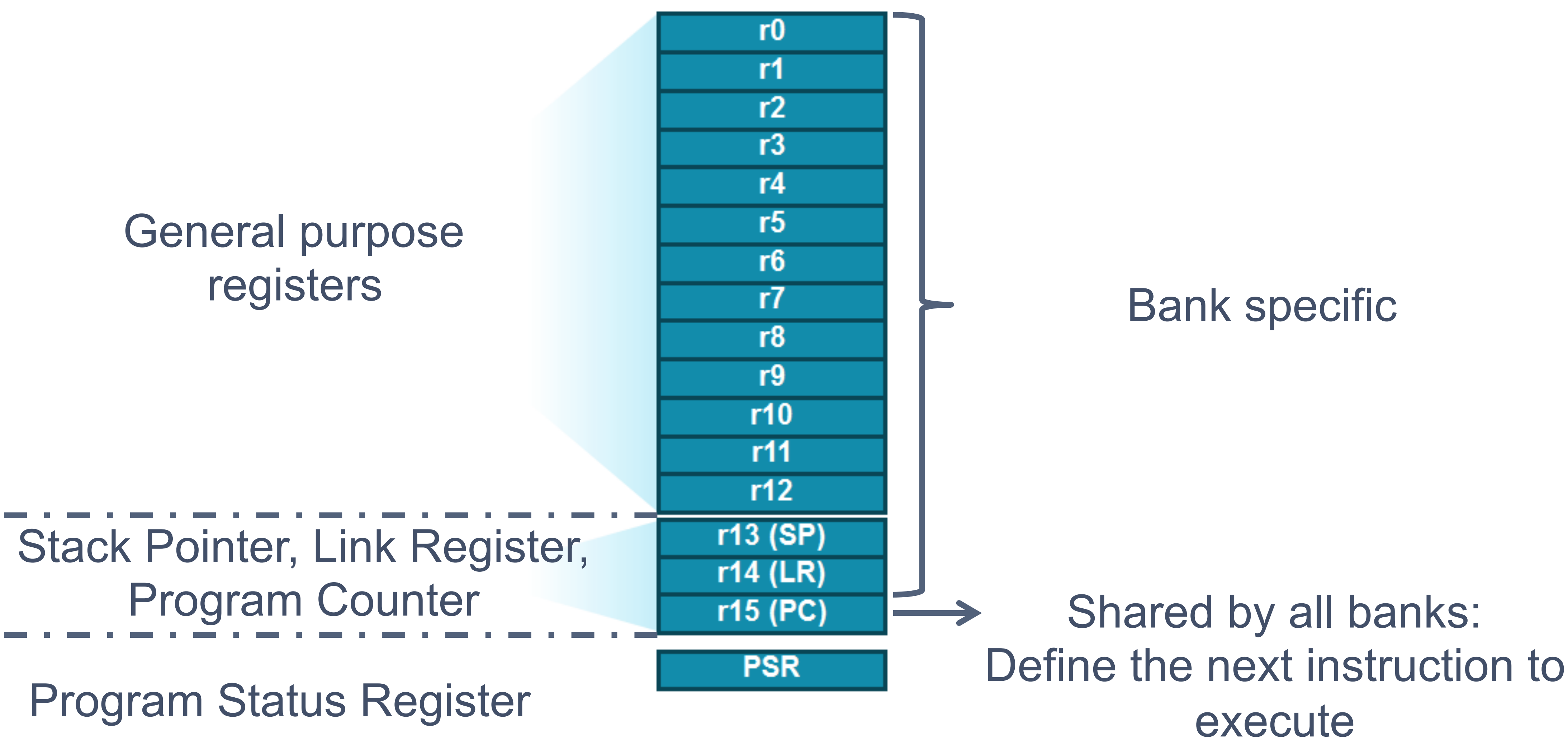
- Freescale IMX6 platform with an ARM Cortex A9 processor
    - RISC infrastructure
    - 792 MHz (1,26 ns/cycle)
    - 32-bit

This processor is also present in:

Cortex A9 overview

Register architecture

- 37 registers separated in 7 different banks
  - User bank:



Approach

- Hands on tool to perform FI
- Assumptions about how to corrupt the PC
- Code implementation (assembly)
- Power FI test with wide parameters
- Result analysis
- Narrow parameters ⟶ raise percentage of success

Set up

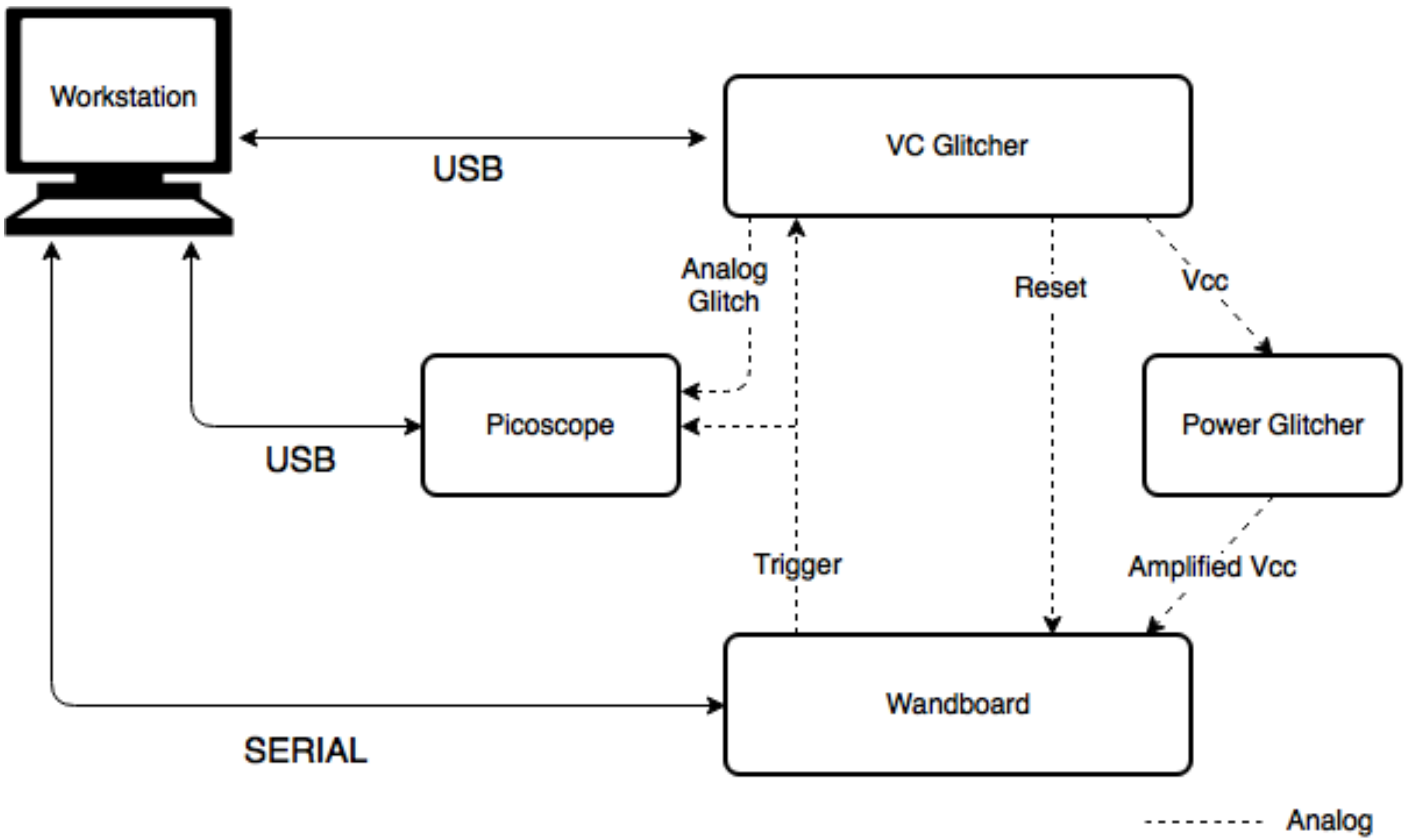
Set of hardware provided by Riscure

- VC glitcher: Glitch generator
- Glitch Amplifier
- Picoscope 5203: Digital oscilloscope for monitoring
- Wandboard

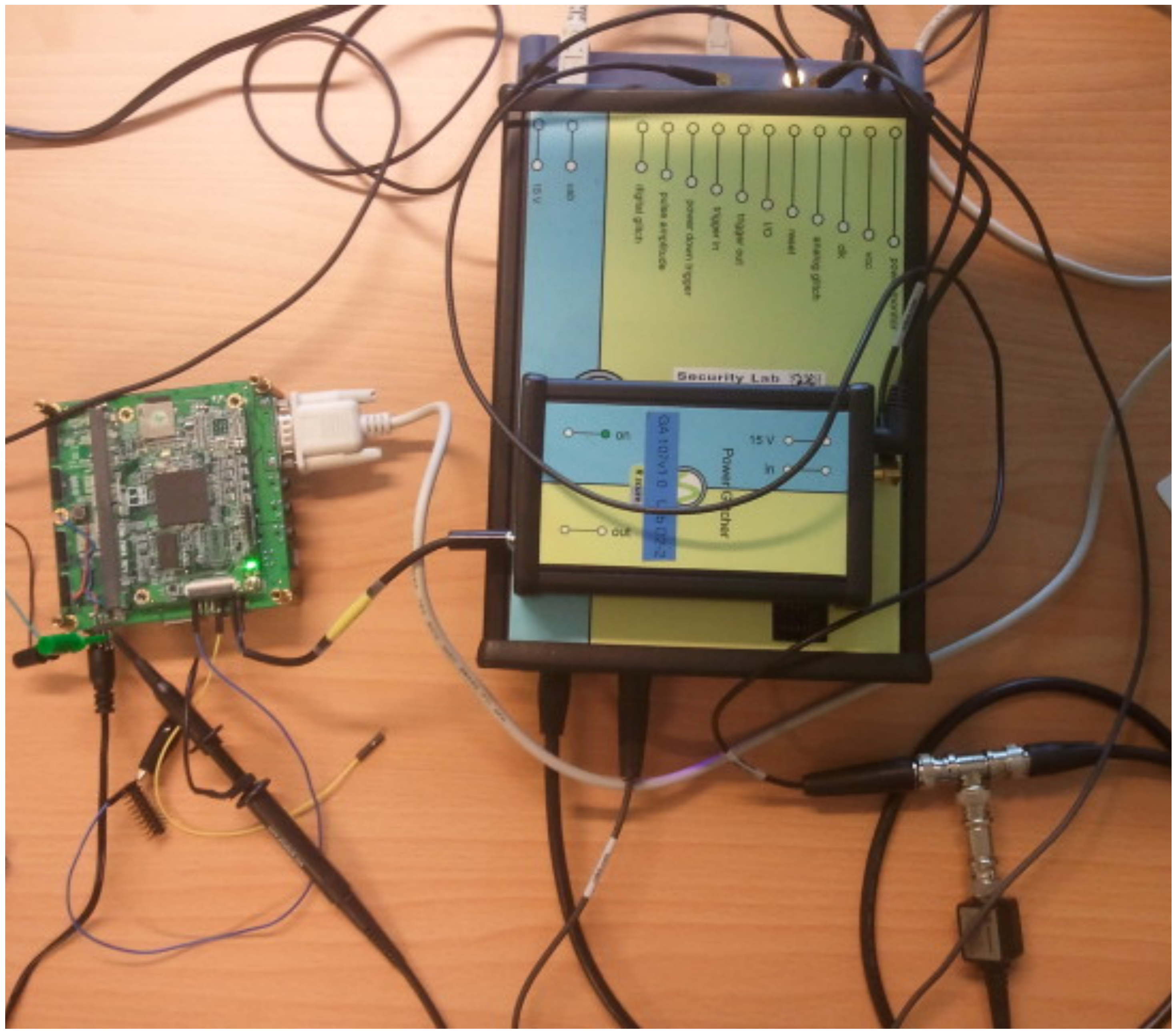
Set of software

- Picoscope 6.0: Oscilloscope software
- Inspector FI 4.8.3: Define FI parameters
- FI GraphIt 1.0: Result analysis tool

Set up (2)


Workstation
USB
VC Glitcher
Analog
Glitch
Reset
Vcc
Picoscope
USB
Power Glitcher
Trigger
Amplified Vcc
Wandboard
SERIAL
Analog

Set up (3)



Assumptions

To corrupt the PC a glitch could:

1. Skip one or more instructions
2. Corrupt an instruction

Code goals:

- Prove the feasibility of these assumptions

Results- Instruction skip characterization

Target: Set of instructions incrementing a counter

Goal: Characterization of such attack vector

Results:

- Counter returned lower values than loop length
- Difference in number of instructions skipped observed

Success Rate: 45%

Results- Instruction skip (2)

Target: End and start of consecutive functions

Goal: Glue functions together

- Value of the registers set in the first reused in the second functions

Results: Success

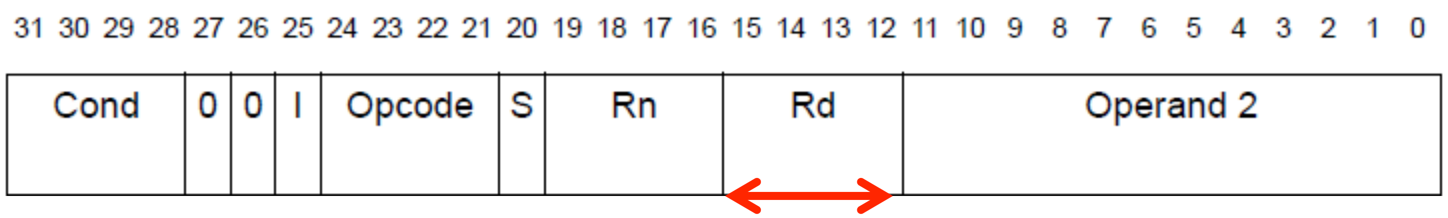
Success Rate: 0,01%

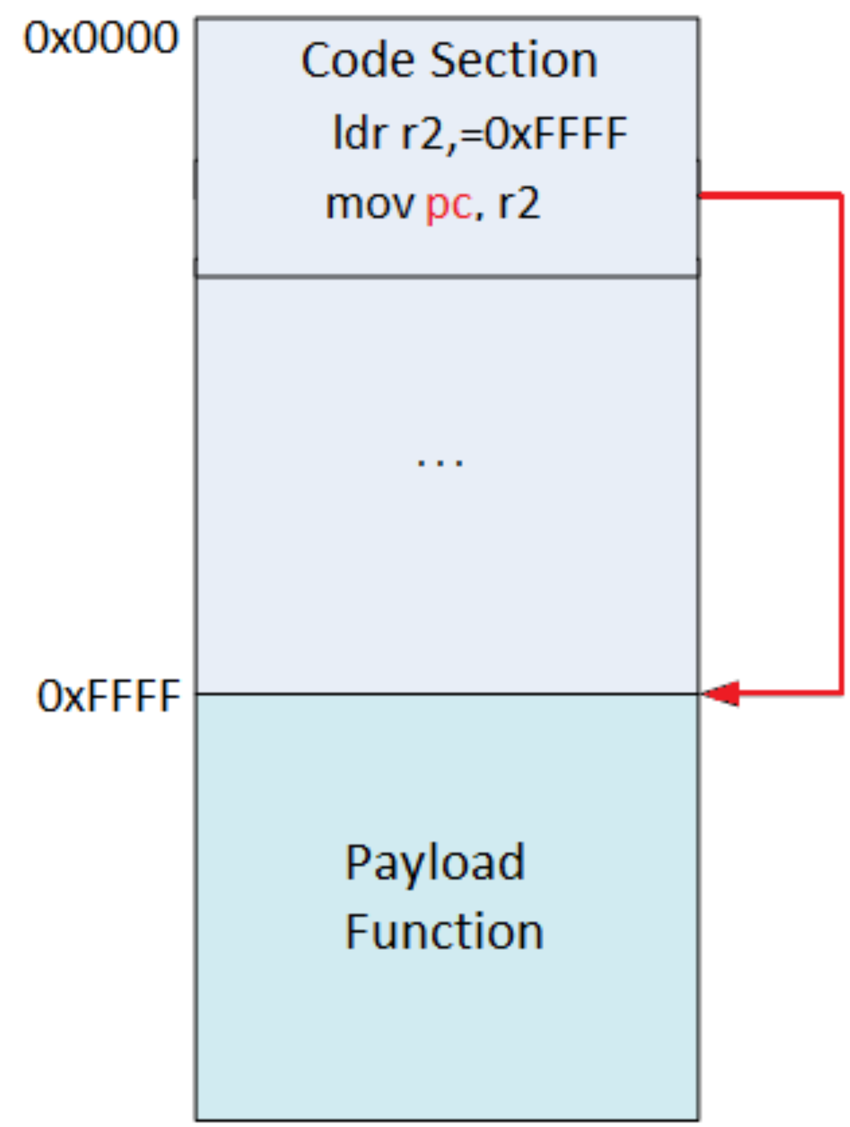
Remark: Exploitable code could not be found in open source implementation investigated

Results – Instruction corruption characterization (MOV)

Target: MOV instruction i.e. MOV PC, R2

Goal: Flip the destination register (12-15 bit) to 1





Result: Success!

Attack vector: Arbitrary code execution

Success Rate: 0,16%

Remark: Instruction often present but not controllable by the attacker

Results – Instruction corruption (LDR)

Target: Load instruction

Goal: Flip the destination register to PC

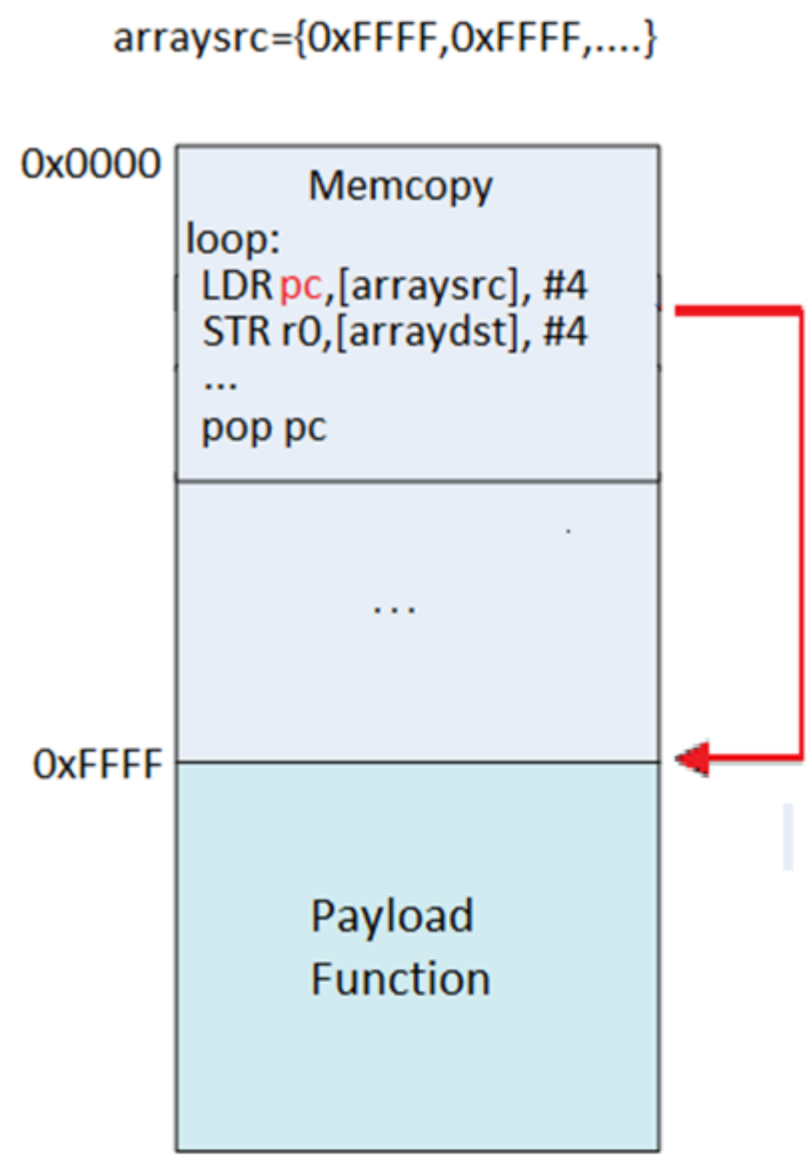
Attack vector: Memcopy

Result: Success!

- Code execution by copying an address pointing to the start of the attacker's code

Success Rate: 3,4 %

Remark: Present in U-boot



Conclusions

Wild jungle jump is feasible with power FI

- By skipping instruction
- Corrupting a MOV or LDR instruction

Attack is possible in existing implementation

- Memcopy

Downsides

- Dependencies to reproduce the attack:
  - compiler version or chain
  - Need of deep understanding of assembly code
- Finding the right FI parameters can be a tedious job

Future work

- Prove the possibility of a wild jungle jump in other architectures (x86, AMD)
- Find other open source real life example of where a wild jungle jump can occur
- Perform a wild jungle jump using other FI techniques

References:

EMFI picture
https://www.riscure.com/
Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Barenghi, Breveglieri, Koren, Naccache. 2012
ARM logo:
https://commons.wikimedia.org/wiki/File:ARM_logo.svg
Wandboard:
http://www.wandboard.org/
I-phone 4S, Ipad2, Samsung GS III:
https://wikipedia.org
ARM instruction decoding:
+http://emucode.blogspot.nl/2010/09/decoding-arm-instruction-set.html
Electro Magnetic Fault Injection Characterization. George Thessalonikefs 2014

Thank you for your attention

Questions?