avatar^2

Marius Muench

34c3 - December 29, 2017

EURECOM
Sophia Antipolis

Contents

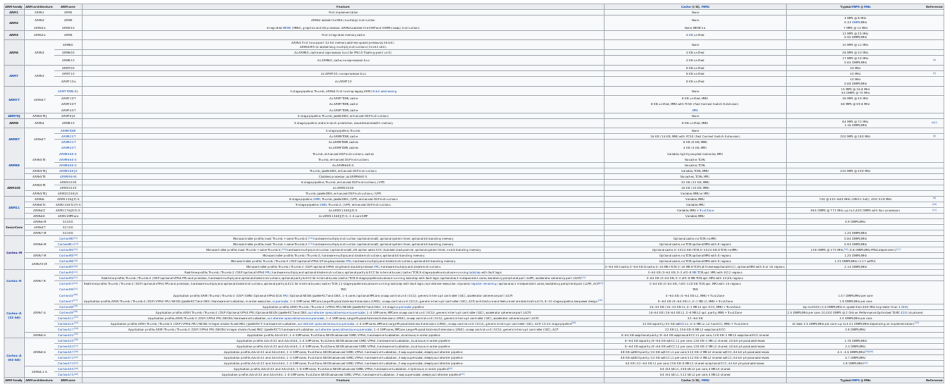
Binary Firmware Analysis

Motivation

- Amount of embedded devices steadily increasing
- Misconfigurations, bugs, and vulnerabilities are common
- A lot of reported vulnerabilities are ”low-hanging fruits”
- Discovery of more complex bugs benefits from sophisticated tooling

Major Challenges

- Variety of platforms
  - Memory layout
  - Peripherals
- Often no OS-level abstractions
- Many devices use monolithic firmware
- Hardware interactions are embedded in firmware code
  - Memory Mapped I/O
  - Interrupts
- Variety of architectures



https://en.wikipedia.org/wiki/List_of_ARM_microarchitectures#Designed_by_ARM

Further Challenges

- Instrumentation
- Emulation
- Fault detection
- Interrupt handling
- Microarchitecture dependent instructions

Tooling Landscape

Binary Analysis Tools for Firmware

- A lot of binary analysis tools for desktop software
- Way less for embedded devices software
  - Especially when considering open source tools
- Often, challenges for embedded devices exceed capabilities of static analysis tools
  - Assumuption about environment may not hold true
  - Difficult to infer peripheral behaviour and interrupts

FiE

- Based on KLEE
- Targets MSP430 firmware
- Symbolic Execution
- Uses explicit analysis, memory and interrupt specifications

Davidson, Drew, et al. "FIE on Firmware: Finding Vulnerabilities in Embedded Systems Using Symbolic Execution." USENIX Security Symposium 2013.

FiE

- Based on KLEE
- Targets MSP430 firmware
- Symbolic Execution
- Uses explicit analysis, memory and interrupt specifications
- Requires source code of firmware

Davidson, Drew, et al. ”FIE on Firmware: Finding Vulnerabilities in Embedded Systems Using Symbolic Execution.” USENIX Security Symposium 2013.

Firmadyne

- Based on Qemu
- Targets ARM & MIPS firmware
- Instrumented Linux kernel
- Automated analysis of web pages and SNMP implementations
- Automated testing with known exploits

Chen, Daming D., et al. "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware." NDSS 2016.

Firmadyne

- Based on Qemu
- Targets ARM & MIPS firmware
- Instrumented Linux kernel
- Automated analysis of web pages and SNMP implementations
- Automated testing with known exploits
- Works only for Linux based firmware with no too specific kernel modules

Chen, Daming D., et al. "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware." NDSS 2016.

LuaQemu

- Based on instrumented QEMU
- Work in progress
- Example targets BCM4358 firmware
- Prototyping of Boards with LUA
- Instrumentation capabilities

https://github.com/Comsecuris/luaqemu

LuaQemu

- Based on instrumented QEMU
- Work in progress
- Example targets BCM4358 firmware
- Prototyping of Boards with LUA
- Instrumentation capabilities
- Requires a significant amount of modeling and trial & error

https://github.com/Comsecuris/luaqemu

Avatar

- Based on S^2E (QEMU+KLEE) and OpenOCD/GDB
- Targets ARM firmware
- Partial emulation together with real hardware
- I/O forwarding
- Orchestration
- Symbolic Execution

Zaddach, Jonas, et al. "AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares." NDSS 2014.

Avatar

- Based on S^2E (QEMU+KLEE) and OpenOCD/GDB
- Targets ARM firmware
- Partial emulation together with real hardware
- I/O forwarding
- Orchestration
- Symbolic Execution
- Heavily tied to the S^2E infrastructure
- Requires the presence of the physical device

Zaddach, Jonas, et al. "AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares." NDSS 2014.

Observations

- A lot of focus on ARM
- QEMU's emulation capabilities are a common building block
- Frameworks are heavily bound to underlying components

The avatar^2 framework

The big picture

- Dynamic Multi-Target Orchestration and Instrumentation Framework
- Focus on firmware analysis
- Python based framework
- Re-designed and re-implemented from scratch
- Open source: https://github.com/avatartwo
  - Research project
  - Released in June 2017

Who?

- Developed by the Software and System Security Group at Eurecom
- Specifically:
  - Marius Muench
  - Dario Nisi
  - Aurélien Francillon
  - Davide Balzarotti

http://s3.eurecom.fr/

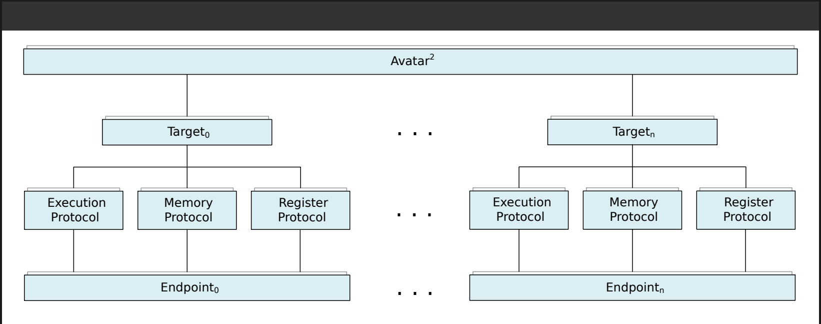
main goals

- Target orchestration
  - Abstraction of debuggers, emulators and other frameworks
  - Easy addition of new targets
- Separation of execution and memory
  - Enables I/O forwarding/remote memory
- State transfer and synchronization
  - Don't keep the state of analysed software local to single targets

avatar^2- components

- avatar^2 core
- Targets
- Endpoints
- Protocols

avatar²- architecture overview



Implemented Targets

Implemented Targets

GDB

Implemented Targets

GDB

QEMU

Implemented Targets

GDB

PANDA

QEMU

Implemented Targets

GDB

PANDA

QEMU

angr[1]

[1] Still under development

Changes to QEMU

Avatar^2 provides a costomized QEMU

- All located in a single subfolder: hw/avatar
- New board: Configurable Machine
  - Already present in the first avatar
  - Allows flexible configuration of emulated hardware
- New peripheral: avatar-peripheral
  - Communicates with avatar^2 via posix message queues
  - Utilizes custom remote-memory protocol

Additional features

- Architecture independent design
- Internal memory layout representation
- Legacy python support
- Peripheral modeling
- Plugin System
  - Assembler/Disassembler
  - Orchestrator
  - Instruction Forwarder

Examples

avatar^2-scripting: High-Level Overview

An avatar^2 scripts needs to:

1. Create the Avatar-object
2. Define a set of targets
3. Optionally define memory layout
4. Specify an execution plan

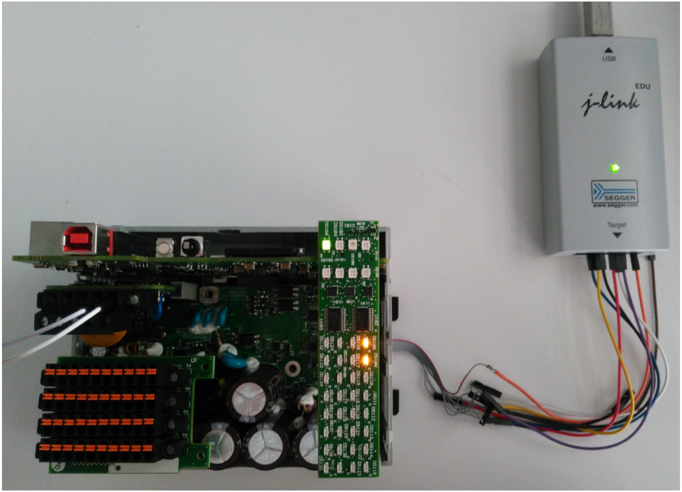
Demo

- Let's move on to a real target!
- Proof of concept implementation of HARVEY[2]
    - Malware for a COTS PLC
    - The plc utilizes multiple boards
    - Code injection via JTAG

[2] Garcia, Luis, et al. "Hey, My Malware Knows Physics Attacking PLCs with Physical Model Aware Rootkit." NDSS 2016.

(Fragile) Demo

Demo backup ;)



Improving Fault Detection

- Part of WYCINWYC[3]
    - Joint work with SIEMENS
- Investigates challenges specific to fuzz testing embedded devices
    - Fault detection
    - Instrumentation
    - Scalability
- Evaluates different strategies to aid fuzz-testing
    - Uses avatar[2] for partial and full emulation of the firmware

[3] Muench, Marius, et.al. "What you corrupt is not what you crash: Challenges in Fuzzing Embedded Devices" To be presented at NDSS 2018

The setup

- Two Targets
  - STM32l152re
  - PANDA
- Target Software
  - expat, a popular XML-parser
  - Artificially inserted vulnerabilities
- Orchestration
  - Board initilization on physical device
  - Emulation of main-loop inside PANDA
- Analysis
  - 5 PANDA plugins to detect different types of vulnerabilities
  - Mimicry of existing techniques for desktop software
  - Doesn't require modification of the firmware

Evaluation

- 100 Fuzzing sessions in different setups
  - Native
  - Partial emulation with I/O forwarding
  - Partial emulation with $avatar^2$-peripherals
  - Full emulation
- Plugins could detect previously undetected faults
- Full emulation provided better performance than native fuzzing
- More details in the paper: http://s3.eurecom.fr/docs/ndss18_muench.pdf

Record & Replay

- Dynamic binary analysis of firmware requires often the device
- PANDA allows to record and replay execution
- Allows exchange of executions fur further analysis without the device

Demo

Symbolic Execution and Complex Software (WIP)

- Firefox with inserted bug
  - Executed concretely inside gdb until function of interest
  - Analysis of only one thread
- Automated memory layout extraction from gdb
- Transfer of layout into angr
- Copy-On-Read
- Symbolic function arguments

Symbolic Execution and Complex Software (WIP)

Preliminary Results:

- Approximatly 10 minutes of runtime
- 36 executed basic blocks
- 21 uniquely accessed pages
- Found the bug

Examples: Recap

5 Examples:

- Dynamic Instrumentation of GDB
- Dynamic Instrumentation of a plc
- Fault Detection with an development board and PANDA
- Record and Replay with an development board and PANDA
- Symbolic Execution with firefox and gdb

Conclusion

Conclusion

- Dynamic firmware analysis is still a challenging topic
- Avatar^2 aims to tackle some of the challenges
- Multi-target orchestration is not limited to firmware

Plans for 2018

- Move main development to github
- Introduce proper versioning
- More, exciting targets

Wanna help?

Get in touch with us:

- #avatar2@freenode
- avatar2@lists.eurecom.fr
- Talk to me

We may be looking for people to join our group in the near future

Shouts

- S3@Eurecom
- jzaddach
- Subwire & domenukk
- Zardus & ccm
- Tasteless

Thank you!