Dynamic Security Analysis

in embedded firmware

Avatar

A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares

Jonas Zaddach, Luca Bruno, Aurelien Francillon, Davide Balzarotti

Static vs Dynamic Analysis

Static analysis inspects source or object code

- lint

Dynamic analysis is performed on running code

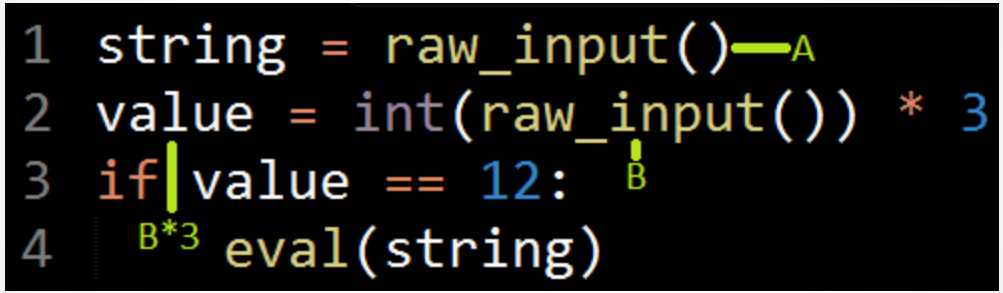
- Symbolic Execution
- Dynamic Taint Propagation
- Whitebox Fuzzing

Symbolic Execution

Determines interesting input

Input provided not actual values

Symbolic Execution - Example



Symbolic Execution

Reasons path-by-path

Path explosion

Dynamic analysis needs an emulator

Difficult to emulate an embedded system

AVATAR is a hybrid model

Architecture

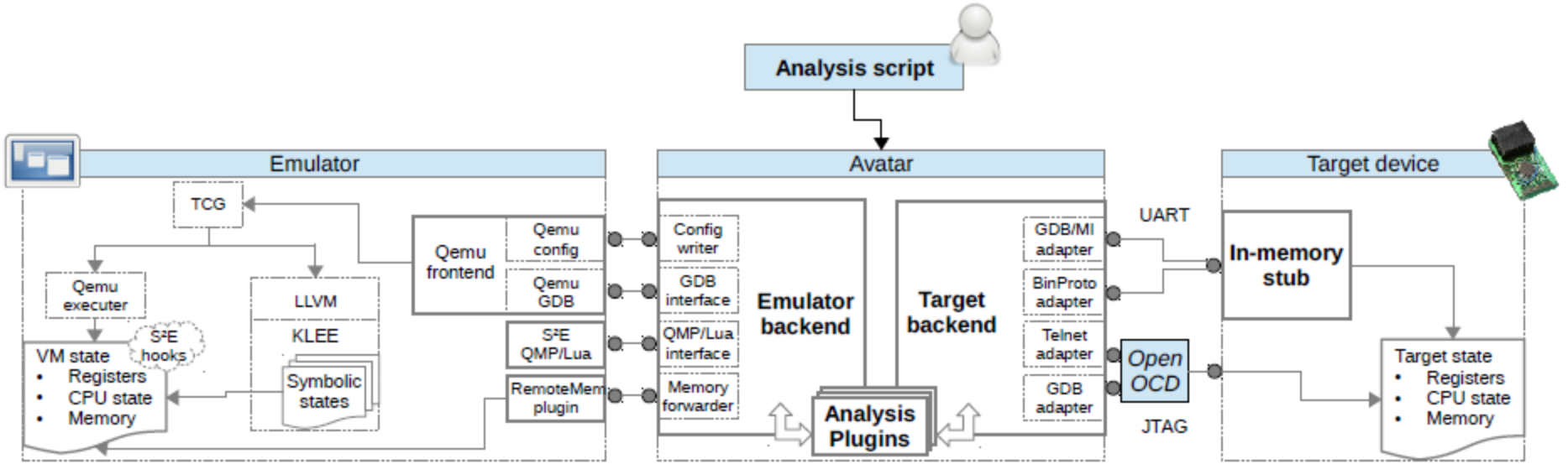


Fig. 1: Overview of *Avatar*.

Full-Separation Mode and Context Switching

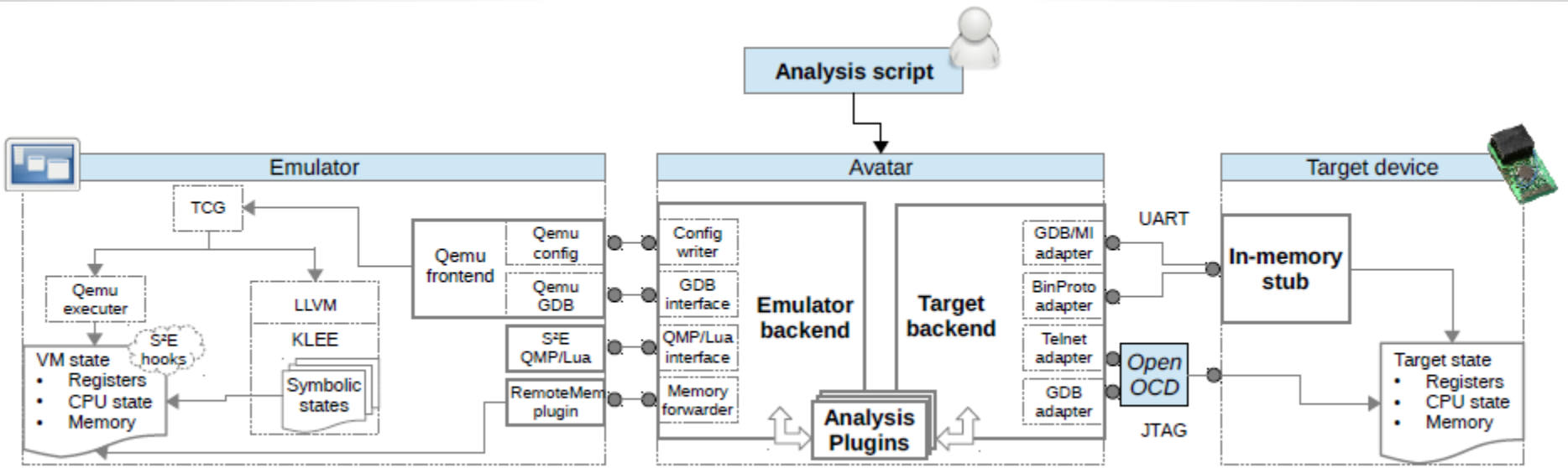


Fig. 1: Overview of *Avatar*.

Memory Optimization

Access type	Read	Write	Cumulative
Code	61,632	-	61,632
Stack & data	646	1,795	64,073
I/O	3,614	2,097	69,784

TABLE I: Number of memory accesses grouped by memory regions for the HDD bootloader.

Replaying I/O Operations

Record and replay later

Selective Code Migration

Mark a function as local to physical device

A light-weight form of the context switch

Makes use of static analysis

Significantly improves performance

Tested hardware

- Hard disk bootloader
- Wireless sensor node (Econotag)
- GSM feature phone

	Target device	Manufacturer and model	System-on-Chip	CPU	Debug access	Analyzed code	Scope of analysis
Experiment VI-A	Hard disk	*undisclosed*	unknown	ARM966	Serial port	Bootloader	Backdoor detection
Experiment VI-B	ZigBee sensor	Redwire Econotag	MC13224	ARM7TDMI	JTAG	ZigBee stack	Vulnerability discovery
Experiment VI-C	GSM phone	Motorola C118	TI Calypso	ARM7TDMI	JTAG	SMS decoding	Reverse engineering

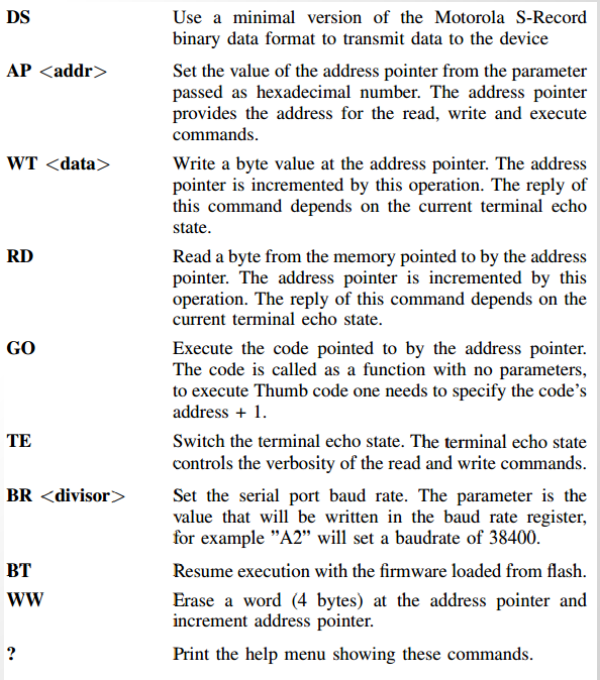
TABLE II: Comparison of experiments described in Section VI.

Hard disk - backdoor detection

Two bootloader stages before OS load

Issues encountered

No backdoor found



DS	Use a minimal version of the Motorola S-Record binary data format to transmit data to the device
AP <addr>	Set the value of the address pointer from the parameter passed as hexadecimal number. The address pointer provides the address for the read, write and execute commands.
WT <data>	Write a byte value at the address pointer. The address pointer is incremented by this operation. The reply of this command depends on the current terminal echo state.
RD	Read a byte from the memory pointed to by the address pointer. The address pointer is incremented by this operation. The reply of this command depends on the current terminal echo state.
GO	Execute the code pointed to by the address pointer. The code is called as a function with no parameters, to execute Thumb code one needs to specify the code's address + 1.
TE	Switch the terminal echo state. The terminal echo state controls the verbosity of the read and write commands.
BR <divisor>	Set the serial port baud rate. The parameter is the value that will be written in the baud rate register, for example "A2" will set a baudrate of 38400.
BT	Resume execution with the firmware loaded from flash.
WW	Erase a word (4 bytes) at the address pointer and increment address pointer.
?	Print the help menu showing these commands.

Econotag - Vulnerability Discovery

Source code changed to add a vulnerability

Symbolic execution found it

No vulnerability found in original firmware

GSM feature phone - SMS

Only one CPU and OS.

Simpler bootloader than HDD

No selective code migration was needed

GSM stack proved too complex

Conclusion

Found no serious vulnerabilities

Proved that Avatar is versatile

Questions?