An FPGA-based Control-Flow Integrity solution for securing Embedded Systems

Nicolò MAUNERO
CINI Cybersecurity National Lab
Turin, Italy
nicolo.maunero@consorzio-cini.it

Paolo PRINETTO
DAUIN - Politecnico di Torino
CINI Cybersecurity National Lab
Turin, Italy
paolo.prinetto@polito.it

Gianluca ROASCIO
CINI Cybersecurity National Lab
Turin, Italy
gianluca.roascio@consorzio-cini.it

Runtime memory vulnerabilities, especially present in widely used languages in embedded systems as C and C++, are exploited by attackers to corrupt code pointers and hijack the execution flow of a program running on a target system to force it to behave abnormally. This is the principle of modern *Code Reuse Attacks* (CRA) and of famous attack paradigms as *Return-Oriented Programming* (ROP) and *Jump-Oriented Programming* (JOP), which have defeated the previous defenses against malicious code injection such as *Data Execution Prevention* (DEP).

Control-Flow Integrity (CFI) is a promising approach to protect against such runtime attacks. Many CFI solutions have recently been proposed, resorting to both software and hardware solutions. Unfortunately, when dealing with embedded systems, pure software defences very often lead to unacceptable overhead, since they mostly rely on code instrumentations to protect control-flow transfer instructions. Hardware-assisted techniques are based either on modifying the processor architecture or on using the debug interface and an external monitor. While the former solution is inapplicable for legacy systems, the latter is limited by the fact that the processor is forced to stop and send traces when required, with consequent heavy performance loss.

The proposed solution exploits the presence of an external FPGA, connected to a microcontroller through a normal parallel interface, without the need of tight integration, thus resulting applicable to any system. The FPGA works as a parallel supervisor of the program's intended *Control-Flow Graph* (CFG). After compilation, the CFG is computed and several information items related to insecure transfer are pre-loaded on the FPGA. During the execution, when the program has to perform insecure transfer instructions, it communicates its locations to the FPGA supervisor, through secure, simple and low-overhead store operations. If an abnormal deviation is detected, the microcontroller activity is immediately stopped.

The proposed solution also secures the execution of interrupt handlers, very frequent in microcontroller applications and often forgotten by state-of-the-art CFI solutions, for the impossibility of tracking them through static analysis.