A FPGA-based Control-Flow Integrity Solution for Securing Bare-Metal Embedded Systems

Vahid EFTEKHARI MOGHADAM
DAUIN - Politecnico di Torino
CINI Cybersecurity National Lab
Turin, Italy
vahid.eftekharimoghadam@studenti.polito.it

Matteo FORNERO
CINI Cybersecurity National Lab
Turin, Italy
matteo.fornero@consorzio-cini.it

Nicolò MAUNERO
DAUIN - Politecnico di Torino
CINI Cybersecurity National Lab
Turin, Italy
nicolo.maunero@polito.it

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

Gianluca ROASCIO
DAUIN - Politecnico di Torino
CINI Cybersecurity National Lab
Turin, Italy
gianluca.roascio@polito.it

Embedded devices are nowadays playing a central role in our lives, as they control most of the objects surrounding us. In addition, such systems create a network of connections that goes far beyond simple isolated LANs and links up devices all over the world. A huge amount of sensitive data is thus exchanged, and related security and privacy issues must be addressed.

In addition to communication security, a relevant aspect is the protection of devices themselves and their resilience to unauthorised intrusions. Physical security is certainly a first step, but not enough, since vulnerabilities may be contained in the code that the systems execute. Many of these vulnerabilities derive from the widespread use of very powerful languages such as C and C++. These languages guarantee a high degree of low-level control, but at the same time they allow programmers to freely manipulate memory pointers, so that common weaknesses such as buffer overflow or dangling pointers come out. Memory corruption vulnerabilities may enable attackers to maliciously take control over the program running on a target machine by forcing it to execute an unintended sequence of instructions present in memory. This is the principle of modern Code-Reuse Attacks (CRAs) and of famous attack paradigms as Return-Oriented Programming (ROP) and Jump-Oriented Programming (JOP).

Control-Flow Integrity (CFI) is a promising approach to protect against such runtime attacks. Recently, many CFI-based solutions have been proposed, resorting to both hardware and software implementations. However, many of these solutions are hardly applicable to microcontroller systems, often very resource-limited. The proposed defence mixes software and hardware instrumentation and is based on monitoring the Control-Flow Graph (CFG) with an FPGA connected to the CPU. The solution, applicable in principle to any architecture which disposes of an FPGA, forces all control-flow transfers to be compliant with the CFG, and preserves the execution context from possible corruption when entering unpredictable code such as Interrupt Services Routines (ISR). It provides a generic, portable, and lightweight CFI solution for bare-metal embedded systems. The firmware is statically analysed to extract CFG information and undergo a process of binary instrumentation. Particular attention is put in identifying only those control-flow transfer instructions that may be vulnerable to an attack in order to reduce to the minimum the introduced overhead. CFG information are stored on the FPGA, through the introduced instrumentation, the FPGA can monitor the execution of the program detecting any anomalies. CPU-FPGA interaction happens by mean of single store instruction, relieving the CPU from heavy computation delegating control-flow integrity checks to the FPGA.