The C programming language is one of the most popular languages in embedded system programming. Indeed, C is efficient, lightweight and can easily meet high performance and deterministic real-time constraints. However, these assets come at a certain price. Indeed, C does not provide extra features for memory safety. As a result, attackers can easily exploit spatial memory vulnerabilities to hijack the execution flow of an application. The demonstration features a real-time connected infusion pump vulnerable to memory attacks. First, we showcase an exploit that remotely takes control of the pump. Then, we demonstrate the effectiveness of BackFlow, an LLVM-based compiler extension that enforces control-flow integrity in low-end ARM embedded systems.

A life-critical embedded system such as a connected insulin pump is a perfect example to demonstrate the criticality of a memory-based attack. The demonstration features “SecPump”, a connected infusion pump that models the dosage and performance of a drug administered to a patient in real-time. SecPump uses a STM32 NUCLEO F446RE micro-controller with FreeRTOS, a real-time operating system and the Bluetooth Low Energy protocol to communicate with a remote interface.

As proof of concept, “SecPump” suffers from a memory corruption bug in the Bluetooth Low Energy packet parser task. Such bug allows an attacker to violate the execution flow of “SecPump” and perform remote code execution. The demonstration highlights a critical code-reuse attack that changes the amount of injected drug to a patient at run-time.

The second part of the demonstration highlights BackFlow, a compiler-based toolchain that enforces indirect backward-edge control-flow integrity. The control flow integrity generated by the compiler relies on a bitmap where each set bit indicates a valid pointer destination. In the demonstration, BackFlow is used to protect the vulnerable infusion pump and makes the previous code-reuse attacks ineffective. The obtained results show that the control flow integrity protection incurs an execution time overhead ranging from 1.5 to 4.5%.

reference paper: “BackFlow: Backward Edge Control Flow Enforcement for Low End ARM Microcontrollers”