Enhancing the Safety and Trustworthiness of Medical Devices

Meng Zhang, Chunxiao Li, Niraj K Jha
Department of Electrical Engineering, Princeton University
Anand Raghunathan
School of Electrical and Computer Engineering, Purdue University

Background

Wearable and implantable medical devices commonly used for diagnosing, monitoring, and treating various medical conditions.

However, new advances open up attack opportunities:
- Increasing programmability: software exploits
- Increasing connectivity: wireless attacks

Different from general-purpose computing system security
- Consequence can be life-threatening
- Extreme resource constraints

Wireless Attack

Setup: insulin pump, glucose meter, Universal Software Radio Peripheral (USRP).

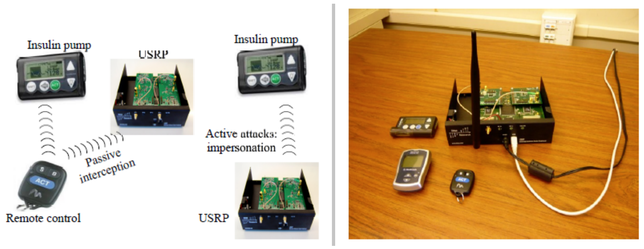

Fig. 1. Experimental Setup.

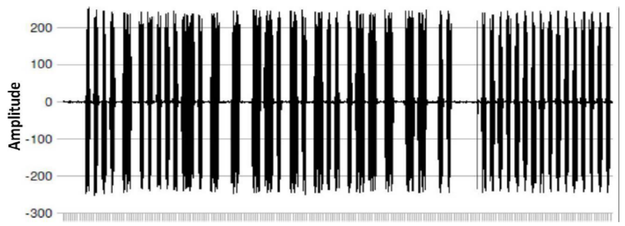

Fig. 2. Signal intercepted by USRP

Attack Scenarios
- **Passive attacks:** disclose glucose level and device PIN
- **Active attacks:** Send outdated data/command; send arbitrary glucose level data to trigger false therapy; stop/resume insulin injection; inject an insulin dose to cause hypoglycemia.

Effective distance of attack depends on antenna, battery level, etc. In our set-up, it is easy to remotely control the pump from more than 20 meters away.

Defense Against Wireless Attacks

MedMon
- Snoops on all radio-frequency wireless communications to/from medical devices.
- Uses multi-layered anomaly detection to identify potential malicious transactions.
- Requires no modification to existing devices.

Anomaly Detection
- Physical: received signal strength indicator (RSSI), time of arrival (TOA), differential time of arrival (DTOA), angle of arrival (AOA).
- Behavioral: underlying information

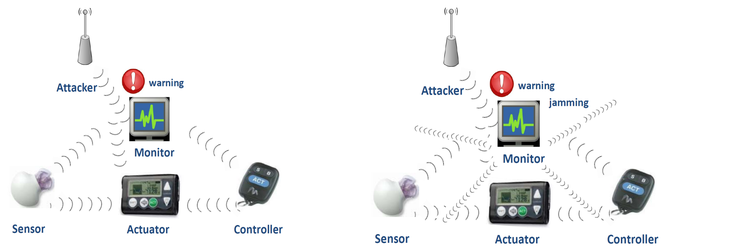

Fig. 3. Passive mode (left) and active mode (right).

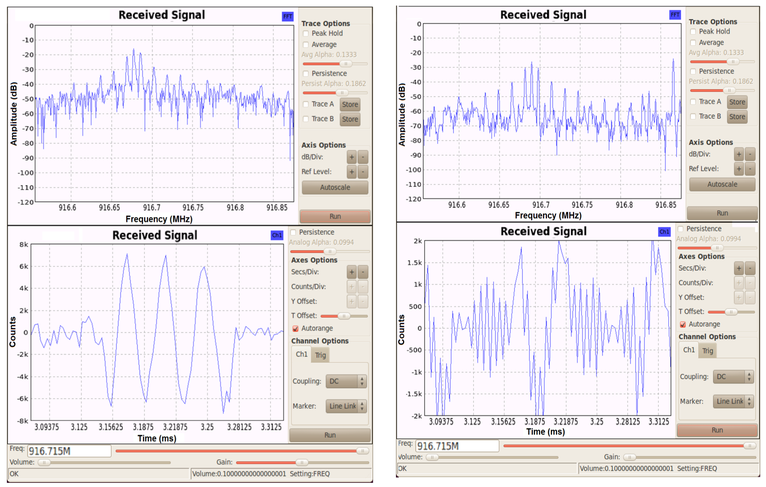

Fig. 4. Received signal in the frequency (top) and time domains (bottom) when MedMon is in passive (left) and active (right) mode.

Actions upon detection (Fig. 3)
- Passive: Notify the user
- Active: Jam the packets so that they do not reach device

Result (Fig. 4): Successfully detects virtually all naive attacks. Successfully jams malicious signal before insulin pump receives it.

Defense Against Software Exploits

Formal verification: process of using mathematical analysis to verify that a design conforms to some precisely-expressed notions of functional correctness.

Limitations
- Plain C/C++ code vs. embedded C code. Need code transformation techniques.
- Generic program properties vs. domain-specific properties and specifications. Need real-world device safety properties verification techniques.

Code Transformation (Fig. 5)
- Direct hardware accesses: Define a unique variable for each memory-mapped I/O access.
- Hardware-specific instructions: Make hardware-specific changes.
- Timer interrupts and peripheral interrupts (ADC, serial ports).

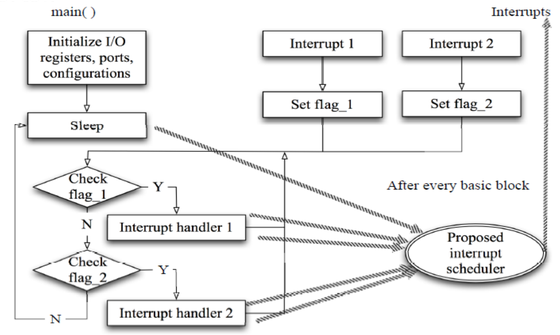

Fig. 5. Code transformation.

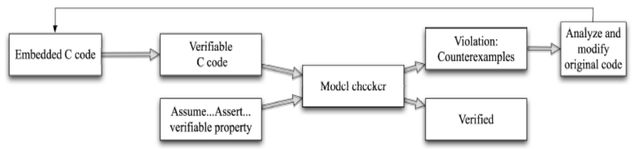

Fig. 6. Verification process is done iteratively until the property is verified.

Pacemaker Software Case Study: Safety properties verified and example vulnerabilities/bugs found