Security mechanisms for low-end embedded systems

A Proof-of-Concept for Home and Building Automation

Thomas Flanitzer

INTRODUCTION

GOALS
- Provide general security in low-end embedded systems
- Allow the execution of arbitrary, untested user applications
- Protect the system and its environment from software attacks
- Solution shall be applicable to standard devices
- No hardware modifications shall be required
- Provide decent flexibility
- Solution shall be efficient

Low-end embedded systems are typically found in areas where very strict demands in terms of cost, power consumption and resources have to be met.

A security attack is an attempt to exploit a vulnerability in a system. The risk of an attack being launched is referred to as threat. A system is secured by defining a security policy which is enforced by security mechanisms.

BACKGROUND

THREAT ANALYSIS

- Embedded System Attacks
  - Software malfunction
  - External interference
  - Malicious code

- Side Channel Attacks
  - Power analysis
  - Timing analysis
  - Fault behavior analysis

- Physical Attacks
  - Tampering
  - Magnetic spraying
  - Decapsulation

ATTACK TYPE COMMONNESS

- Static code analysis
- Code signing
- Proof-carrying code

- Intrusion detection
- Software monitoring
- Sandbox, virtualization
- Self-checking code
- Attack specific counter mechanisms
- Hardware supported mechanisms
- Secure coprocessor
- Physical partitioning
- Harvard architecture
- Execution prevention

PROPOSED ARCHITECTURE

ARCHITECTURE OUTLINE

User applications are executed in a sandbox and can only interface with the system and its peripherals through a restricted user API. The execution is managed by the system software which implements the required libraries and the network stack. The invocation frequency of execution is managed by the system software which implements the required functionality (EIB/KNX network stack & system software. Java libraries provide the user applications with the required functionality (EIB/KNX network stack & system peripherals). Invocation monitoring and a flexible configuration are applied as modifications to the NanoVM.

PROOF-OF-CONCEPT

PROOF-OF-CONCEPT OUTLINE

A hardware platform closely resembling the functional profile of a SAC-device in HBA systems (in the specific case in an EIB/KNX network). An ATMEG168 microcontroller is used together with the Freebus basic circuit for EIB/KNX interaction. The NanoVM Java interpreter is as the core component of the software. Java libraries provide the user applications with the required functionality (EIB/KNX network stack & system peripherals). Invocation monitoring and a flexible configuration are applied as modifications to the NanoVM.

IMPLEMENTATION CHARACTERISTICS

HARDWARE
- ATMEG168 Microcontroller @ 8MHz
- Freebus basic circuit for interfacing with EIB/KNX networks
- www.freebus.org

SOFTWARE
- NanoVM, a heavily reduced Java interpreter
- → www.harbaum.org/till/nanovm
- NanoVM native Java libraries for EIB/KNX interaction and for accessing peripherals
- Invocation monitoring as a modification of the bytecode interpreter

RESULTS

The example program shows basic EIB/KNX bus interaction. After only a few initialization steps an infinite loop is entered where the group object is polled for changes. If present, its value is signaled via a LED. Additionally, the state of the pushbutton is checked. If it has been pressed, a toggling value will be written to group object 1.

CONCLUSIONS

CONCLUSIONS
- Promising architecture
- Reasonable combination of adapted security mechanisms
- Proof-of-Concept demonstrated feasibility and usefulness
- Interesting platform for future investigations

Contact: t.flanitzer@gmx.at