Keywords: security, embedded security, software isolation, IoT, internet of things, access control

APPLICATION NOTE 6761

HOW TO PROTECT IOT DEVICES USING A SOFTWARE ISOLATION SOLUTION

Abstract: Internet of things (IoT) devices must be secured from communications, asset, and integration standpoints. This application note discusses the security considerations for each of these areas, and presents a software isolation solution that provides a strong level of protection for smart, connected designs. A similar version of this application note appeared in EENews Europe in January 2018 and in Elettronica Oggi in May 2018.

Introduction

The scenarios and situations that connect the IoT all share the same security requirements related to communication, assets, and integration. Communications must flow securely, assets must be securely stored and processed, and the integration of the IoT device must be consistent from a security standpoint.

Before we delve into ways to protect your IoT devices, first consider all the key aspects of an IoT device: the architecture, communications, integration, and assets.

Architecture

Building connected objects means combining many functions, such as the ability to perform multiple processing tasks, to secure some data, and to communicate with back-end entities such as gateways, servers, or the cloud. It is a tall order of functionalities that must be integrated into the smallest, cheapest, and simplest embedded system design.

This aggregation of features, however, leads to an even more important exposure to attackers. Attackers see the assets handled by the IoT devices as their goldmine and their communication interfaces as the entry point into these assets, facilitated by a permissive integration and design.

Communications

An IoT device is a link in the chain from data acquisition to data exploitation, or between the command control and the operation. Without any communication interface, such a device is useless, as it would be impossible for it to exchange data or update its firmware and services. This means that an IoT device must always be wired or wirelessly linked to a network.

Integration

Once the hardware device is in the field, we can say the main milestone has been achieved. An important enough business case has been identified to justify the development, manufacturing, and deployment of the device. The developer must also find ways to exploit this device for the longest possible lifetime. The best way to do this is to improve and add features along with applications. However, this means that the same platform and the same microcontroller host several different applications. These applications must coexist, concurrently executing to address the device features.

Talking about multiple applications implicitly leads to a rich OS, with memory isolation mechanisms. The consequence is that only high-end CPUs (e.g., based on an Arm® Cortex®-A core) with large memories (RAM, flash) achieve these requirements. This appears to strongly contradict the primary goals of being the smallest, cheapest, and simplest.

Assets

The connected device handles assets, including sensitive and valuable data. Whether it acquires, modifies, sends, or stores the data, in the end, it exposes this information. This security requirement also adds some complexity.

Access Control Example

As an example, consider an embedded device attached to a computer that reads and handles fingerprints. It communicates in a bidirectional mode using a wireless channel (Bluetooth, Li-fi) to run applications and to host the fingerprint minutiae verification information and a secret key . This device's first purpose is to control users' access to a highly valuable service located on a remote server, accessible through a client computer.

This is the two-factor authentication method. When a user requests access to the service, a random number (a challenge) is generated and sent by the server to the embedded device through the computer. Then, if and only if the owner's fingerprint matches the stored minutiae, the challenge is cryptographically linked (encrypted or signed) with the secret key, and finally sent back to the remote server for authentication.

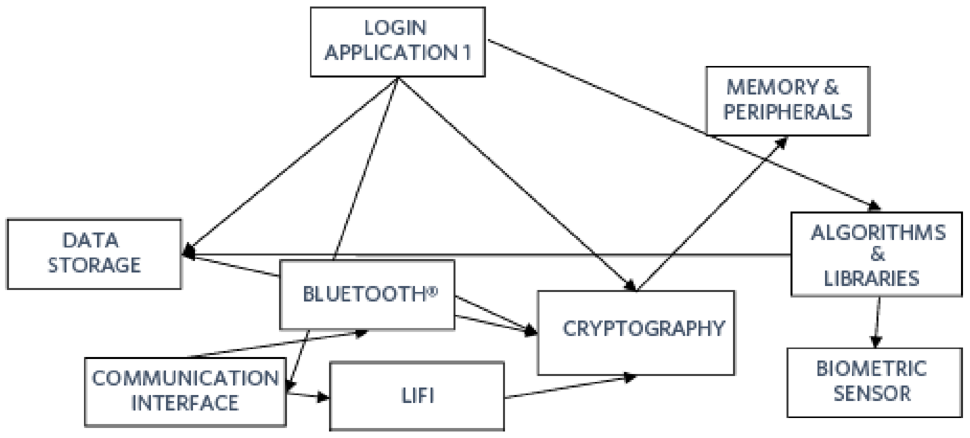

Figure 1. IoT device with a flat architecture model on an MPU-equipped microcontroller.

In the interest of extending this device's lifetime, new applications can be added to provide additional access controls to other high-value services. The biometric capability is very attractive and can be the start for many innovative services, like payment processing. But from a security standpoint, having multiple providers, applications, and keys sharing the same hardware platform is a drawback.

This kind of device is generally constrained by the price and the form factor. Therefore, the choice of the microcontroller running the device leans in favor of the smallest embedded memories and the simplest architectures, and the modern choice is the Arm® Cortex®-M processor. Whatever the software development model, either internal team or subcontractor, the complexity and the variety of the embedded code increases tremendously when moving from single to multiple applications. Consequently, since the Cortex®-M core only features a memory protection unit (MPU), there is a very high risk that the device's software adopts a flat architecture model. In such a model, all applications share the same privileges and resources. Only good practices, the good will of application developers, thorough code verification, and luck can guarantee that each application does not disturb or pollute others.

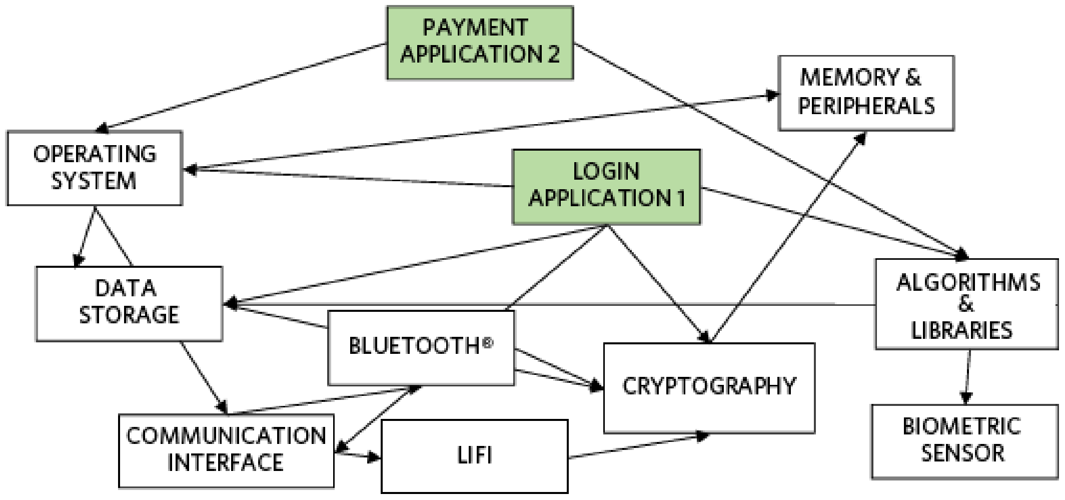

Figure 2. Payment terminal on a flat architecture model on an MPU-equipped microcontroller.

Problem: A Weakness in One Affects All

The riskiest proposition of a flat architecture model is that any weakness or failure in any application can impact all the other applications and, therefore, how the entire device functions. Without real protection and segregation, an attack can spread from the weakest application to the others because of the uncontrolled resource sharing and the lack of execution privilege distinction.

A preferred attack channel is generally the external communication interface, which presents an easy path for an attacker. It is, by definition, accessible from outside the device and ideally remotely accessible over a network. The software stack handling the communication is always functionally validated but very rarely verified security-wise.

Typically, an attack path targeting a weakness in a communication stack takes advantage of a buffer overflow or a lack of input data check. For instance, an attacker may inject malicious code in the microcontroller's internal RAM through the improperly checked communication buffers. When this code is within the platform and executed, the attacker can take full control of the software running on the device. The consequences are only be limited by the attackers' imagination such as stealing biometric information, performing relay attacks, and any other scenario based on the embedded applications.

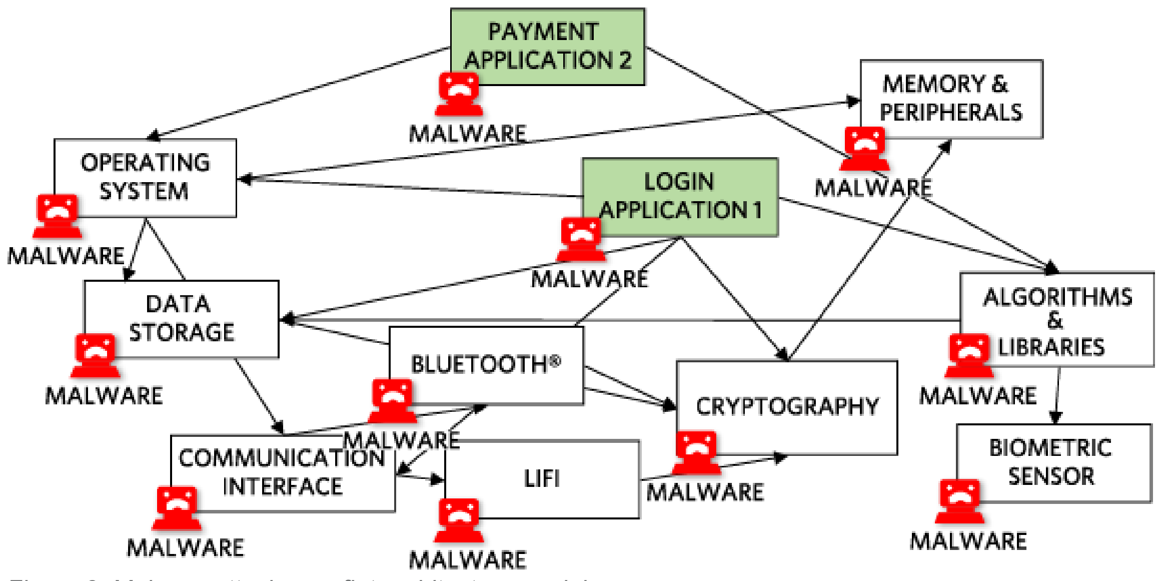

Figure 3. Malware attack on a flat architecture model.

Solution: Isolating the Software
To counter a malicious attack, designers can turn to isolation software solutions that utilize hardware security mechanisms such as a memory management unit (MMU) or the Arm® TrustZone®. Called hypervisors or virtual machines, these software solutions use hardware mechanisms to define separate software boxes or containers, each containing an application. The hypervisor's main objective is to allocate resources to each of these boxes, to control these resources' usage, and to control the communications between these boxes. It is commonly thought that such an approach is impossible on Arm® Cortex®-M-based microcontrollers because they do not have an MMU and only an MPU.

Maxim's DeepCover® Security Framework (DSF) software isolation solution proves that a high level of security can be reached and guarantees a very strong isolation between the software boxes running on the Cortex®-M-based microcontrollers. At the heart of this solution, there is a hypervisor that enforces a strict software architecture configuration at any time, defining boxes, the boxes' resources, and the boxes' interactions. Different applications run in separate boxes, each with their defined set of privileges and resources. Applications can interact with each other through gateways. Everything described in the configuration is guaranteed by the hypervisor that guarantees that the boxes' execution matches the configuration. Access out of allowed memory ranges, attempts to interact with another box out of an allowed gateway, direct access to disallowed resources, and any other differences with the intended configuration are detected right away. The hypervisor manages and controls other boxes' resources and execution, keeping its own ones for its execution.

This software isolation solution provides a high level of confidence in terms of application security. A security laboratory has rated the framework as compliant with the very demanding PCI-PTS POI security requirements for software isolation. What is secure for the payment can also be considered secure for the IoT.

In the example, DSF allows a simple but efficient architecture design that consists of the exclusive allocation of a specific box for the management of the biometric characteristics, including access to the fingerprint sensor and the extraction of the minutiae. By preventing access to this sensor from other boxes, and thanks to the hypervisor's control over this access, any fraudulent or unwanted attempt to read the sensor is detected and an exception is raised, leading to a safe failure. Following the same model, other boxes can be defined and developed with a focus on their very specific functional objective, after a careful study of their needs in terms of resources and peripherals. The communication links between boxes are enforced by the hypervisor, which guarantees that only these links can be used for data exchanges between the boxes. It is impossible to perform an arbitrary jump to a random code location or to provide direct access to another's boxed resources.

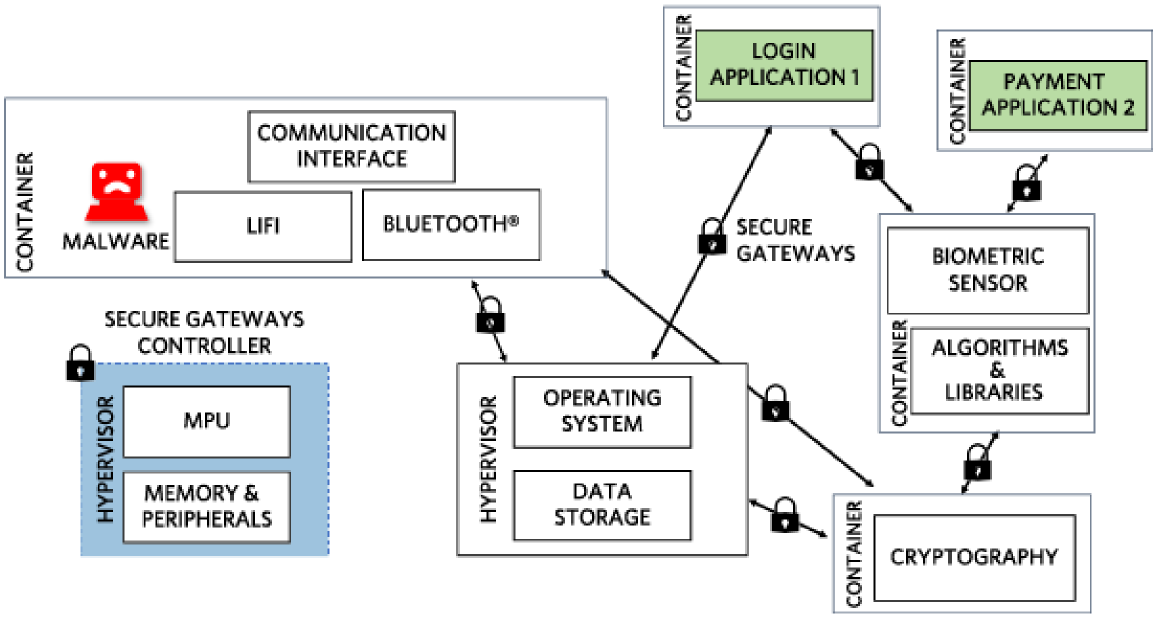

Figure 4. Malware containment using the DSF software isolation solution.

Such a framework does not improve developers' skills or turn buggy software into a bug-free application. However, it does mitigate the attack scenario described because it prevents a weakness in an application from exposing the whole firmware. Since the bug remains contained in its box, its value is dramatically decreased.

A great benefit of this solution is that it allows hosting of multiple applications from multiple providers with different levels of confidence and robustness on a single device. The owner of the most demanding application can now accept other applications on the same hardware device without the risk of weakening his own application and exposing his assets.

Benefits: Easy, Trustworthy Application Updates

Given the high level of control in the boxes' architecture and resource partitioning, applications can be updated easily and with trust. Applications are fully controlled by the hypervisor, which can modify, update, or remove them through simple operations without harming the functioning of the others. In addition, that secure update can propose additional services like encryption without exposing the keys to any box.

Finally, another important benefit is that, as one box execution does not hamper another box execution, any certification granted to one box is not impacted by another box modification and the certification remains valid. What was a single, monolithic piece of software, where each modification could impact the robustness of the whole entity as well as the certification of some portions of it, has been transformed into a set of independent, isolated images, each maintaining their own properties and bugs.

Trademarks

Arm® is a registered trademark and registered service mark of Arm Limited.
DeepCover® is a registered trademark of Maxim Integrated Products, Inc.
Cortex® is a registered trademark of Arm Limited.
TrustZone® is a registered trademark of Arm Limited.
