print | close

# What's the Difference Between Software and Hardware for Embedded Device Security?

*Electronic Design*
Michael Armentrout
Wed, 2016-07-06 09:02

Massive interest in development for the Internet of Things (IoT) has raised the volume of conversation about system security for embedded devices *(see figure)*. This is great news, as the industry is moving rapidly from thinking about security as an afterthought and now considering security strategies as part of the design process. Embedded designers can look at experience in similar networked systems to shorten their learning curve.

The core challenge in security is to provide a "root of trust" that's used by each device in a system to mutually validate authenticity and prevent unauthorized activity or attacks. Trust relies on a cryptographic mechanism that curtails an attacker's ability to obtain system information. There are multiple ways to implement the cryptographic mechanism, which leads to questions about how to design a robust security approach that protects both system integrity and user privacy:

• Is it feasible to use only software to execute the cryptographic mechanism that protects system code?

• How can a design isolate the cryptographic mechanism from attack?

Related

5 Questions Automotive Designers Should Ask About Hacking

What is Really Happening in the World of Cybersecurity?

7 Things to Know About IoT Security

Security evaluations for embedded devices involve understanding the tradeoff between the cost of protecting the system and the risks and consequences of a successful attack. The next step is to consider in what ways the embedded device is vulnerable and what methods an attacker might use to penetrate the system. An understanding of these factors—which involve both technology and business case issues—forms the background for determining a security strategy.

### Attack Methodology

The objectives of an attack are consistent, regardless of whether the system serves an industrial, commercial, or consumer market:

• Eavesdropping on data or commands, with the intent to obtain confidential information.

• Using an unauthorized (fake) device to introduce false data or corrupt a control process, with the goal of

sing an unwanted reaction or masking a physical attack.

• Using an unauthorized (fake) server to send incorrect commands that trigger unwanted events.

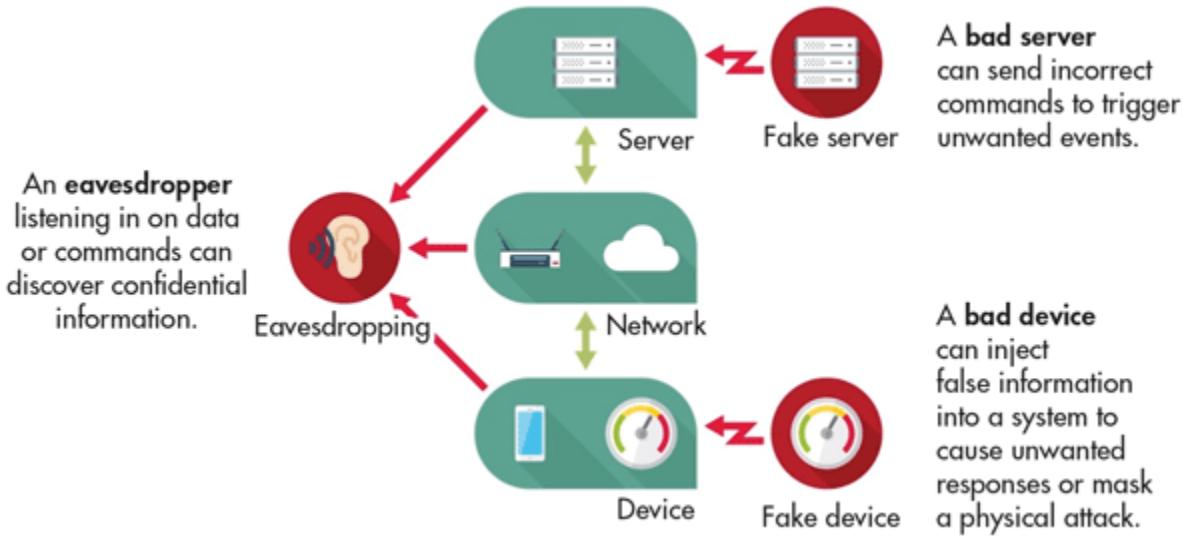
Attack strategies also are consistent. The first step, attack planning, begins by understanding the target devices that allow access to a system. In most cases, the attacker will have physical access to a sample of the actual device they are planning to attack. Using readily available tools, devices can be taken apart to gain access to system memory and electrical processes, and thus create a picture of potential vulnerabilities. Techniques to attack the system are then developed in a controlled setting before being remotely launched against systems in the field.

Given this approach, effective device security requires three elements: solid security processes, strong isolation of the security code and keys, and protection against both remote and physical attacks.

**Security Approaches**

Historically, embedded developers that have implemented device or system security rely on software implementations. The argument in favor of this method is that a design team can access existing and industry-proven security processes and implement them on the existing processor of a device without the need for additional components. However, a solid security process alone doesn't provide strong isolation or protection against physical attacks. Strong security processes also may remain vulnerable because errors appear even in the highest quality software, as documented in the work of Capers Jones.[1]

The trend toward more feature-rich devices with greater connectivity and remote upgradeability further raises the likelihood that errors will be found and exploited by bad actors.




Another approach is to implement an area on-chip where the critical code resides and is executed. This adds some additional cost to the device and fulfills the first two requirements for effective security. It enables good security processes to be implemented and provides isolation of the security code from the main applications.

The drawback to this type of trust-zone implementation is that the specialized area of the chip is only logically separated. It still shares the resources of the full chip, often including the on-chip memory, which creates opportunities for analysis and exploitation from code outside of the trust zone. This type of remote attack on a trust zone has been demonstrated on multiple occasions, including a 2015 Black Hat presentation by Di Shen.[2]

ıddition, this type of security implementation provides no efficient protection against physical attacks and ves device secrets vulnerable to exposure through physical analysis (thplanning stage) and subsequently during the execution stage in some cases. Even with these drawbacks, trust-zone protection may be sufficient in instances where the device isn't easily accessible for a potential attacker to tear down and analyze and when it's been implemented carefully by security experts.

The third option for embedded device security is to introduce a dedicated security processor in the embedded device. This may be designed to operate with both encrypted memory and encrypted code running wholly or in part across the entire range of operations. This type of solution achieves all three critical security elements: solid processes, strong isolation, and physical protection.

Physical separation from the application adds to the strength of protection, because there's no opportunity to observe critical cryptographic information in the clear or mount an attack that leverages the interface between a trust zone and other areas of a system chip. It also simplifies the task of the device developer and system software team. Even if a development team includes experts on security, the programming investment to implement cryptographic functions on a general-purpose MCU is high. Since any coding project introduces the chance of errors that might be exploited by attackers, the "drop-in" solution is much less complex.

The pressure to economize on component cost and minimize device size is often cited as a reason for integrating security onto a device CPU. With dedicated security controllers available in form factors suited to electronic payment cards, the cost, size, and even power requirements of a dedicated chip frequently aren't significant from a system viewpoint. It's important for developers to consider the overall risk and the impact on an organization's business model if a vulnerable system is successfully attacked in the field.

References:

1. Capers Jones, "Software Defect Origins and Removal Methods," December 2012.

2. Di Shen, "Attacking your "Trusted Core," Exploiting TrustZone on Android."

*Looking for parts? Go to SourceESB.*

**Source URL:** http://electronicdesign.com/iot/what-s-difference-between-software-and-hardware-embedded-device-security