Security and the embedded system

 In February, when Kevin Butler, the marketing facade of Sony, retweeted, “@TheKevinButler Lemme guess... you sank my Battleship? RT @exiva: 46 DC … C2 Come at me, @TheKevinButler,” Sony unwittingly published its PlayStation 3 master signing key for the world to see and use. Two months later, we learned developer James Laird reverse-engineered the Apple AirPort Express private key by dumping the read-only memory (ROM) contents. These are just two examples in a long history of high-profile security breaches, causing embarrassment and expense to many. Unfortunately, this begs the question: “If corporate giants like Apple and Sony can’t protect their embedded systems, how can I?”

Securing an embedded system can seem like a daunting task, but the best way to begin is by clearly understanding the answers to the following fundamental security questions:

• What are you trying to protect?
• Who are you trying to protect against?
• What is the cost if the information you’re trying to protect is compromised?

With in-depth answers to these questions, developers can better tailor security solutions to their system’s specific needs. When determining what to protect, the following questions must be explored:

• What software or data must be kept secret?
• Which communication channels need to be secure?
• Will you allow third party applications?

Hackers are not omnipotent beings; they are intelligent, motivated individuals and groups with ample time and resources. By understanding hackers’ goals, developers can focus resources on protecting their devices from those who wish to cause harm, and mitigate the risk of attack from those who don’t (e.g. homebrew developers). This will not only reduce the cost of the solution, but it will also increase the robustness.

Figure 1 illustrates how to secure an embedded system. This system needs to protect both the measurement of power and communication of those measurements to the operator, and it needs to be protected from anyone seeking free power. If a meter is compromised, this could represent a significant cost for the company. Like many embedded systems, the location at the end users' home or business gives attackers unfettered access to unprotected hardware or software.

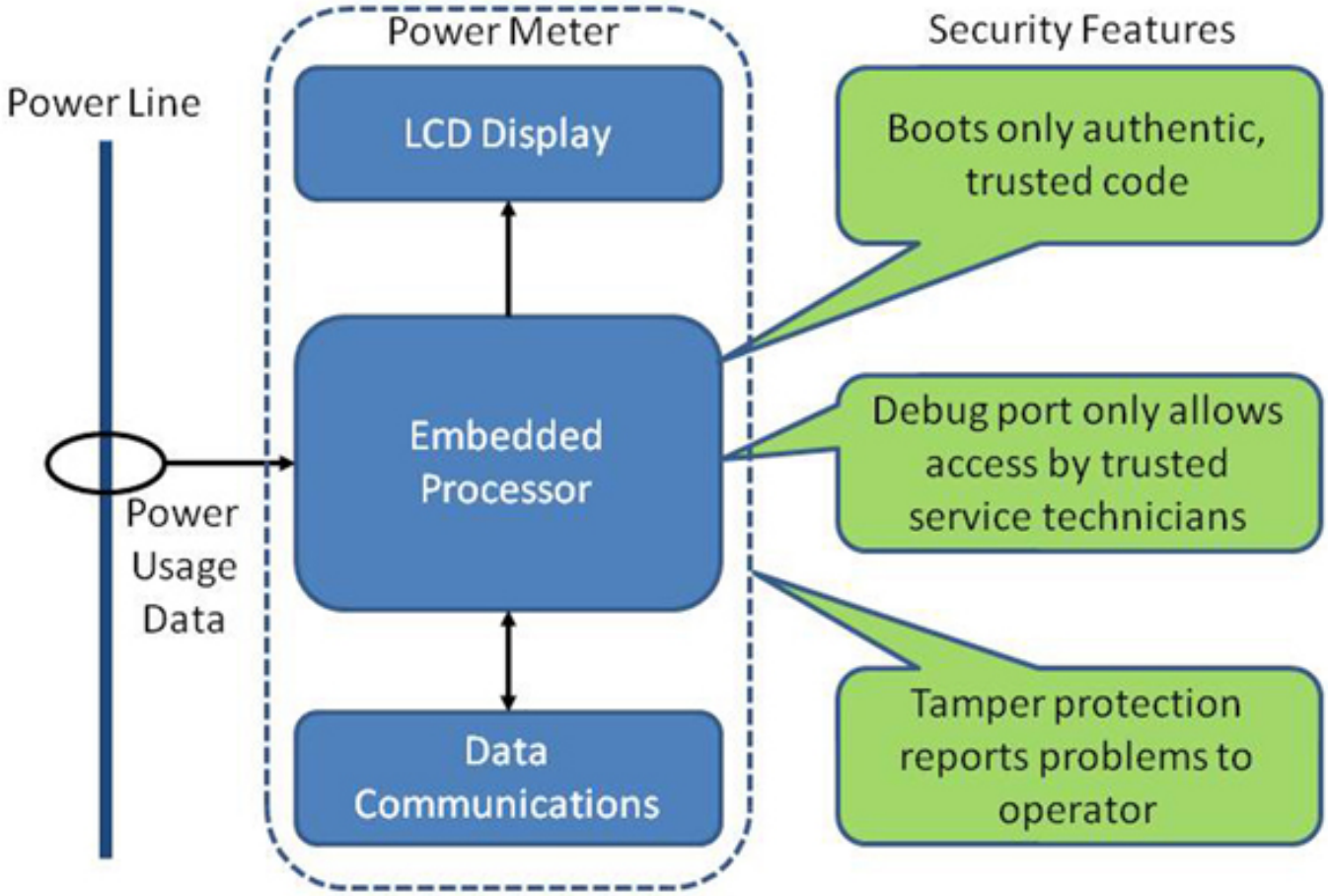

Figure 1: Embedded Security Solutions for Electric Metering

To adequately protect this embedded system, developers might use the following security tools:

1. **Make sure the software running on the meter is trusted and authentic**. Verify that secure boot capabilities start from a hardware-based root-of-trust and use authentication algorithms such as RSA 2048/4096 or elliptic curve cryptography.
2. **Protect the debug port allowing only authorized service technicians access to the meter**. Use a device that allows operators to only open an individual system, not a single global password.
3. **Incorporate tampering protection that reports back to the operator**. Tamper protection should actively monitor the system and decide, based on the event, the appropriate response.

As the list of security breaches grows and technology pushes into critical markets such as medical devices, security has become a requirement for many embedded systems. Fortunately, the ability for developers to protect software and hardware exists and will only improve as offerings from companies become more widespread. By understanding the answers to the fundamental security questions, developers can create robust and secure embedded systems and take charge of their security future.