Secure boot is an important link in the chain of security. It might seem surprising, then, that the secure boot function is not today universally implemented in Linux-based embedded devices due to some assumptions.

Truth or myth?

Five common assumptions about the implementation of secure boot in embedded Linux-based devices

AUTHORS

Iisko Lappalainen
Senior Manager Technical Pre-Sales & Solutions, MontaVista Software, LLC.

Sachin Kaushik
Senior Technical Marketing Specialist, MontaVista Software, LLC.

Contents

Executive Summary

A PC connected to the internet is vulnerable to attack in a multitude of ways, but the authenticity of the operating system that it runs is not in question. If you switch on a PC and the Windows® logo flashes up on the screen, you can be very confident that the computer is running a safe and valid version of the Windows OS.

You have this assurance because of secure boot, a standard authentication function that runs at power-up to authenticate the OS image before it is launched by the CPU. It means that it is almost impossible for rogue OS software to hijack the PC and take over its operation.

Secure boot is an important link in the chain of security for PCs. But embedded devices based on a Linux® operating environment are equally compromised if a rogue Linux image is permitted to run on them. It might seem surprising, then, that the secure boot function is not today universally implemented in Linux-based embedded devices.

Perhaps this is partly because many embedded developers continue to nurse various assumptions about secure boot – assumptions which in some cases are misleading or even downright false. In the course of MontaVista's many conversations about security with embedded developers, five of these preconceived ideas about secure boot recur time and again.

Developers commonly question the need for secure boot in devices which are protected by firewalls, encrypted network communication and physical access controls.

Some developers who admit the potential risk that the OS could be compromised will argue that the risk is so small, and the cost of implementing secure boot so great, that the effort is not worthwhile.

The Linux community also often questions the role of Microsoft in providing the security keys which underlie secure boot implementations on x86 hardware. To some, this puts in doubt the open-source nature of any Linux-based system implementation that includes a standard secure boot function.

Others worry about the effect that secure boot will have on the performance of the host system.

The last assumption that is frequently aired is that, once a development project has been started without provision for secure boot, it is too late to add it later.

Now a detailed examination of the five assumptions listed above is the subject of this article, in which the risks and costs associated with a breach of boot security are described, as well as the resources and tools available to help with secure boot implementation.

Among those resources are the security technology experts at MontaVista, who are on hand to help if you have questions about secure boot.

Solution Overview

Secure boot is a standard function first developed for the Windows personal computing platform, and now broadly supported by the computer industry. Its purpose is to guarantee that the software, including the operating system image, loaded on to a computer's hardware at start-up is the software that the device manufacturer intended it to run, without any modifications other than properly authorized software updates.

Secure boot is an extremely secure authentication process for the operating system image and other software components. It closes a potential entry point for malicious attack, such as a rogue or infected OS. The malware will be prevented from running, eliminating the possibility that it could hijack the device's hardware.

In desktop computing, secure boot forms a crucial element of the security architecture used to repel the threat to networked computers from hackers, terrorists and cyber-criminals. This threat to a primarily Windows® platform-based ecosystem is familiar.

Among the general population, there is much less awareness of the threat to embedded computing devices on which a Linux® operating system runs. And this has helped to fuel the development of a number of widely-held assumptions about the function and implementation of secure boot in an embedded Linux environment.

This article examines the most common of them.

Assumption 1: my embedded system does not need the protection that secure boot affords

The vulnerability of a system of networked workstations and servers is easy to understand. But an embedded system running a version of the Linux OS could be as functionally simple as a sensing appliance providing a stream of measurements to an industrial controller over an Ethernet connection that is behind a firewall.

In a typical industrial or process control set-up, such data streams will be encrypted before transmission from the sensor to the controller. It is common for the developers of this kind of industrial appliance to assume that the security threat is to the data stream, because the network is the main point of vulnerability, rather than to the device, and that secure boot is not therefore necessary at the device level.

But this would be to ignore the threats to the integrity of the device itself, which arise most often from:

Physical tampering under the cover of the device itself, for instance by a rogue employee or external engineering contractor.

A network-borne hacking attack on the device's operating code, aimed at hijacking it.

Secure boot implemented in hardware eliminates such threats. At start-up, firmware hosted in a hardware root of trust (such as an Intel® Trusted Platform Module (TPM) or Arm® TrustZone technology) checks the secret signature of each piece of the boot software and the operating system (see Figure 1). If the signatures are valid, the device boots, and the firmware gives control to the operating system.

Since a hacker or other threat cannot know the signature, any rogue OS or malware will be denied permission to run.

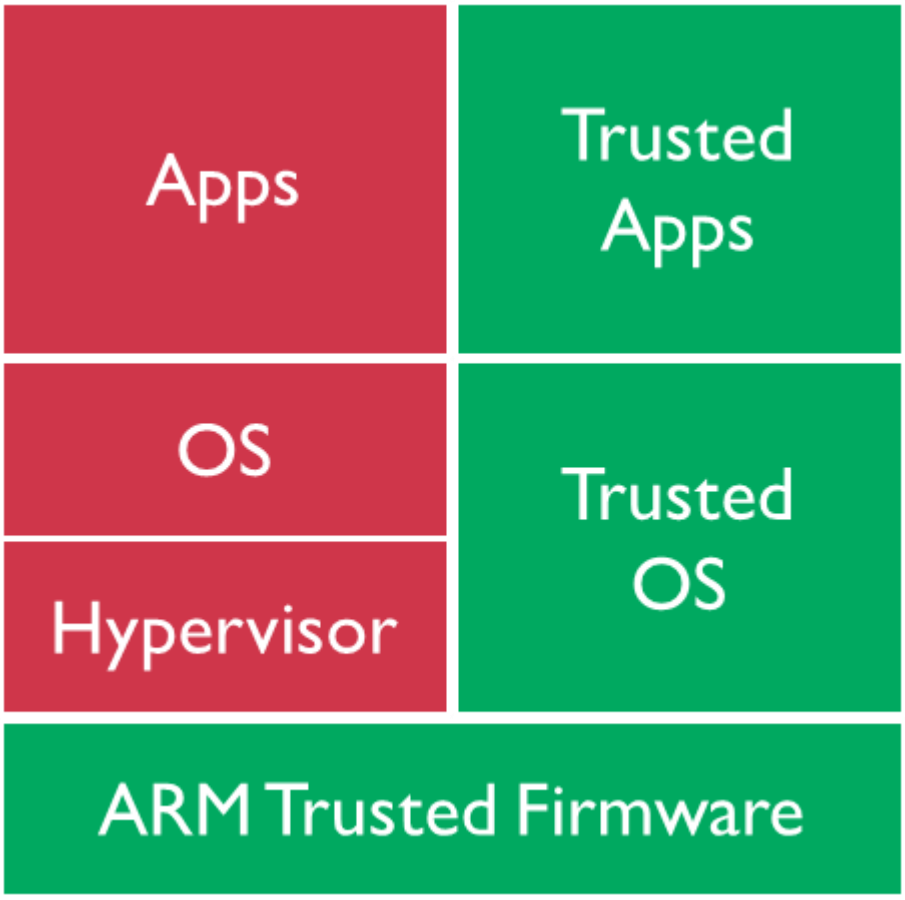

Fig. 1: Arm provides an open-source reference implementation of low-level secure world software known as Arm Trusted Firmware. (Image credit: Arm)

Assumption 2: the small risk to my device is not worth the huge effort required to implement secure boot

This assumption depends on three beliefs: that the threat to an embedded device of hijacking its Linux OS is small; that implementing secure boot is arduous; and that the time and cost involved in implementing secure boot is a poor investment. There are good reasons to question all three of these beliefs.

This article has already suggested that there is a real and present threat to every embedded device that uses a Linux platform, from employees of the device manufacturer as much as from remote cyber-criminals.

Even if the risk of attack appears small – and in truth, it is almost impossible to estimate the probability of an attack – it is much easier to quantify the cost of an attack if it were to happen. And that cost can easily rise to frightening levels. The damage extends far beyond the need to repair or replace a customer's compromised equipment. There is the potential liability for losses suffered by a customer because of an attack made via a compromised device. And the potential reputational damage to the device manufacturer's brand is possibly the most costly of all.

So there is a very strong investment case for the cost of eliminating this risk by implementing secure boot.

And third, the implementation of secure boot does not have to be arduous. It is true that security in embedded computing is a specialist field, and the jargon and complex concepts can appear intimidating.

But secure boot itself is a standard technology that is well supported in the x86 and Arm Cortex® processor architectures. For x86 processors, Intel supplies reference design building blocks to device OEMs to accelerate the implementation of secure boot on its processors. Likewise, manufacturers of Arm Cortex-based MPUs support secure boot implementations on their devices.

Within a Linux environment, the task of implementing secure boot is accelerated even more for users of a commercial-grade Linux such as MontaVista, because developers benefit from the use of an OS platform optimized and tested to run on the chosen hardware target, and from detailed guidance from MontaVista's security experts.

Assumption 3: secure boot requires proprietary software

For devices running on a Windows platform, Microsoft provides a dedicated secure boot implementation. And according to the Ubuntu Wiki, 'Most x86 hardware comes from the factory pre-loaded with Microsoft keys. This means we can generally rely on the firmware on these systems to trust binaries that are signed by Microsoft, and

the Linux community heavily relies on this assumption for Secure Boot to work. This is the same process used by Red Hat and SUSE, for instance.' (Reference: wiki.ubuntu.com/UEFI/SecureBoot).

But here, Microsoft is operating as a certified source of keys: this does not compromise the open-source character of the Linux environment, including of secure boot implementations in Linux.

In fact, open-source secure boot solutions have been tested across a far greater number and a wider variety of embedded devices than Microsoft secure boot firmware has been: the Linux community's experience suggests that the open source solutions have been secure and stable.

Assumption 4: implementing secure boot will impair the performance of my device

Adding any software process to a system entails the use of processor cycles and memory capacity. In the case of secure boot, however, these processes only run at start-up, and so have no effect on normal operation.

In addition, most secure boot implementations normally use a dedicated hardware resource – an Intel TPM or Arm TrustZone – to provide the root of trust (see Figure 2). When this hardware is used for secure key storage and to run authentication firmware, the overhead on the host CPU of running secure boot processes is almost zero.

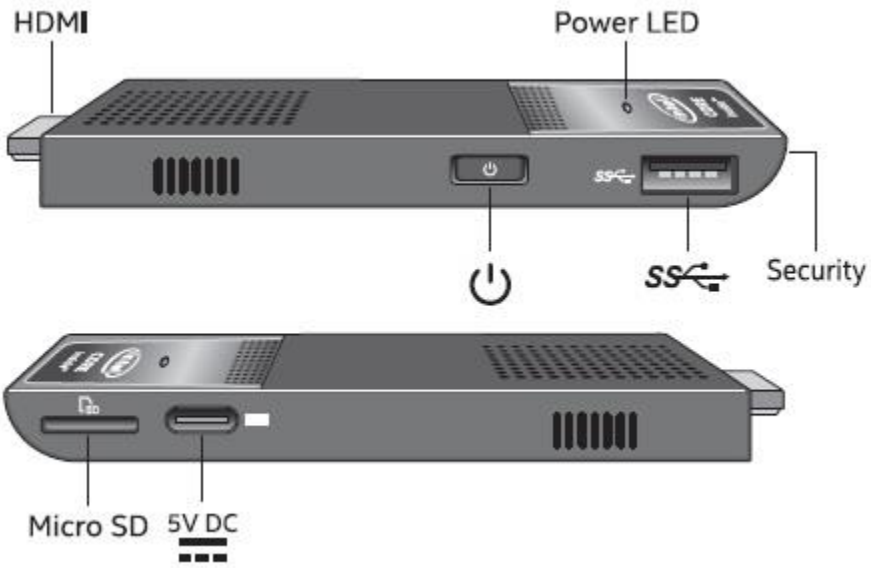

Fig. 2: the Intel Compute Stick STK2mv64CC, an x86 device which includes a discrete Intel TPM and supports hardware-based secure boot

Assumption 5: secure boot has to be designed in from the beginning of a project

In an ideal world, secure boot is specified as a functional requirement from the outset of a product development project, allowing for the integration of root of trust hardware into the system design.

But if dedicated security hardware is not available, secure boot can be implemented in software. The main caveat for this approach is that it does not protect against the threat of physical tampering with a device's hardware components.

In a software implementation of secure boot, MontaVista users benefit from reference design examples and expert guidance from the company's security experts on porting the examples to any supported hardware target.

Surmounting the barriers to successful secure boot implementation

The lesson from MontaVista's work with its user base is that secure boot is an essential part of any comprehensive device-level security strategy, and that its implementation is far quicker and easier than most developers initially fear it will be.

The ability to implement secure boot alongside other security functions in embedded devices is only going to become more important in future as new smart and connected models of device operation proliferate. To take just one example: personal health monitoring equipment, directly connected to the internet, can enable physicians to continuously monitor patients remotely and set alerts to trigger pre-emptive actions, such as prompts to take medication. In this application, the authenticity of the device's software is not only a matter of commercial integrity – it is critical to the safety of the user.

Few OEMs' development teams are large enough to include dedicated security engineering specialists. So MontaVista users benefit from the support and guidance of MontaVista's security experts, not only with secure boot, but more widely to guide OEMs in best practice for key creation and storage, for provision of an end-to-end root of trust, and for configuring a process for securely updating devices in the field.

Of course, the open-source community provides a wealth of resources for integrating a secure boot process into home-grown Linux platforms as well. But a commercial-grade Linux OS provides a guaranteed stable and supported platform for system development, and also provides ready-made software and support for secure boot as well as many other common system functions.

Conclusions

As this list suggests, there are many, widely held reservations about secure boot – yet in the real world, security breaches of connected devices are happening more and more frequently, and with more devastating effect. Of course, security in embedded devices is a topic with many branches, but for Linux users the first line of defense should be the use of an authentic OS image.

MontaVista Software, LLC | 2315 North 1st Street San Jose, CA, 95131 | www.mvista.com

DOC. ID. (MVTP-SecureBoot-051518)