Protect Critical IoT Devices with VxWorks Secure Boot and Secure Loading

By Ka Kay Achacoso
EXECUTIVE SUMMARY

Government reports indicate that hacking attempts on national critical infrastructure continue to rise. As the real-time operating system supporting critical infrastructure equipment, VxWorks® provides essential security features to protect important equipment from serious attackers. For tight execution control, VxWorks secure boot prevents devices from running unauthorized software and loading compromised data. This paper explains the mechanics of VxWorks secure boot and secure loading, including examples of VxWorks making use of hardware security features provided by Intel® architecture boards and NXP i.MX6 boards to ensure that its software security is rooted in hardware. Devices use VxWorks secure boot and secure loading to provide software tamperproofing.

TABLE OF CONTENTS

Executive Summary .......................................................... 2
Introduction ................................................................. 3
Software Tampering .......................................................... 3
Asymmetric Key Pairs ......................................................... 3
Build Time and Runtime Processes ...................................... 4
Establishing Trust ............................................................ 4
Secure Boot on Intel Architecture ....................................... 5
Secure Boot on i.MX6 Boards ............................................. 6
Secure Boot on Other Boards ............................................. 6
User Application Authentication ........................................... 7
VxWorks with Security Profile .......................................... 7
Conclusion ......................................................................... 7
INTRODUCTION

As a provider of the energy infrastructure to a city, or as a financial institution managing customer assets, or as the Defense Department controlling communications equipment, organizations rely on proper operation of their equipment to deliver safe and efficient services to their customers. Some organizations are subject to frequent attempts by attackers to break into their equipment and change its intended operations. An adversary may try to hack into a smart power grid to effect large power surges, eventually causing a city’s electricity to shut down. A thief might try to run snooper applications on a retail payment system in an attempt to steal user PINs.

VxWorks runs as the underlying real-time operating system for critical devices. VxWorks comes with security options to protect against several types of attacks, including attacks that make the device do something unintended, attacks that steal data and intellectual property from the device, and denial of service attacks. VxWorks secure boot addresses the first type of attack by preventing the device from running an attacker’s software or compromised software.

This paper goes into a technical deep dive about VxWorks secure boot and secure loading and gives details on hardware root-of-trust examples for Intel architecture boards and NXP i.MX6 boards.

SOFTWARE TAMPERING

Devices operate on properly tested software from the manufacturer. If a device is tricked into running malicious software, an attacker can drastically change the behavior of the device. For protection against this kind of attack, devices need to differentiate between intended software and unintended software. This ability to differentiate between good and bad is achieved through establishing trust in the software loaded by the device.

The software loading stages during VxWorks boot-up are shown in Figure 1. At boot-up, the device firmware loads a boot loader. The boot loader loads the VxWorks operating system. The VxWorks operating system loads software libraries and runs user applications. Without secure mechanisms in place, a malicious person with sufficiently sophisticated technology can tamper with the boot loader, switch out the operating system, and inject snooper applications onto the device, causing the device's execution to veer off into unintended operations.

To prevent a device from executing unwanted software, at each step the device must first check that the software image is trusted before loading it. This check is called authentication. Software authentication determines two facts: First, the software comes from a trusted source. This means the software comes from the device vendor and not from a hacker. Second, the software is still in its original form, and not modified through, for example, a man-in-the-middle attack.

In Figure 1, the diagram of the VxWorks application boot-up process shows three stages of loading. Security requires authentication to occur in all three stages. At each stage, a setup process and a runtime process ensure proper authentication operations.

ASYMMETRIC KEY PAIRS

The setup starts with an asymmetric public/private key pair generated using an asymmetric cryptographic algorithm, such as the Rivest-Shamir-Adleman (RSA) algorithm. The unique property of asymmetric key pairs is that only a private key can generate a unique signature that will pass authentication when checked with its corresponding public key.

The setup process and runtime process ensure proper authentication operations.
The mechanics of signing and authenticating using asymmetric key pairs works as shown in Figure 2. An image can be any block of bits and bytes. In secure boot, the image can be a boot loader or the VxWorks OS. In secure loading, the image can be user applications, libraries, or software modules.

When an image passes through a hashing algorithm such as SHA-256, the result is a shorter 32-byte sequence of data that uniquely identifies the image. This image hash is reproducible, so whether the hashing process runs on a big server or on a small device, the original image will always hash into the exact same image hash.

The nature of an asymmetric key pair is that a block of data encrypted with one half of the key pair can only be decrypted into its original form using the other half of the key pair. Applying this concept to the signature process, an image hash encrypted with a private key produces an image signature. The image signature can only be decrypted into its image hash with the corresponding public key. Thus the image signature uniquely identifies the public/private key pair and the image hash, which uniquely identifies the image, effectively providing both integrity and authenticity checks.

BUILD TIME AND RUNTIME PROCESSES

At build time, a signing tool uses the private key to generate a signature of the image and public key in the form of a certificate. This signature is unique to the private key and to the image, and it is added to the image and public key. The private key is kept secret by the device vendor and is never loaded onto the device. Only a designated person holding the private key can generate valid signatures.

At runtime, the authentication process starts when the loader receives a signed image. From the signed image, the loader extracts the original image, the signature, and the public key. The loader first validates that the public key is a trusted key. Public key validation is a gate in the authentication process. The validation process can take many forms, depending on hardware. For example, there can be a list of trusted keys stored in a static list on-board, or the device hardware may have signatures of trusted keys in unmodifiable memory. The device can also verify the public key with a third party, especially later in the boot process when networks are available.

Once the public key is validated, it verifies two facts: First, the image's signature was generated by the corresponding private key. Second, the image is still the exact same unmodified image. A missing signature, an image signed with a different key, or an image that has been tampered with causes an authentication failure.

The mechanics of authenticating an image by inspecting its signature goes back to Figure 2 of the previous section. The authentication process involves two operations: hashing of the original image to obtain the image hash, and decrypting the signature with the public key to obtain the image hash. The device ends up with two resulting image hashes from the two operations. If the two results match, the signed image passes authentication.

ESTABLISHING TRUST

The protection against loading malicious software involves signing with a private key and authenticating with its corresponding public key. Trust in the key pair depends heavily on two conditions:

- The private key is held secret by the device vendor.
- The trusted public keys list on the device cannot change.
The first condition for trust in a key pair relies on tightly controlled access to all private keys. An attacker who gains access to private keys has the power to generate boot loaders, VxWorks images, and user applications that pass authentication. Secure device vendor organizations must have key management processes in place that restrict the signing of software binaries to certain authorized users, and to track the use of the private keys. Many organizations involve the IT department to implement key management security processes, sometimes with the help of software key management systems (KMS) for the administration and guarded distribution of keys, which may be physically located in locked rooms with stringent access requirements.

The second condition for trust in a key pair relies on the assurance that at runtime, the device uses only the designated public key for authentication. The check for a key in the trusted keys list is called key validation. An attacker who can change the public key information on the device can replace the trusted public key with one of his own and cause untrusted software to pass authentication. The device must prevent unauthorized changes to the trusted keys list.

Secure boot requires that key validation establish root of trust at the hardware level. VxWorks uses hardware to ensure that only authorized changes can be made to the trusted keys list. Implementation of key validation is hardware dependent. The next two sections show how secure boot works on two of the supported hardware platforms.

**SECURE BOOT ON INTEL ARCHITECTURE**

On Intel architecture, VxWorks takes advantage of the UEFI secure boot feature. At the root of trust is the platform key (PK), which is an asymmetric public/private key pair. The device vendor puts the public PK onto the board via the BIOS. On the device, the public PK is password protected and a change requires the physical presence of the board. The device owner keeps the private PK a secret.

The device vendor generates a key exchange key (KEK) pair, wraps the public key in a certificate, and requests the device owner to sign the certificate using the private PK. The device owner uses the BIOS to put the PK-signed KEK certificate onto the device. UEFI secure boot allows only PK-signed KEK certificates to be loaded. Thus, the PK validates the KEK.

The device vendor generates a list of loader signing keys, which are used to sign any images to be loaded. These images can be the UEFI loader, the VxWorks kernel image, and the VxWorks user applications. The list of loader signing keys is called the white list database. The device owner signs the white list database with the KEK private key and uses the BIOS to put the white list database onto the device. UEFI secure boot allows only a KEK-signed white list database to be loaded. Thus, the KEK validates the white list database.

This chain of trust from PK to KEK to all the keys in the white list database is shown in Figure 5.

![Figure 4. Public and private key management](image)

**Figure 4. Public and private key management**

![Figure 5. VxWorks secure boot using UEFI loader](image)

**Figure 5. VxWorks secure boot using UEFI loader**
The device vendor signs the UEFI loader with one of the private loader signing keys and attaches the loader signing public key to the UEFI loader image. At runtime, before the UEFI firmware loads the UEFI loader, the UEFI firmware checks first that the loader signing public key on the image matches one in the white list database before using the public key to authenticate the UEFI loader image.

Once the UEFI loader passes authentication and loads successfully, it uses the same process to verify that the attached public loader signing key matches one in the white list database before continuing with the authentication process. The same authentication steps run when VxWorks loads user applications.

More advanced features of the UEFI secure boot, such as blacklist database and alternative white list database contents, are not discussed here. You can learn more about UEFI secure boot from [http://www.uefi.org/specifications](http://www.uefi.org/specifications).

With the VxWorks development environment, the process to set up secure boot is greatly accelerated. A VxWorks project can build the secure UEFI loader. The platform includes tools to sign the UEFI loader, VxWorks, and user applications. To get started, a development set of PK, KEK, and loader signing key pairs is automatically generated. In production, separately generated production keys easily replace the development keys in the setup and signing process.

**SECURE BOOT ON I.MX6 BOARDS**

The secure boot example for i.MX6 boards takes advantage of the fast authentication feature of the High Assurance Boot (HAB) architecture on the boards. In this case, the boot loader is U-boot. Firmware on the i.MX6 board authenticates U-boot, U-boot authenticates VxWorks, and VxWorks authenticates user applications.

The i.MX6 secure boot setup starts with the NXP code signing tool (CST), generating four storage root keys (SRK) in an SRK table. The keys in the SRK table are hashed together, and this master hash is burned permanently onto an eFuse on the i.MX6 board. Every time the board boots up, the i.MX6 firmware expects U-boot to carry the SRK table in its signature. The HAB architecture validates the SRK table using the eFuse master hash value.

Once the SRK table is validated, any of the SRKs in the table can be used to authenticate U-boot. HAB makes use of a command sequence file (CSF) section attached to U-boot to load the SRK table and start the fast authentication process. Both the main image (U-boot or VxWorks) and the CSF section have their own signatures validated by an SRK. HAB also has a key revocation mechanism. See NXP’s Application Note AN4581 titled “Secure Boot on i.MX50, i.MX53, and i.MX 6 Series using HABv4” for details.

The same SRK table validation happens when U-boot loads VxWorks. The signed VxWorks image carries a signed CSF section and the SRK table. The SRK table is validated against the master hash in the eFuse. One of the SRKs authenticates the VxWorks image. If authentication passes, U-boot loads VxWorks.

**SECURE BOOT ON OTHER BOARDS**

Secure boot mechanisms depend on hardware implementations. VxWorks also supports a number of other boards not described here. If you have access to Wind River Support Network available with support for the VxWorks Platform, you can see other secure boot examples in the VxWorks Security Profile section of the Wind River Knowledge Library.
Once VxWorks boots via the secure boot process, the operating system can load user applications, which take the form of kernel-mode runtime modules, user-mode executable applications, or user-mode shared libraries. As dynamically loaded applications, these applications must also be authenticated.

VxWorks secure loading provides the authentication mechanism for VxWorks-loaded user applications. Secure loading relies on the same two security conditions governing a public/private key pair. Runtime public key validation can be set up to use the same hardware-rooted trust mechanism as secure boot.

Both VxWorks secure boot and secure loading ensure authorized software execution but do not hide software and data from users. All image content, signatures, and public key certificates are visible to any snooper smart enough to extract its content. If content is left unprotected, an adversary can steal intellectual property or use the content to exploit vulnerabilities in the device. VxWorks achieves protection against data theft using automatic encryption and decryption mechanisms. Data encryption and decryption is a topic for another paper.

**VXWORKS WITH SECURITY PROFILE**

VxWorks provides safe, secure, and reliable software solutions for IoT devices. Security Profile for VxWorks adds additional security features for those devices that must meet stringent security requirements. VxWorks secure boot and secure loading are key components of Security Profile.

**CONCLUSION**

Equipment manufacturers can thwart software attacks on their products by securing their devices, starting from the earliest boot-up stages and moving through to all software loading processes. For devices that receive software upgrades and updates, authentication of any new software is essential. VxWorks secure boot and secure loading in the VxWorks Security Profile both address software authentication to ensure that only authorized software runs on VxWorks devices. When security matters, rely on VxWorks.