Secure Firmware Update in Schneider Electric IOT-enabled offers

Jean-Philippe FASSINO
Cyber-Security Architect
Schneider Electric
Grenoble, France
jean-philippe.fassino@schneider-electric.com

El Ayachi MOKTAD
Embedded Platforms Architect
Schneider Electric
Andover, USA
el-ayachi.moktad@schneider-electric.com

Jean-Michel BRUN
Global Cyber Security Architect
Schneider Electric
Grenoble, France
jean-michel.brun@schneider-electric.com

***Abstract*— In this paper, we describe the design of the Secure Firmware Update integrated in Schneider Electric embedded products for maintenance and sustainability purposes. Furthermore, we present the associated end to end security process, including a focus on asset management services in our IOT-enabled offers.**

Keywords*— Firmware signing, Public Key Infrastructure, Firmware update, Cloud, IoT, asset management.

I. Introduction

Schneider Electric commercializes products whose lifetime can reach 30 years. These products and systems could be used in critical infrastructures (smart grid, oil & gas, water supply, …) or in industrial production installations (e.g production shutdown has a direct impact on our customer revenues). They require a high level of security and availability which complexes firmware update management.

To prevent these products from becoming obsolete sooner than expected due to evolution in its environment, firmware update is a key feature for sustainability of the product. It's also a key feature to fix vulnerabilities and bugs. It's also useful to enhance firmware with future industry evolutions, like alignment with future IoT protocols or more recently switch to SHA2 following deprecation of SHA1.

On top of that, firmware update mechanism is a great point of attack for hackers to substitute the genuine firmware by a malicious one (e.g. the famous Jeep Hack (1) (2)). Cyber security industrial standards exist: IEC62443 for IACS (Industrial & Automation Control System), IEC 62351 (Security for smart grid, Energy Control Automation), country regulations for the critical infrastructure operators (Military programming law in France voted in December 2013, Critical infrastructure law voted in Germany in July 2015, etc.). Consequently, firmware update service must be strongly secured and require a high security process.

We describe in this paper the secure firmware update service provided by Schneider Electric to its customers. This service is based on: firmware authentication and integrity, end-to-end security process, a portable firmware update and asset management services for IOT-enabled offers.

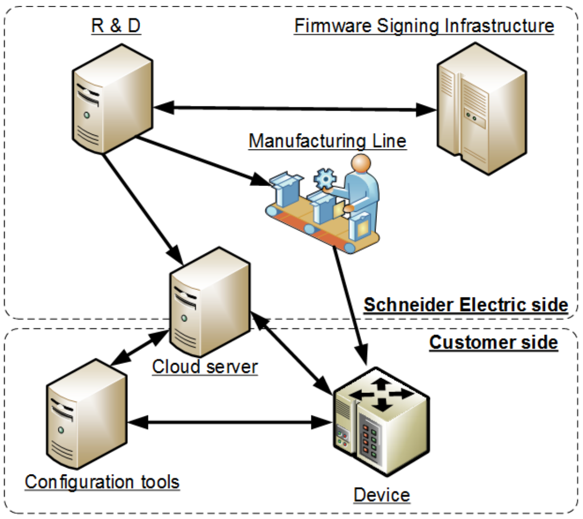

Figure 1. Entities involved in secure firmware update

II. Entities involved in firmware update

Figure 1 shows the entities involved in firmware update and the communication flow where the firmware transit.

Firmware is developed by R&D team. When validated, the firmware is signed by the firmware signing infrastructure and delivered to the manufacturing line to produce new products and to a Schneider cloud server to deliver update to products installed on customer side.

For IOT-enabled offers, the update could be done automatically depending on the security policy and need of the customer installation. For on-premises offer, the firmware should be download on the device configuration tools which will update the firmware on the device through local connectivity (USB, ZigBee, …).

When the device receives a new firmware, it verifies it, performs an update of its flash and reboot.

III. End-to-end security

As we show in introduction, firmware update is a great point of attack for hackers. It's especially true when the product is not managed remotely (e.g the firmware transit through many entities). To mitigate these threats, Schneider

Electric signs the firmware of its products and reply on an end to end security process to verify this signature.

A. *Signing the firmware*

The objective of signing the firmware is to ensure the integrity and authentication of embedded firmware in products. For sustainability purpose, Schneider Electric has made the choice to rely signature mechanism on RFC5652 PKCS#7 (3) and RFC5280 X.509 certificate standards (4):

- The firmware signature consists to: compute the hash of the firmware, encapsulate it in a PKCS#7 and sign this PKCS#7 with the private key associated with the signing certificate of the product.
- The firmware authentication consists to: compute the hash of the firmware, compare it with the hash contains in the PKCS#7 and verify the signature of the PKCS#7 with the public key contains in the public certificate.
  For machine with an Internet connectivity, the verification of the validity of the signing certificate could be performed to detect potential compromising of the signing certificate. This verification is based on CRL published by Schneider Electric security server.

B. *Securing entities involved in firmware update*

To detect as soon as possible a security issue, all entities has been secured.

1) *R&D team*

To mitigate attack at development or build time, like inserting vulnerabilities or backdoors in the firmware, a Security Development Lifecycle (SDL) is deploy in R&D team, including secure coding rules and code review. The security compliance is tested by Static Code Analysis tools.

2) *Firmware Signing Infrastructure*

The firmware signing infrastructure provide the PKI used to sign the firmware image.

In case of detection of the compromising of a signing key (customer feedback, log analysis, …), the associate certificate will be revoked and a new CRL publish on Schneider Electric security server.

3) *Manufacturing Line*

The risk of commercializing a device containing a malicious or modified firmware exists, e.g. an attack on the manufacturing line. To mitigate this risk, the manufacturing line authenticates the firmware before flashing the product and redo the authentication on the flash content.

4) *Configuration tools*

Configuration tools authenticate the firmware before sending it to the product. Since it may have Internet connectivity, the signing certificate is also verified. This mitigates the risk of updating installed devices with a malicious firmware; by substituting the genuine firmware before it reaches the configuration tools (attack on Schneider server, man in the middle, etc.).

5) *Device during update*

When receiving a new firmware, the device authenticates it. This mitigates the risk of updating installed devices with a malicious firmware; by substituting the genuine firmware before it reaches the device (attack on the configuration tools, configuration tools spoofing, etc.).

6) *Secure boot*

For device supporting secure boot, firmware loaded from the flash is authenticates at boot time. This mitigates the risk of hardware attacks on the content of the external flash memory (e.g. flash replacement or alteration) and of software vulnerabilities (e.g. flash patching through code injection attacks).

C. *End to end security process*

The verification is then done at every steps by an end to end secure process.

R&D checks the firmware signing before sending to the Manufacturing Line which checks again before device production.

On customer site, a verification of the firmware signing is done by configuration tools before starting the Firmware update on device. A second verification is done at the device level. In case of compromised firmware, malicious firmware will be rejected.

Schneider Electric provides an end to end security process to guarantee its firmware genuineness, taking also into account the availability requirements that can be found in Industrial critical environment.

IV. FIRMWARE UPDATE

Schneider offer is heterogonous; it’s composed by devices with different constraints regarding available memory, safety requirements and real time constraints.

On the other hand, firmware update can apply to the whole firmware (operating system + application), application only or only part of the application. The hardware itself can be upgraded thanks to FPGA fabric present in some devices.

A. *A flexible firmware update implementation*

To ease deployment of secure firmware upgrade across this large offer, we developed a highly portable (across different hardware and software platforms) and modular implementation offering the basic functionalities (secure package management is platform agnostic) while platform specific extension can be developed by the product team.

B. *Integration in multiples architectures*

1) *High-end vs constrained memory devices*

Industrial high-end devices (with MB of memory) includes industrial controllers (PLCs), building controllers, circuit breakers, communication gateways and others. With such devices, the full firmware package is loaded into main application memory before being processed (signature authentication, parsing and flashing).

For devices with low amount of memory (IOT sensors, …) firmware update can run in streamed mode where the package is pre-authenticated prior to firmware binaries loading by checking the signature of package metadata that contains binaries hash. firmware binaries are then directly flashed while received block by block. Once all binaries are loaded and flashed a second authentication round is executed.

2) *Firmware update flow*

The firmware update module can run in different ways depending on the product needs and constraints:

- Run firmware update in the main application: All the update process is executed while the application is running. The device is simply rebooted when update process is terminated.
  An optional flip flop mechanism ensures compatibility with XIP execute in place device or recovery in case of power fail during the upgrade.
  This is the simplest implementation. It's also the most suitable for connected devices as the firmware update has access to all device connectivity permitting convenient firmware update methods while ensuring high availability of the application as it continue running while the device is being updated.
- Integrate firmware update in the boot loader: It reduces the size of required memory but limits firmware update methods to the connectivity available in the boot, often a basic usb connection to a PC tool.
- Run firmware update in a dedicated mode: When firmware update is requested device is rebooted in firmware update mode. This allow more advanced firmware loading methods while preserving memory usage as firmware update doesn't run at the same time as the application. However, this dedicated firmware cannot be updated.
- A mix of previous, run firmware update in the application to take advantage of connectivity and run the flashing part in the boot or firmware update mode to preserve memory usage.

C. *System upgrade*

In some configuration only one device has necessary connectivity to do a remote firmware update while all other devices have only local connection to this main device. For these cases the firmware update can process a package with binaries for other devices and is capable of routing those to the right devices using router extension (See figure 2).

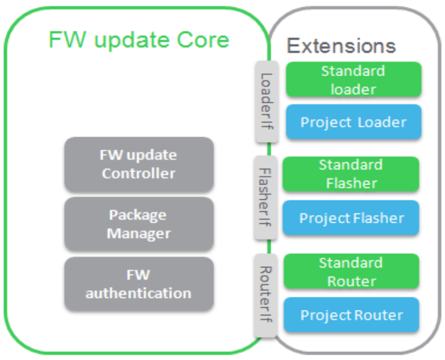

Figure 2 : Firmware update modular architecture

The main device acts then like a PC configuration tool from the sub devices point of view; the main device is typically a large memory device and depending on the sub device capabilities the download can be cached or streamed.

In all cases firmware package signature is authenticated by the main device before initiating the transfer and by the sub device once the package is received.

V. Firmware update & Asset Management in IOT-enabled offers

Schneider Electric IOT-enabled offers deliver analytics and optimization services for energy efficiency and asset performance management for our customer installation.

Schneider Electric IOT-enabled offers have the capability to remotely manage the industrial IoT devices on the customer site and can then provide asset management service on installed assets. It gives the possibility to:

- Update the firmware of IoT devices through the cloud connection, fitting the high level of security and availability requirement of the customer installation.
  Even if the connection is secured, this is useless for this use-case since the firmware update was secure by itself.
- Check the end of validity of certificates and thus of the firmware signature. This is particularly useful for device with long lifetime.
- Audit the installation to detect attack and potential compromising on installed device as soon as possible.
- Revoke compromised certificate and inform our customer on this security alert.
- Propose a plan to manage firmware update with new signing certificate. This firmware update can be automatically managed on non-compromised installation. device. On compromised installation, Schneider Electric propose expertise service and help to impacted customer to manage the incident.

The asset management service in IoT-enabled offers is an additional step of the end to end security process to guarantee the Schneider Electric firmware genuineness.

VI. References

1. **Kaspersky.** Black Hat USA 2015: The full story of how that Jeep was hacked. [Online] August 6, 2015. https://blog.kaspersky.com/blackhat-jeep-cherokee-hack-explained/9493/.

2. **Miller , Charlie and Valasek , Chris.** Remote Exploitation of an Unaltered Passenger Vehicle. [Online] August 10, 2015. http://illmatics.com/Remote Car Hacking.pdf.

3. **RFC-5652.** Cryptographic Message Syntax (CMS). *IETF.* [Online] September 2009. http://tools.ietf.org/html/rfc5652.

4. **RFC-5280.** Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile. *IETF.* [Online] May 2008. https://tools.ietf.org/html/rfc5280.