Securing Software Updates for Trains

Tatiana Galibus

UCLouvain, Crypto group, Louvain-la-Neuve, Belgium
tatiana.galibus@uclouvain.be

Abstract. We propose the secure procedure for the automated railway update and maintenance. The proposed procedure is derived from the Uptane update framework. Testing and validation phase, additional manual approval procedure and update progress control are integrated into the Uptane framework in order to conform to the railway safety requirements and norms. The possible metadata and repository customization is proposed and specific railway update attacks are discussed.

Keywords: The Update Framework · Railway transportation system · resilient security · Uptane framework · secure repository · secure update.

1 Introduction

Railway transportation security relies on correct, safe and secure maintenance. That's why configuration management is the critical aspect of railway security. Configuration management and update are important processes in railway transportation as they assure the correct and uninterrupted functioning of the signaling devices. This process is part of any standard railway system ([1], [2]) but as the role of complex digital components grows, the manual maintenance of thousands of trackside digital devices becomes more and more difficult and costly. The automobile industry has already developed robust and secure procedures for such automatic update and maintenance [5], [6], validated in the industrial environment. Railway industry has not yet undergone such transformation but with all the complexity of infrastructure and number of devices to maintain, an automatic solution is inevitably in demand. The paper is focused on the security of the automation of trackside device update, which can deliver significant benefits for the railway industry. Additionally, the railway applications should conform to safety requirements [3], [4], i.e. SIL3 safety constraints are highly recommended for the server part and communication board on a device [3] and SIL4 constraints for the signalling board on a device [4].

In order to conform to these safety requirements, we adapt the update model from the automobile transport industry where the automated over-the-air (OTA) updates are a commonly accepted standard. The security of ground vehicle updates lies on the efficient implementation of TUF (The Update Framework) [7]. TUF is a framework that supports the compromise-resilient security of software updates. An adaptation of TUF, Uptane is a de facto standard for automobile updates: in July 2018 the IEEE/ISTO Federation began formally standardizing

Uptane under a non-profit consortium called the Uptane Alliance [8]. Uptane is currently is part of Airbiquity [6], the leading automatic update and orchestration platform for more than 40 million ground vehicles. Airbiquity deploys updates for such companies as Ford, Renault, Bosch, Toyota among others. In this paper, we customize the Uptane procedures i.e. the update workflow and metadata verification process to the specifics of the railway industry. We demonstrate how these adaptations protect from the specific attacks on the trackside device updates.

Our main contribution is adaptation of TUF/Uptane key principles for the railway updates. The specifics of the railway system require several enhancements such as:

1. Unique *target.json* files for each trackside device in order to avoid the installation of the wrong software.
2. Keeping track of successful updates in order to manage the uni-cast update process.
3. Full verification on each device i.e. verifying target metadata on each device against the target metadata approved by the installation manager.

2 The railway update workflow

We shall consider the generic railway update workflow in order to demonstrate the specifics and complexity of the train and trackside update process [1]. The railway company should support the development of its products at the 3 life-cycle stages : generic, specific and installation. Unlike Uptane model, frequently, the chain of component suppliers is completely within the control of the company. In other words, OEM has greater responsibility on the software and firmware code as well as configuration files which demands stricter security requirements. The railway company should assure the integrity and authenticity of all components developed by all subcontractors on all levels and all sites whereas in the automobile industry the components might originate from the chain of independent suppliers and a single signature of a manufacturer is enough to verify the authenticity. The update framework should support the security and safety of each component, baseline and the whole update package.

2.1 Level 1: Generic product/subsystem design

A generic trackside subsystem is composed of several components, each of them being composed of several LRU (line-replaceable unit). Each LRU component binary may include software, a configuration or data preparation file and firmware and can be used in multiple projects. The generic design tests can be delivered

[1] The industrial partner of our research is an important stakeholder in railway transportation. We can not specify the details of update process due to project specifics. We outline the generic norms and procedures, based on railway specification documents such as [3], [4] and discussion with our partners.

along with the product in order to support its testing and validation [3]. All generic products/tools are maintained by independent developer teams located on multiple sites and coordinated by the design team. The design team is in charge of the integration and validation of generic products. It delivers a baseline of all components to the customer. The generic baselines can be stored in the generic repository of railway environment. Further, they can be used by the project design.

2.2 Level 2: Project design

The project design team is responsible for a dedicated location on the rail (i.e.: a station or a city) specified by the corresponding configuration. In order to initiate a project, one generic baseline can be chosen in order to maintain the compatibility of the component versions. It can be upgraded later. The generic repository can keep track of the baselines in use or permit the customers to manage it. Additionally, it is sometimes necessary to design specific products or features. In this case, the specific developer group is in charge of design, tests and validation of project specific products. A specific product can be tested and stored in a dedicated repository or together with the generic ones depending on the complexity of the project. The project binaries can be stored on the project repository of railway environments in order to be downloaded by the independent installation teams.

2.3 Level 3: Project installation and maintenance

Once the update is ready for the installation, railway safety protocols [3], [4] require CB (Control Board) to be organized including the customer authorities in order to initiate the download of the new version on the target devices. The new version once approved needs to be securely transferred to all devices and be validated by all of them. As railway safety protocols [4] require additional installation tests and validations for the new version, it should first be tested during the limited time slots. Once the tests and safety activities are finished successfully, a second CB has to be organized in order to definitively switch to a new version. The project installation teams coordinated by an installation team manager are in charge of installation, validation and delivery of the installation on the device.

2.4 Security procedures in railway update

The architecture of the railway update framework is based on a hierarchy of community repositories and requires security guarantees at each level. The community repositories have suffered the series of devastating attacks in the last 2 decades [9] and developed robust and transparent security mechanisms based on secure package managers and metadata support [10]. On the contrary, to our knowledge, no railway company provides robust secure channels between its distributed repositories. The security on the update is based on TLS/SSL, which

was shown to be unreliable for community repositories with multiple developer teams in [10], [11]. Strictly speaking, currently, there is no security guarantee on the contents of update in the railway infrastructure. An adversary with knowledge of processes and protocols can easily execute a damaging attack i.e. install malicius software or provide outdated or incorrectly formulated update package. This can put the railway at the risk of critical incidents. The secure update framework should be adapted to the specific security requirements of each level. In this paper, we focus on the security of update delivery at the installation level. Currently, the thousands of devices are updated manually which requires a lot of time, effort, money and does not provide enough security guarantees, being dependent on the human factor.

3 TUF and Uptane background

The Update Framework (TUF) [7] is a security framework designed to protect the software repositories, such as Microsoft Windows Update, Ubuntu, the Python Package Index (PyPI), RubyGems, or Docker Hub, from attacks on update functions. The core of TUF security is based on signed metadata related to files or images kept on the repository. Attacks can be detected and prevented when the metadata is verified before the software installation. TUF does not prevent a compromise but limits its impact when it happens. TUF is designed around 4 key principles:

1. Separation of duties, i.e. different metadata files are signed by the repository administrators using 4 basic roles: the root, timestamp, snapshot, and targets.
2. Threshold number of signatures. The metadata file must be signed using a minimum threshold t out of n keys.
3. Key revocation by signing new metadata or by setting up an expiration date.
4. Using offline keys physically disconnected from the Internet.

Uptane is the first software update framework for automobiles that addresses a comprehensive threat model [5]. It enhanced the security principles of TUF [7] by adding several principles in order to increase the resilience to automobile attacks:

1. using additional storage to recover from attacks;
2. broadcasting metadata to to prevent incorrect version distribution;
3. using a vehicle version manifest, or data file signed by every ECU about what it has installed, to detect compatibility attacks;
4. using a time server to limit the denial of the latest updates.

4 Attacks on a railway update system

We consider the intruder or insider with capabilities to intercept/modify network messages, compromise device or update server components and pretend to be a device and ask for update. On the high-level the attacker in the railway update context has same generic goals as described in [5].

1. Read update: steal the contents of an update;
2. Deny update: Prevent the updates to reach the trackside equipment;
3. Misuse update: Make a device install an outdated or badly packaged update;
4. Malicious update: Modify the update in a malicious way so that the valid device functionality is affected causing it to fail or behave incorrectly;

4.1 Generic security attacks

The railway update is vulnerable to almost all the generic update attacks. We reproduce the list of generic attacks mentioned in [5]:

1. **Read updates:** the man-in-the middle attack permits the adversary to get access to update contents, even if it is protected with a signature.
2. **Deny updates:** adversary can block the update in one of the following scenarios
   (a) **Drop-request attack:** the network traffic can be blocked for some trackside devices so that they cannot receive the update. The human factor excludes the drop-request attack on the generic and project design level i.e. this attack is possible between installation team and device.
   (b) **Slow-retrieval attack:** causes the update to be received so slowly that a vulnerability can be exploited meantime. This attack can happen on all levels.
   (c) **Freeze attack:** does not permit to receive actual update. This attack is possible between the on-site installation team server and device.
   (d) **Partial bundle installation attack**: the latest update is not installed by all device components.
3. **Misuse updates:** an adversary can make an update harmful for a device in one of the following scenarios
   (a) **Rollback attack:** Device installs outdated update. The known vulnerabilities are still present on the device.
   (b) **Endless data attack:** Sending an indefinite amount of data to a device.
   (c) **Mixed-bundles attack:** Causes device to install incompatible component versions.
   (d) **Mix-and-match attack**: Attacker is able to release a random mix of software versions by compromising repository keys.
4. **Malicious updates**: adversary substitutes the contents of update forcing the device install a malware.

Apart from these we discovered specific attacks on trackside equipment that do not exist in the automobile update.

4.2 Specific attacks

We derive the classes of specific attacks analyzing the implementation of the safety requirements [4] to the railway update process. We didn't apply formal vulnerability analysis which can be the subject of another paper. We suppose that the attacker can target any of 3 specific railway procedures:

1. **Distribution of update to multiple devices.** The update is not distributed for each single device independently but in a synchronized manner for the entire network segment (or one trackside line of the project) which may contain hundreds or thousands of control devices to be upgraded. All of them have to receive the validated update within a short time slot. In other words, an attacker can perform a malicious action to a single device in the update network which should be prevented with minimal consequences.
2. **Testing phase.** The new version undergoes a series of tests before being approved for a commercial use, so there is a time delay between the update installation and actual switch to the new version.
3. **CB approval confirmation.** If the installation team was compromised and has got access to a non-approved version from the project repository, it can make devices install the non-legitimate version bypassing the installation team manager.

We may formulate the specific railway attacks:

1. **Device mismatch**: The update is correct, but is distributed to a wrong device in the network. This can lead to an installation of the incompatible software on a device, incorrect functioning of the signal device and errors in train positioning and interlocking. This attack is not covered by Uptane functionality, as Uptane is focused on the mismatch attacks on the device components or on the version mismatch of the update for the whole system or network. This attack is easier to execute if the metadata file is not composed specifically for each device i.e. the components to be installed on a trackside device are selected from the complete metadata file received by the device for the whole network in the project. In order to protect from a mismatch attack, the verification procedure on a device has to correspond to the specifics of this metadata format.
2. **Test phase attacks**: the tests start before all devices confirm the update installation or the new version is compromised during the testing phase. If the tests start before the devices confirm the installation of the fresh version, this could lead to incompatibility errors and failure of the testing phase which would make the whole update inefficient i.e. the attacker is able to deny the update. If the version is compromised during the testing phase it could lead to the successful end of the testing phase and installation of erroneous version. Uptane does not consider this phase at all as it does not exist in the automobile context.
3. **CB approval absence**: The device switches to the updated version before the CB approval due to a compromised installation team server. If the version is not approved by CB it can't be used by the devices as in this case the major responsibility on the failure is on the manufacturer of the update. This phase does not exist for automobiles, so Uptane framework does not protect against it.

To conclude, we have identified that all major attacks on the automobile update are also possible in the trackside update process. However, there are some

specific aspects of train updates that cannot be protected without adaptation of the generic update TUF/Uptane framework. The first security requirement is a strong compromise-resilience i.e. it would be infeasible for an attacker to achieve the attack goal even for a single device as the failure of a single device can be critical for the train interlocking resolution. The second requirement is secure recovery: the system should to be able to recover from attacks in the most secure way possible. Physical attacks, remote exploits and random failures are out of the scope of the system.

5 Uptane application for train updates

The update framework can be applied on all 3 levels of the railway update process: generic, project design and installation. In this paper, we focus on the application of TUF/Uptane at the on-site installation level, including metadata verification and specifics. It is considered, that the update package has already arrived at the installation server verified and signed by the project director. This section is our contribution to the railway update process. We propose the procedures and metadata verification based on the attack and context analysis. Other Uptane adaptations can be proposed later.

5.1 Customizing Uptane design at the installation level

In this section, we present a straightforward application of Uptane to the installation process. We propose the scenario, when the devices pull an update from the installation or team manager (TM) repository which corresponds to the image repository in Uptane framework, where the team manager signs the canonical image metadata [5]. Team manager on his/her behalf should verify and sign the binaries from the project repository, get CB approval and upload the images and freshly generated metadata (with his/her signature as a root) to the repository. Sometimes, in order to simplify the task, TM could directly sign the root metadata for the approved update or just add his/her signature on top of *root.json* file from project repository. Specifying the secure link between the project repository and TM repository is a subject of another paper. Ultimately, the devices trust the signature of the TM root, not the project director's one as CB is the final decision instance represented by TM root of trust. It is important that there is no link between the project repository of the design phase and installation phase other than CB-approved repository. This process is not specified in the Uptane framework so additional security procedures are necessary to protect it.

After that, installation team (IT) servers pull the specific approved images from this secure repository, sign and encrypt them and forward to the trackside. The installation team coordinates and orchestrates the updates of all devices on a specific line. The devices pull the update as directed by the IT server. The IT repository corresponds to the director repository in the Uptane framework, which is responsible for selecting and encrypting the customized set of images for

the specific device. Obviously, the IT repository should maintain the inventory database in order to make a decision on the version and baseline to be installed and a set of device keys in its network zone. The IT repository might contain encrypted images of an update if additional security is required. In the railway framework, the multiple independent director (IT) repositories correspond to a single image (TM) repository.

When a device would contact the IT repository, it would identify itself with the ID, serial number and signature. Given the device ID the IT director should consult the inventory database, perform dependency resolution for the required images and sign the unique metadata or instructions on the names, sizes and hashes of the images to be downloaded and installed by the device. This guarantees immunity to man-in-the-middle device mismatch attacks.

In the full verification scenario of Uptane [5], each device is required to download one set of metadata from the IT server and another, generic one, from the TM repository. The first set should be specifically customized for this device and the latter is required in order to verify the specific set against the metadata approved by CB and signed by the team manager. This Uptane procedure requires customization as TM metadata contains the signed instructions for all images to be installed in the network. Each device should verify only specific information on hashes, names and sizes for several files from this generic metadata. In the Uptane framework such verification is optional, but we propose to make it obligatory for the railway update in order to achieve greater compromise resilience and avoid non-approved version installation. The following diagram 3 illustrates the proposed application of the Uptane process in the railway context. All components and procedures of Uptane are applicable in this scenario. Additional safety requirements to SIL4 boards on the trackside will require additional safety checks to be performed after Uptane metadata verification. The update

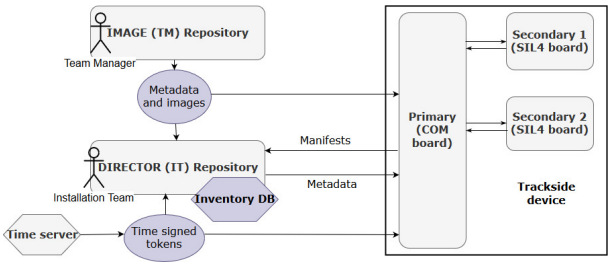

Fig. 1. UPDATE HIGH-LEVEL DIAGRAM.

of a single device in such scenario will not be an independent procedure but a part of network update. In order to control the network update, it is suggested to maintain an update progress database as a temporary file or a part of the inventory database. This database would contain the confirmation of successfully validated and installed updates for each trackside device. Once all devices will confirm the successful installation of a fresh version the testing phase may start. The procedure is not specified in Uptane, so this customization is obligatory in

order to comply with railway security requirements. The Uptane repositories and roles in the railway framework can function as proposed on the following diagram 2. We may conclude that the principal adaptations of the Uptane framework at

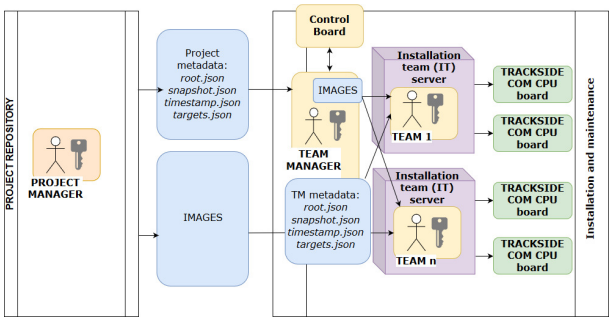

Fig. 2. ON-SITE REPOSITORIES AND ROLES.

this level are:

1. Separate *target.json* files for each trackside device in IT (director) repository in order to protect from the device mismatch and device freeze attacks.
2. Target metadata in the TM and IT repository are not identical and its verification includes only comparison of device component file names, sizes and hashes. This allows to manage multiple director repositories corresponding to common shared image repository (TM).
3. In order to proceed to testing phase,the IT repository needs to keep track of successful updates of all devices. The update progress database allows to control the start time of the testing phase and manage the update failures.
4. Full verification on each device is required as this it assures the identity of the installed version and CB approved one. This allows to support the strong compromise-resilience and minimize the risk to install a non-approved update version.

5.2 Roles and keys

The proposed IT repository roles and keys are presented in the figure 3. Each IT

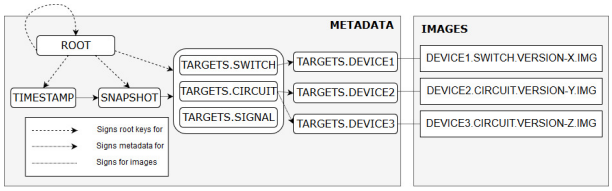

Fig. 3. INSTALLATION TEAM REPOSITORY ROLES.

server would be an Uptane director repository with 4 basic TUF roles so that each single device can verify the authenticity and timeliness of update:

- *root*: recommended minimum 8 keys with 5 keys valid signature threshold [8].
- *timestamp* and *snapshot*: one single key per role.
- *targets*: if possible, set up a separate target key for each type of device to reduce the possibility of device mismatch attack.

When the TM receives project metadata signed by the project director, he/she should verify it and download the binary images. After that, he/she must call the CB and ask to approve the version. Once the approval is received he/she will put a signature on top of the signed root metadata file or formulate new root metadata, add download links to target metadata and re-sign it, and generate fresh timestamp and snapshot metadata. He/she will forward it to each IT repository. The roles and keys on the TM repository are identical to those on Uptane image repository [8], [5].

6 Security considerations

Essentially, the TM and IT repositories function similarly to Uptane image/director repositories. Similarly, the informal security considerations [5] hold for the railway update as the same distribution of roles and responsibilities and key validity threshold support compromise resilience. The security analysis of the railway framework in generic part is identical to the Uptane framework security analysis [5].

The proposed adaptations allow to address specific trackside attacks (see 4.2), as TUF and Uptane do not have tools for such scenarios.

Table 1. SPECIFIC ATTACKS AND MEASURES.

Attack type	Measures and adaptations
Device mismatch:	uniquely tailored targets metadata temporary inventory database targets key corresponding to the specific device type
CB approval absense:	full verification for each device image repository metadata signed by TM direct upload URL in the targets metadata
Test phase attack:	temporary inventory database TM as a link between OT and CB Final version verification after the testing phase

It can be concluded, that the proposed enhancements: testing phase protection, full verification of CB-approved version, device-specific metadata and update progress control database provide the stronger compromise-resilience and a solution to balancing between safety and security requirements for the trackside update as none of these specific attacks is considered in TUF and Uptane.

Regarding the formal approach evaluation, to our knowledge, the protocol security proof, similar to [13] does not exist for this framework, though it is a state-of-the-art standard in the automobile industry and community repository

updates. The intuitive security analysis [7], [5], [11] has clear foundation on compromise-resilience, based on the number of keys and targets a well defined class of attacks. In future work, it should be feasible to design a formal model of secure update similar to [13] and prove its consistency for a railway framework.

7 Conclusion

The paper proposes an adaptation of TUF and Uptane for securing railway updates. We studied the generic update process in the railway environment in collaboration with an industry partner and identified generic and specific attacks on the trackside infrastructure in order to address the security and safety requirements. We studied the Uptane/TUF secure update system as the industrially validated solution in order to design the customization.

We propose the design of repositories, roles and metadata management adapted to the railway update process. As the railway update is more demanding compared to that of automobiles the additional security measures are taken into consideration such as testing phase, full verification and update progress control. Uptane is currently being validated in the testbed of the Digitrans demonstrator. Uptane update framework is expected to be implemented on a simulated section of trackside. Some devices will be virtual, and some will be implemented on Raspberry PI computers. The interface of Uptane should be adapted for the needs of signaling devices according to the recommendations given above. Each signaling object should be represented by COM board and one or two SIL4 boards.

Currently, TUF and Uptane are state-of-art mechanisms used by multiple automobile companies and the corresponding standard is in the approval process by IEEE to be adopted as secure update for ground vehicles [8]. It is used by IoT update frameworks as well, such as ASSURED [12]. Though it introduces new risks such as the key theft or compromise, the framework security guarantees allow to recover with minimal damage after such incidents. In our future work, we plan to adapt it to all levels of the railway update process i.e. for securing generic and project repositories as well as developer groups contributing binaries to the trackside components.

8 Acknowledgements

I would like to thank Dr. Florentin Rochet and Franois Koeune (UCL, Crypto group) for the valuable comments on the paper and Eric Denayer and Michel Rousseau (Alstom research) for the valuable discussions on the railway update process. This work has been funded in part by the Walloon Region (competitiveness pole Logistics in Wallonia) through the project Digitrans (convention number 7618).

References

1. Standard: CENELEC - EN 50159. Railway applications safety-related communication in transmission systems (2010).
2. Standard: ISA/IEC 62443: Industrial Network and System Security.
3. Standard: EN 50128 : railway applications software for railway control and protection.
4. Standard: EN 50129 ; railway applications safety related electronic systems for signalling.
5. Uptane: securing software updates for automobiles, 2019: `https://uptane.github.io/`. Last accessed 5 Apr 2019.
6. Airbiquity: OTAmatic update and management, 2019: `https://www.airbiquity.com/product-offerings/software-and-data-management`. Last accessed 5 Apr 2019.
7. TUF: The Update Framework, 2019: `https://theupdateframework.github.io/`. Last accessed 25 Apr 2019.
8. Uptane IEEE-ISTO Standard for Design and Implementation, 2019: `https://uptane.github.io/uptane-standard/uptane-standard.html`. Last accessed 5 Apr 2019.
9. Bellissimo, A., Burgess, J. and Fu, K.: Secure software updates: Disappointments and new challenges. In: Proceedings of the 10th conference on USENIX Security Symposium-Volume 10, p.22. USENIX Association (2001).
10. Cappos, J., Samuel, J., Baker, S., and Hartman, J.: A look in the mirror: Attacks on package managers. In: Proc. 15th ACM Conferenceon Computer and Communication Security, pp. 565–574. ACM Press, New York (2008).
11. Samuel, J., Mathewson, N., Cappos, J., and Dingledine, R.: Survivable key compromise in software update systems. In: Proceedings of the 17th ACM Conference on Computer and Communications security, pp. 61–72. ACM Press, New York (2010).
12. Asokan, N., Nyman, T., Rattanavipanon, N., Sagedhi, A.-R., and Tsudik, G.: ASSURED: Architecture for secure software update of realistic embedded devices. In: Proceedings of EMSOFT'18, Turin, Italy, article No. 16. IEEE Press Piscataway, NJ (2018).
13. Lewi, K., Kim, W., Maykov, I., Weis, S., and Facebook: Securing Update Propagation with Homomorphic Hashing, 2019: `https://eprint.iacr.org/2019/227.pdf`. Last accessed 24 Jun 2019.