MENDER

A MENDER WHITE PAPER

Security considerations for remote management of software in IoT devices

IoT security threats are different from traditional IT security environments such as servers, laptops, desktop computers and mobile devices. Particularly in server IT, security is mainly focused on protecting data stored in large server infrastructures that are in close proximity and can be reachable at relatively low cost. However, in IoT while data protection concerns still exist they mainly extend further into the physical world and most often at large scale with heavy reliance on battery and wireless connectivity where interruption in operations can cause millions of dollars of damage within a short period of time. Therefore, it inherently becomes harder to secure IoT devices as compared to other environments.

> **IoT security threats are different from traditional IT security environments.**

The security challenge has drawn the attention of users, developers, device manufacturers and government agencies around the world. The hacking of a wide spectrum of smart devices such as smart fridges, and baby monitors to the infotainment system in your car are indicative of a security trauma being caused by the nature of these devices being online and vulnerable to attacks.

For the most part, the lack of security concerns in IoT has to do with the industry being in its 'gold rush' phase where in every marketplace vendors are rapidly pursuing to develop and distribute the next 'big thing' before competitors do, and customers not caring or understanding the security aspect. Under such a competitive business environment, functionality becomes the main focus and security takes a back seat.

This paper is written for technical leaders addressing security threats when designing connected products and how software updated remotely if implemented poorly can provide additional attack vectors to already existing device vulnerabilities. The paper discusses the need for a secure software update process and dive into specific use cases and how Mender over-the-air (OTA) software updates can securely manage devices remotely.

A secure software update process must be at the core of product design

Prevent and contain security threats

A device security breach incident can interrupt operations, damage systems and negatively impact both virtual and physical processes. This translates into unhappy customers and lost business. Attackers can steal data, compromise them and hold it ransom.

Consider the TRITON[1] attack case in the recent year, where a number of publicly identified malicious software families targeted industrial control systems. The malware contained executable code that caused total disruption of the operation. The malicious actor exploited a remote terminal access to an engineering workstation to deploy the TRITON framework to reprogram the system controllers. Though it is difficult to know exactly how they gained access and if the workstations had too many privileges in terms of accessibility, it is possible that Role Based Access Control would have helped to significantly reduce the risk of unauthorized software deployments. If the attack was that the engineering workstation had an open session to the over-the-air (OTA) server which it used to deploy malicious software in a group then RBAC could have limited the impact of the damage by only allowing the attacker to deploy to a small group of control systems, or potentially none at all (if the engineering workstation did not have deployment rights).

Additional layer for preventing security threats could be with audit logs, with which in the case of the TRITON attack if implemented one could see which user created a remote terminal session to which device and when. Audit logs can be an important basis for increasing security, policy compliance and accountability by logging events that can later be analyzed in case of operational incidents. Audit logs can be a great tool to do post mortem analysis in order to prevent similar malware attacks in the future. In the TRITON case where the engineering workstations were dismantled, the audit log can be used to check the software deployments that were made against them and who created the deployment and when.

In the above case, security breaches could have been prevented entirely or at least significantly contained if security measures were installed in a timely manner using a secure, efficient and risk tolerant software deployment process.

If priority is given on securing software that keeps the OS, system and applications working on connected devices, then vendors must have a security-by-design philosophy at the center of the software updating solution to prevent malicious attackers infiltrate devices and maintain the health of their devices in the field.

1 Reference:
https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html

> **A device security breach incident can interrupt operations, damage systems and negatively impact both virtual and physical processes.**

Comply with government regulatories

Government bodies around the world have started to mandate the implementation of security protection measures in all internet connected devices. In June of 2020 the European Union[2] (EU) introduced a new cybersecurity standard for all consumer IoT products. It hopes the new standard will lead to better security practices and more vendors adopting a security-by-design principle when developing new consumer IoT products. Similarly in the U.S., a cybersecurity legislation[3] was passed through US Congress which facilitates the tightening of controls around IoT products the Federal Government can procure. The legislation is based on adherence to a set of security standards which guard against the threats of software vulnerabilities, credentials snooping and malicious takeovers. This legislation will now be taken by standards body NIST[4] and federal procurers and put into practice. This landmark legislation will drive IoT hardware and software vendors to raise the security standards in their offerings to align with the requirements of the Federal Government.

Over-the-air (OTA) software update capability in IoT devices will be a regulatory requirement in all the industries in the coming years which could potentially mean fees and serious business consequences if not adhered to, therefore devices built today need to be mindful of this.

[2] Reference: *European Standard, Cyber Security for Consumer Internet of Things, Baseline Requirements (2020)*. Retrieved from: www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf

[3] Reference: https://www.ssa.gov/legislation/legis_bulletin_092220.html

[4] Reference: *NIST, Security and Privacy Controls for Federal Information Systems and Organizations, SP 800-53 (April 2013)*

Attack vectors in the software update process

A secure software update process can be defined as: a) the authorized device has the latest available authorized software, b) it has the correct software files downloaded and installed, and c) no adverse effect on expected device operation and functionality results from checking, downloading and installing software files. Implementing a secure software update process requires preventive strategies against a number of potential attack vectors.

Table 1 provides a list of known potential security threats related to software update systems along with the weaknesses that make them possible.

Attack vector	Role in an attack
Installation of malicious software	Malicious actors can infiltrate and provide arbitrary software files in response to requests and install anything they want on the client (device) with no one detecting the illicit use.
Unverified software deployment	An obsolete or altered version of a software is knowingly installed on a device that may contain vulnerabilities which is exploited by attackers.
Compromised build systems	System firmware such as the bootloader is the first code to run on a device on startup and can undermine the operating system by changing bootcode, and patching the OS kernel. This can also be used to deliver malicious firmware to other components such as the root file system and application-level code.
Physical attacks	File downloaded with an endless stream of data causing harm to devices such as memory exhaustion interrupting device operations.
Device-server communication	Attacker updates software in transit; A device will send its backup out to the cloud and will suffer a short downtime during reboot. If the connection is unencrypted and the update files are unprotected, a hacker could steal or alter sensitive information.
Convoluted updates	Attackers send clients with files that did not initially exist and this can result in outdated versions of dependencies being installed, and complications in device operations may occur.
Unnecessary software dependencies	Malicious actor forces the client (device) to install the desired software by first allowing it to install an unrelated code. This code may have known vulnerabilities that will exploit the device.
Indefinite seize	Attacker continues to present files to a software update manager with files that the client has already installed. As a result, the client is kept unaware of new files.

Table 1: Potential security threats posed by insecure software update process.

Implementing a secure software update process requires preventive strategies against a number of potential attack vectors.

Triangle of Trust™: Mender security-by-design principle

To ensure systems are secure against attack vectors the remote management of software should adhere to three pillars of trust that look to protect people, devices and software (figure 1). This security-by-design principle focuses on delivering updates securely, maintaining the identity of the client-server communication endpoints, ensuring device authentication, user authorization together with hardware security integration to defend against any potential attacks during the update process. The goal for Mender is to provide a robust, secure and efficient update process. An important part of this is to give the Mender client running on the device the ability to verify that updates come from a trusted source. More on this in the sections that follow.

> To ensure systems are secure against attack vectors, the remote management of software should adhere to three pillars of trust: People, device and software.

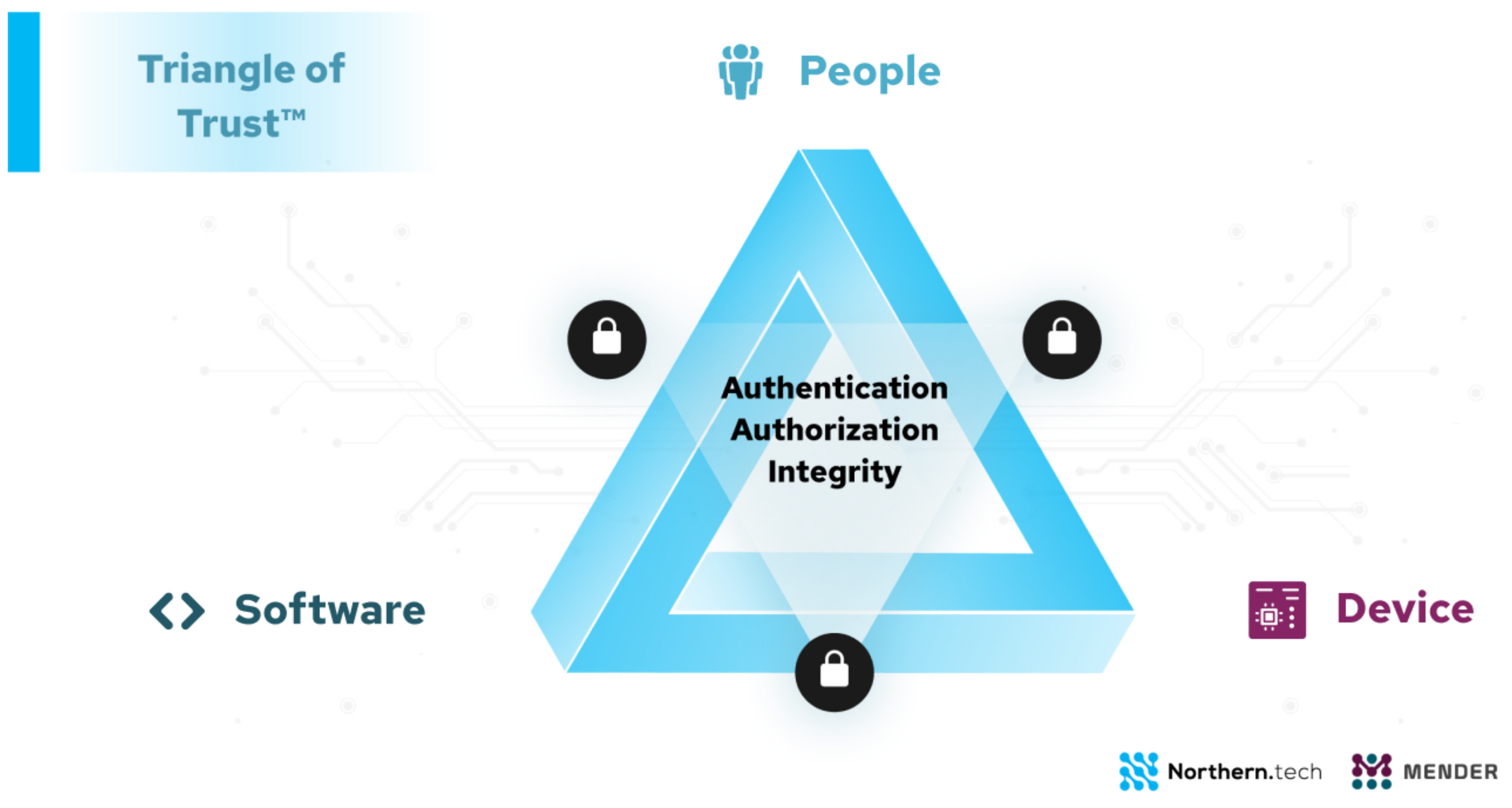


Figure 1: Mender's three pillars of trust in managing connected devices.

Mender's defensive strategies

Man in the middle

The Mender client runs on devices, meaning it is integrated as part of the operating system image at build time, and communicates with the Mender server in order to authorize and get updates. Communication between the client and server happens via a REST API over a TLS-encrypted channel. The Mender client relies on the operating system's root Certificate Authorities (CAs) to verify the server's identity by default. You can also configure the client to use a specific certificate for chain validation, for example when using self-signed certificates. This ensures that the Mender client will only connect to a verified server, and no man-in-the-middle attack is possible.

> **The goal for Mender is to provide a robust, secure and efficient update process. An important part of this is to give the Mender client the ability to verify that updates come from a trusted source.**

Unauthorized device

Mender leverages public key cryptography for protecting updates to devices from unauthorized use. Each device in a Mender managed fleet has a unique public key and private RSA key with a default length of 3072 bits. You can generate this offline and provision it with the device storage, otherwise the Mender client will automatically generate a key pair when it launches for the first time. Once generated, private keys cannot be changed or retrieved by means of API calls. If you decide to re- generate the keys on the device, it will require going through the authorization process again.

This process can enable customers to integrate with their existing Public Key Infrastructure (PKI) for simplicity, pre-authorize to automatically approve devices which can help in a factory level setting, and avoid manual acceptance of devices which could prevent scalability.

Mender further provides additional hardware security later for authentication. Hardware Security Modules (TSMs) and Trusted Platform Modules (TPMs) securely store keys inside hardware, making them tamper proof and harder to steal. Device applications like Mender only operate on these keys, such as requesting signing and decryption, rather than reading the keys and operating with them directly. This ensures that the OTA update process with Mender is secured in the same way as other applications leveraging cryptography operations on the device.

Device network intrusion

Major internet-wide IoT botnets like Mirai, Hajime and BrickerBot all relied on reconnaissance for open ports as a starting point to attack devices. The Mender client initiates all communication by connecting to the server, so no open ports on each device are required on the device in order to use Mender. This helps protect against vulnerabilities caused by some devices in the update process where manufacturers do not require authentication to connect to a router using protocols such as TR-069 or TR-064. When there are no open ports, there is no way from the outside world to get inside the device without intercepting established connections. As long as the Mender client can connect to the server over HTTPS, updates can be scheduled. An example of this could be seen in the Mirai botnet[5] case where attackers accessed DSL telecom routers with open ports and launched attacks against some 900,000 Deutsche telecom customers. The port was used by network operators for remote management of routers and setup boxes which also lacked client-server authentication.

[5] Reference:
https://www.bankinfosecurity.com/mirai-botnet-knocks-out-deutsche-telekom-routers-a-9565

Unauthorized software

Mender supports a digitally signed OTA software update process. A Mender artifact signature verification allows the client to have access to the public part of the keypair used for signing the Artifacts. The Mender client can validate the artifact payload (i.e. code to update) independent of the communication channel verification. This provides an additional layer of protection. If the signature verification check passes, the client considers the update to come from a trusted source and continues. Otherwise, the client refuses to proceed with the update and raises an error message. A management server being compromised, might allow malicious software to be downloaded, but it will not be installed as the signature verification will fail. An example of this was seen in the Tesla S[6] attack case where the hackers used vulnerability in Tesla's Linux operating system to gain full privilege on the car's infotainment system. They simply overwrote the gateway's firmware with their own without code signing. "Cryptographic validation of firmware updates is something we've wanted to do for a while to make things even more robust", said Tesla's CTO JB Straubel.

[6] Reference:
https://www.wired.com/2016/09/tesla-responds-chinese-hack-major-security-upgrade/

Being able to provide signed images greatly enhances the security of the system. Maintaining the private keys separately from the code base and server infrastructure allows decoupling of these components. This means that even if the server is breached, for example, then attackers will not be able to install new images onto client devices as they will not have the appropriate signing keys. It is crucial that the application of the signatures be a separate step from the rest of the build and deploy mechanism to best protect this resource. Figure 2, illustrates the high level flow of this process and the components shown are essential for the Artifact signing and verification process.

"Cryptographic validation of firmware updates is something we've wanted to do for a while to make things even more robust", said Tesla's CTO JB Straubel.

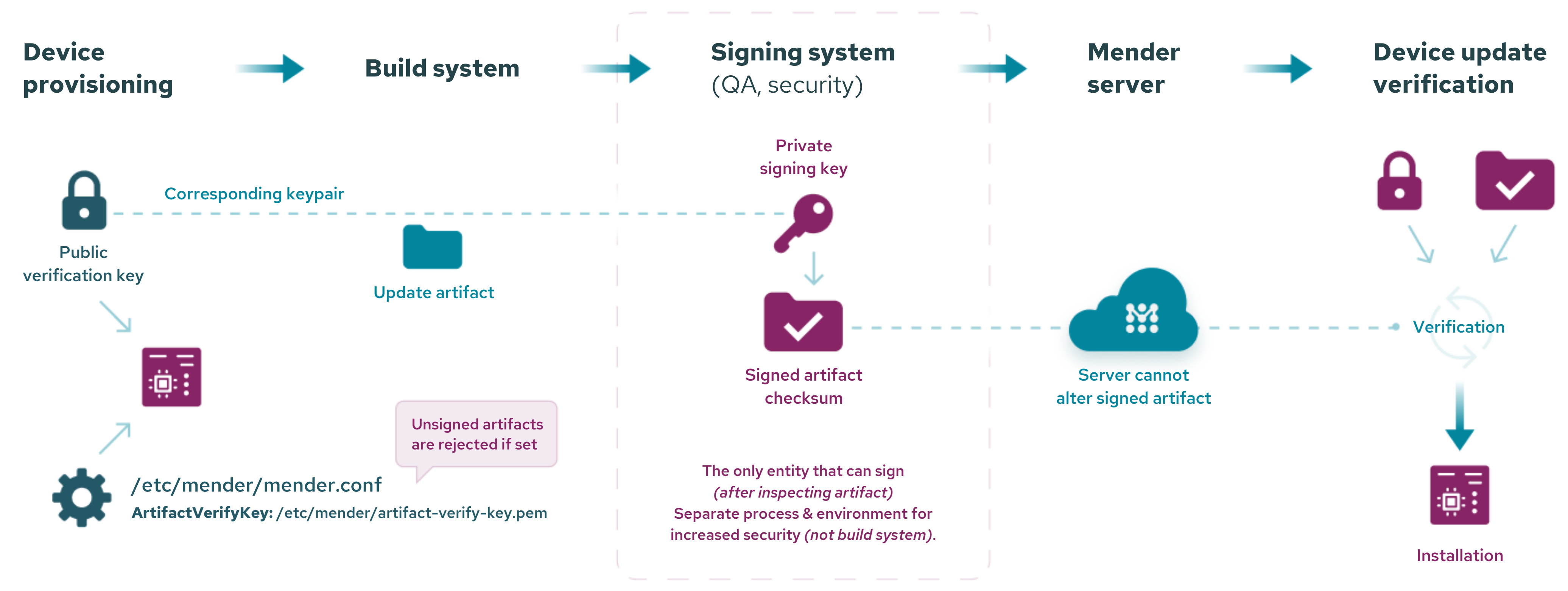

Device provisioning
Build system
Signing system (QA, security)
Mender server
Device update verification
Corresponding keypair
Public verification key
Update artifact
Private signing key
Signed artifact checksum
Server cannot alter signed artifact
Verification
Unsigned artifacts are rejected if set
/etc/mender/mender.conf
ArtifactVerifyKey: /etc/mender/artifact-verify-key.pem
The only entity that can sign (after inspecting artifact) Separate process & environment for increased security (not build system).
Installation

Figure 2: Signature management flow with Mender.

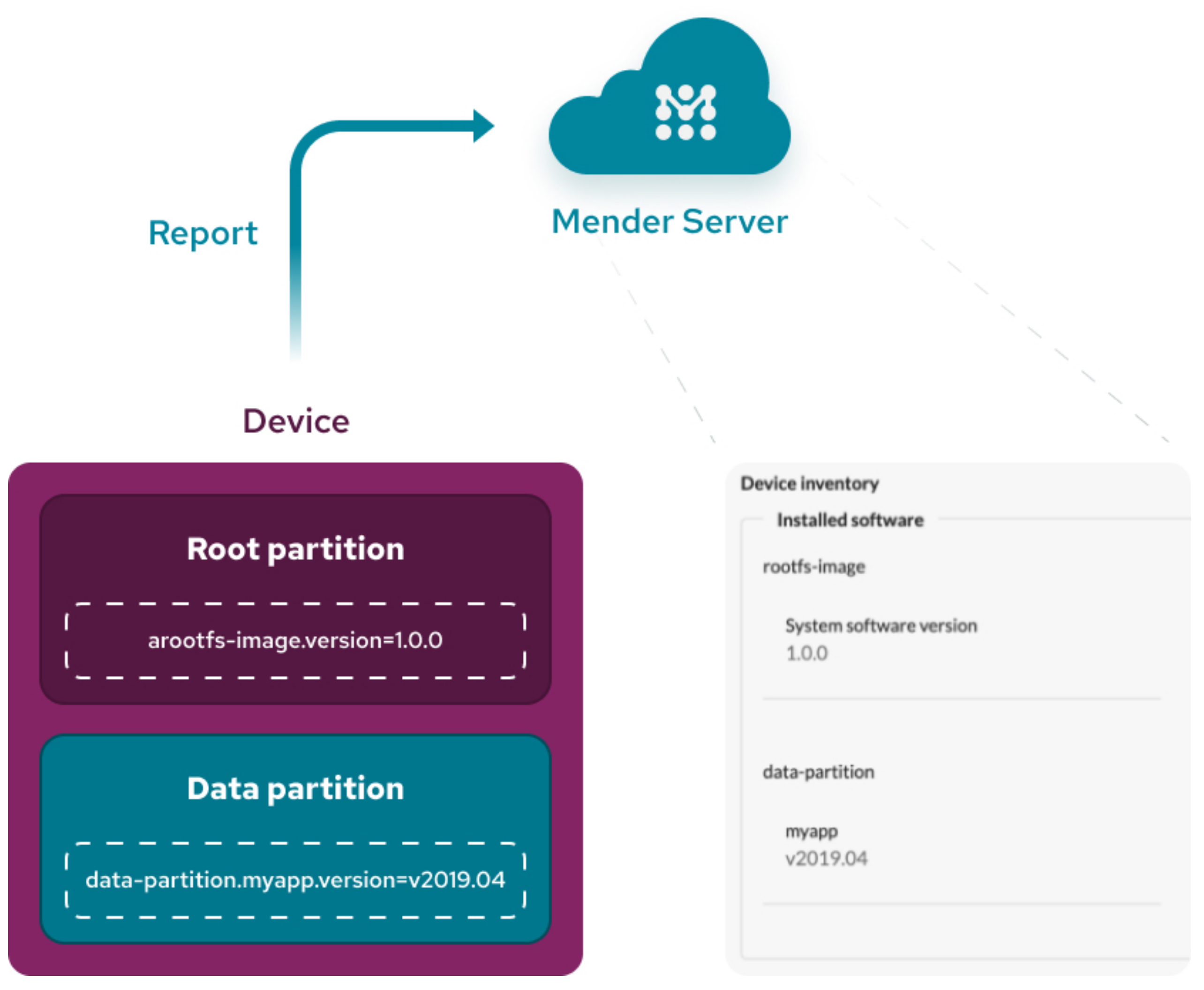
Unauthorized user

Mender offers strong measures to avoid unauthorized access. Users interact with devices either via API calls relayed by the Mender server, or via the web UI. In both cases, the connection is HTTPS encrypted with TLS. Mender offers two Factor authentication. Further, as an additional feature layer of security, Mender comes with Role Based Access Control which access privileges based on the responsibility of the user and required tasks (note the TRITON attach case above).

Software versioning and device compatibility

Mender ensures that the intended software is downloaded and installed on the intended device. In order to ensure this, Mender has additional metadata alongside the raw bits of the updated Artifact payload (i.e. software code to update). Depending on the version of this software used, the metadata might be different, but must contain the name of the build, version of the software, device type(s) the software is compatible with, and checksum of the software so that it is not corrupted during transit or storage. An example could be seen in the Airbnb 'smart locks'[7] case bricked by a bad software update where Lockstate, the manufacturer of the door lock, had pushed a firmware update to locks that should not have been updated with that software version. Though it is difficult to know exactly how the update process handled this, it could have been prevented by an update process that ensures the validity of the software to be downloaded and the compatibility of the device. Mender supports software versioning for both system and application updates with each having their own name and version tuples. Figure 3, shows version reporting by the Mender client as inventory data in the Mender web UI.

[7] Reference: https://www.extremetech.com/computing/254177-internet-things-smart-locks-bricked-bad-firmware-update


Report
Mender Server
Device
Root partition
arootfs-image.version=1.0.0
Data partition
data-partition.myapp.version=v2019.04

Device inventory
Installed software
rootfs-image
System software version
1.0.0
data-partition
myapp
v2019.04

Figure 3: Software version reporting.

Conclusion

Mender is a fast growing open source and commercial remote software manager. To design a secure software update process, a variety of attack vectors need to be taken into account. Mender adheres to a security-by-design principle governed by the Triangle of Trust™ to ensure secure end-to-end software updates.

A secure and robust software update process should be the foundation built into the product development strategy. The software update framework must be designed to minimize hackers' ability to breach the update process and harm devices by modifying and installing malicious software on them. To ensure this, the underlying criterion lies precisely in a framework that must consider the three pillars of the triangle of trust: people, device and software. This is critical for delivering the capabilities and functionalities needed by developers, device manufacturer's and end users. Only by planning ahead with the right mindset and design philosophy can you ensure a secure remote software update strategy.

> Mender adheres to a security-by-design principle governed by Triangle of Trust™ to ensure secure end-to-end software updates.

MENDER

About Mender.io

Mender.io is a leading provider of a secure and robust end-to-end over-the-air (OTA) software update manager for IoT devices. Mender makes it easy to deploy updates to a large number of devices by providing efficient and risk tolerant OTA deployments. Mender enables its customers to stay competitive in a fast-moving market by helping them deliver high-value services on an increasing number of connected devices with growing software complexity. With an active open source community supporting a large number of different hardware and operating systems and growing every day, Mender has quickly become the trusted choice by some of the world's most respected brands.

Mender documentation

https://docs.mender.io/getting-started

Mender Hub community:

https://hub.mender.io

Mender on Github:

https://github.com/mendersoftware/

MENDER

CONTACT

+1 650 670-8600

contact@mender.io

mender.io