Emdec, 2024-04-16

Security Concept Trainboard

Purpose

The document describes the security concept for the Trainboard (see trainboard.ch) from an IoT perspective.

Structure of this Document

First the context and the scope of the security concept are presented. Then, the assets to be protected and the threats to these assets are detailed and analysed. Some requirements are derived from this analysis. Finally, an implementation concept is presented.

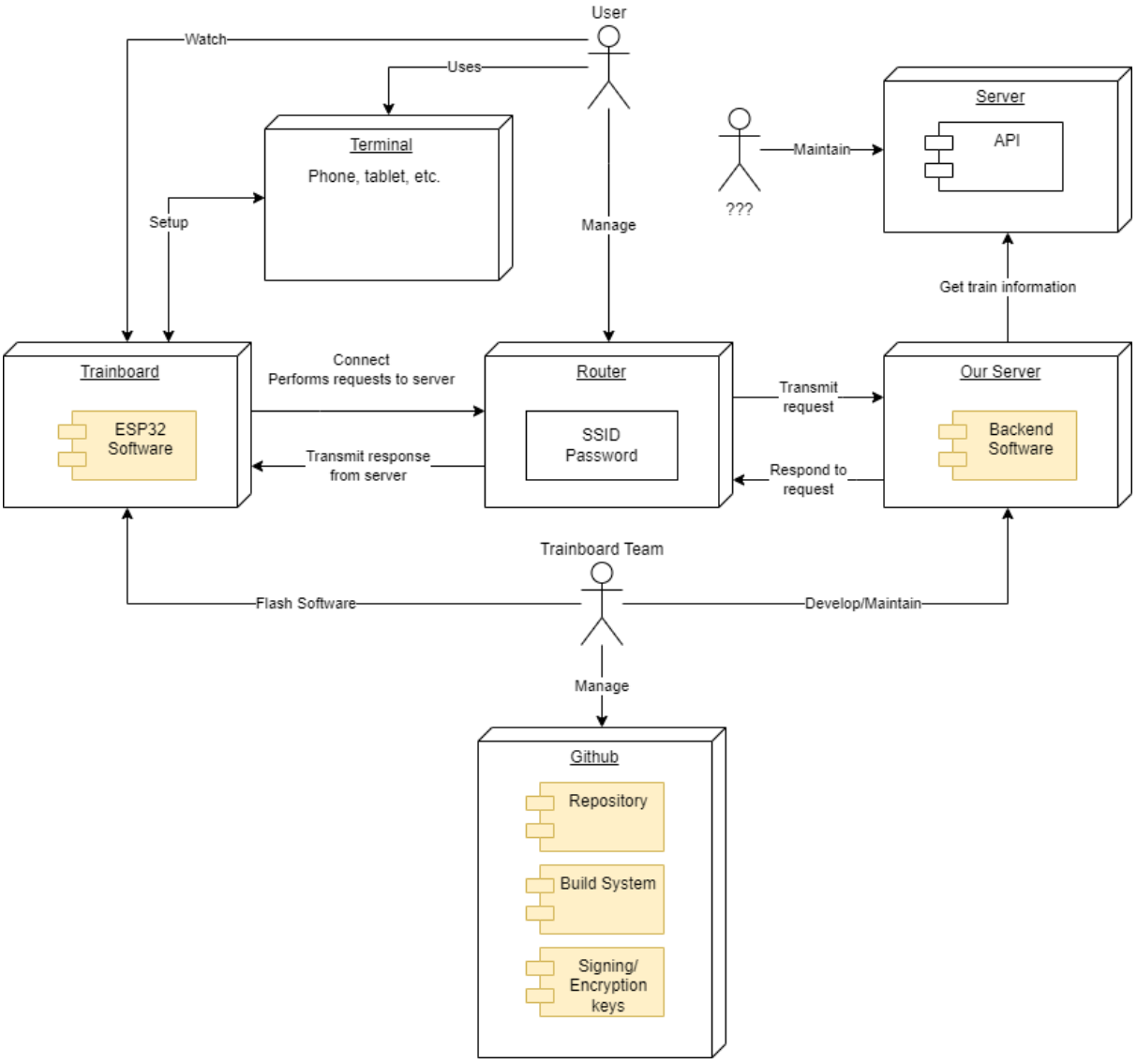
Reference Documents

The following documents are used as reference.

#	Name	Link
D1	Introduction to security for STM32 MCUs	https://www.st.com/resource/en/application_note/an5156-introduction-to-security-for-stm32-mcus-stmicroelectronics.pdf

Context and Scope

The following figure shows the context of the Trainboard. The scope of the security concept described in this document is the ESP32 as well as the backend software.



Emdec, 2024-04-16

Threat Analysis

Assets

The following table details what must be protected and the associated risk.

#	Asset	Risk
A1	Light Sensor Data	Privacy: presence detection in customer's home
A2	User access data for private network	Usurpation of customer identity
A3	Cryptographic keys	Deployment of malware to the devices
A4.1	Deployment infrastructure (GitHub + Server)	Loss of code base
A4.2		Malicious software updates
A4.3		No software updates
A5.1	Backend Software	Denial of Service
A5.2		Functionality break: what is shown is not what was intended to be shown
A6.1	Device Software	Reverse-engineering / Counterfeit
A6.2		Intellectual Property
A6.3		Functionality break
A6.4		Overwrite with malicious software
A7.1	Device Hardware	Counterfeit
A7.2		Misuse hardware
A7.3		Functionality break

Threats

The following table shows the potential threats in relation to the assets previously defined. They are grouped into three levels: remote, where the attacker has no access to the physical device. Local, where an attacker has access to the physical device, and chip for the sake of completeness.

#	Level	Threat	Technique	Targeted Asset(s)
T1.1	**Remote**	Access to back-end server	Social engineering, hacking, exploit known weaknesses	*A1, A5.1, A5.2*
T1.2		Access to GitHub account	Social engineering, hacking, exploit known weaknesses	*A3, A4.1, A4.2, A4.3*
T1.3		Eavesdropping	Monitor unencrypted traffic, exploit security bugs	*A1, A2, A5.1, A6.1*
T1.4		Usurpation of server identity	Spoofing	*A5.2, A6.3, A6.4*
T1.5		Usurpation of board identity	Counterfeit board	*A5.1*
T2.1	**Local**	Read-out of flash	JTAG, SPI interface	*A2, A6.1, A6.2*
T2.2		Boot modification	Boot pins, JTAG	*A2, A6.1-4, A7.2*
T2.3		Monitor serial output	USB interface	*A1, A5.1, A6.1, A6.2*
T2.4		Power glitches	Advanced techniques	*A6.3, A6.4, A7.3*
T2.5		Fault injection	Advanced techniques	*A6.3, A7.3*
T2.6		Side-channel analysis	Advanced techniques	*A2, A3, A6.3*
T2.7		Reverse-engineer board	Read part numbers, analyse traces	*A7.1*
T3	**Chip**	Read-out of flash	Invasive techniques	*A2, A6.1, A6.2*

Assessment

The following table evaluates the risks associated with each threat and shows possible countermeasures.

Threat	**Risk**	**Countermeasure**	**Applicable (yes/no)**
Access to back-end server	Mid	Use strong passwords	Yes
		Do not share the log-in information with people other than the trainboard team	Yes
		Use latest frameworks or packages	Yes
		Apply security patches	Yes
		Do not send light sensor data to server	Yes
Access to GitHub account	Low	Use strong passwords	Yes
		Do not share the log-in information with people other than the trainboard team	Yes
		Keep secrets in an appropriate place (not in the repo!!)	Yes
Eavesdropping	High	Use encrypted communication (HTTPS)	Partially[1]
		Minimise frequency and duration of sending sensitive data (e.g. passwords for home network)	Yes
Usurpation of server identity	Mid	CA certificate	??
		Secure boot	Partially
Usurpation of board identity	High	Check MAC addresses of boards	Yes
		Limit # Requests per board	??
		Ban boards that do not comply	??
Read-out of flash	Mid	Disable JTAG interface	No
		Flash encryption	No
Boot modification	Mid	Disable alternate boot modes	No

[1] During provisioning of the board, HTTP is used. Otherwise, HTTPS.

		Disable bootloader modification	No
Monitor serial output	High	Disable serial console in release mode	No
Power glitches	Low	N/A	No
Fault injection	Low	N/A	No
Side-channel analysis	Low	N/A	No
Reverse-engineer board	High	Scratch away part numbers	No
		Coat board with opaque coating	No
		Contract with customer or reseller to prohibit derived work	??
Invasive Read-out of flash	Low	N/A	No

Requirements

Id	Description	Mitigated threat	Done
R1	The development team shall be aware of the security risks related to credentials (server or build infrastructure).	T1.1, T1.2	
R2	The password for the server and build infrastructure shall be strong.	T1.1, T1.2	
R3	The passwords shall only be shared amongst the minimum necessary number of people.	T1.1, T1.2	
R4	The libraries and packages used on the server shall be kept up-to-date, at most at the oldest Long-Term-Support release.	T1.1, T1.3	
R5	Cryptographic secrets shall be saved in a secure place on the build infrastructure (e.g. Github Secrets)	T1.4, T1.5	
R6	All communication between the server and the device shall be secure (e.g. HTTPS).	T1.3, T1.4, T1.5	yes
R7	The firmware image shall be signed.	T1.4	
R8	The bootloader shall start the firmware image only if it could successfully verify the firmware image’s signature.	T1.4	
R9	The board shall not send any data related to the ambient light sensor to the server.	T1.3	yes
R10	It shall be possible to update the software remotely (OTA update)	T1.3, T1.5	yes
R11	The firmware shall be updated as soon as there are security patches for the used libraries.	T1.3, T1.5	
R12	The customer shall be informed that the credentials are stored on the device.	T1.3	
R13	The customer shall be informed that the provisioning of the board is done in an un-secure manner.	T1.3	

Remaining Threats

Some threats are not addressed by the requirements. They are listed below, along with a rationale.

1. Chip-level threats: The goal of attacks at the chip-level are mostly intellectual property. As all our code will be open source and we are not responsible for the protection of the hardware IP inside the chip, no countermeasures are taken.
2. Local threats:
    a. The intended use environment of the device is in areas where there is restricted access to the board (e.g. people's homes). It is unlikely that attackers will get physical access to the chip to read out credentials for the home network. Moreover, the source code will be open source, so there is no threat of reverse-engineering code.
    b. In the spirit of open source, the customer should be able to flash another software to the board. Thus, no barriers like secure boot, JTAG disabling, or eFuses are implemented.
    c. This means that all techniques needing physical access to the device are not considered as applicable threats and are not addressed by any security measure.
3. Provisioning: The greatest weakness of the security concept is the provisioning of the device. To minimise usage of project resources, an existing library is used for the provisioning of the device. This library uses an insecure protocol, meaning credentials for the network are transmitted to the board in plain-text. However, this is only necessary during the setup of the board. The likelihood of an attack at that particular moment is not critical. To mitigate this risk, the customer will be informed of these risks.

Emdec, 2024-04-16

Implementation Concept

Id	Description	Responsible	Req.
I1	The Development Team reads and understands D1.	All	R1
I2	The Development Team reads and understands this document.	All	R1
I3	The settings for the access of the server infrastructure are chosen in a conservative manner (access denied by default, granted when necessary).	Ueli	R3
I4	The credentials for the server are stored in a secured password manager (e.g. Google Password Manager, 1Password, etc.)	Ueli	R2
I5	The cryptographics secrets for the signing of the application and the generation of the certificate are stored in a secure location (e.g. Password manager, or GitHub Secrets).	Emile	R5
I6	The packages and libraries used on the server are watched and security patches applied as soon as possible. (nginx, waitress, flask, and debian security updates)	Ueli	R4
I7	Implement an OTA update functionality where the firmware can be pushed to the server and the boards automatically perform the update. The check is performed every 15 minutes.	Ueli / Emile	R10
I8	The libraries used in the firmware shall be watched and updated to the newest version as soon as possible: - ESP IDF - Arduino - FastLED	Emile	R11
I9	Implement HTTPS communication between the board and the server.	Ueli / Emile	R6
I10	Sign images using esp-idf build scripts during the build process and deploy only signed images.	Emile	R7
I11	Modify the bootloader so it verifies the signature of the image.	Emile	R8
I12	Add information about the non-secure provisioning in the README.md of the project's repository.	Ueli / Emile / Lenny	R12, R13
I13	Add an FAQ entry on the website to inform the customer that communication is not encrypted during provisioning and that the credentials are stored unencrypted on the device.	Lenny	R12, R13
I14	Add an FAQ entry on the website to inform the user that once the board is set-up, the communication between the	Lenny	

	board and the server is secure.		