IOT SECURITY CHECKLIST

December 2022

v1.1.0

SINTEF

Changelog

v1.1.0

- Q10: Add key generation offline, in production, as a measure against low randomness.
- Deleted Q27, which was similar to Q23.
- Fixes typo.

Driven by trends such as Industry 4.0 and digital twins, new technology paradigms are appearing in most industry domains, such as decentralized architectures, interconnected wireless devices across layers, edge-to-cloud communication and general-purpose components from a variety of vendors. This shift of paradigm introduces new attack surfaces for attackers to explore and poses a serious risk, especially in the context of critical infrastructure.

Research has shown countless of times vulnerabilities found in IoT devices are often easily exploitable even by inexperienced attackers while sometimes having a huge impact. This document aims at helping manufacturers and integrators to tackle those low hanging fruits. It can also be used by companies to quickly evaluate a product's security features or to establish procurements.

This checklist is also part of the wider project "Security Checklists" which aims at providing checklists that help dealing with security in different domains. The objective is not to provide exhaustive checklists, but rather to highlight the most common issues in a particular domain. They can for instance be used as a ground for discussion about security in a project, and we thus advise on doing so from the beginning.

This document is a deliverable of the Ragnarok project at SINTEF which focuses on (I)IoT security.

Checklist

The IoT Security checklist is a questionnaire-like document to be used for a self or guided assessment of an IoT device. The objective is to raise awareness on specific weaknesses. It aims to be domain agnostic. The questions come from both our experience working with IoT devices and guidelines such as the "Baseline Security Recommendations for Internet of Things in the context of critical information infrastructures" from ENISA.

Following the example of the OWASP Application Security Verification Standard, three levels are defined:

🟥 **Level 1**: Level 1 is the bare minimum security IoT devices should strive for. Complying with this level should counter attackers who are using simple and low effort techniques to identify easy-to-find and easy-to-exploit vulnerabilities. In the case the IoT device is processing sensitive data or critical for operation, you probably don't want to stop at this level.

🟧 **Level 2:** Level 2 aims to defend against the most common risks associated with IoT devices today. It is appropriate for devices processing healthcare data or other sensitive assets. Threats to level 2 are typically skilled and motivated attackers focusing on specific tools and techniques that are effective to discover and exploit weaknesses within application. Aiming at this level should be enough for most devices.

🟩 **Level 3**: Level 3 is typically reserved for devices requiring a significant level of security verification, such as in the military, health and safety or critical infrastructure domains. If you think your device must comply with level 3, then this checklist won't be enough in itself (it can provide a good start though) and you probably want to also look at IoT certification schemes such as Common Criteria, FIPS-140 or PSA Certified.

Hardware

🟥 **Q1. Are the debug ports disabled?**

Debug ports often used in development such as JTAG or SWD can be used by an attacker with physical access to the device to acquire the firmware of the device (and thus potential cryptographic material) and perform reverse engineering of it to identify software vulnerabilities. The UART is commonly used as a debug interface too and can provide valuable information to an attacker on what the device is doing.

🟥 **Q2. Are interfaces allowing memory access disabled?**

Similar to Q1, interfaces allowing memory access such as DMA should be disabled if not required to prevent an attacker to access memory. DMA being a useful feature for many IoT applications, an alternative is to restrict the memory accessible to it, by configuring the Memory Protection Unit (MPU) for instance.

🟧 **Q3. Do you use tamper proofing mechanisms?**

Similarly to obfuscation (Q7), one should rely on tamper proofing for security as any device or system can be broken by a person with sufficient knowledge, time and resources. Having

tamper proofing mechanisms[1] can however discourage or slow down certain attackers. Tamper proofing should not rely on network connectivity.

Q4. Is the device designed to make it difficult to access pins and electrical traces?

An attacker can easily probe a microcontroller's pins if they are exposed (through TSOP chip package for instance). Prefer BGA package when possible and avoid exposing electrical traces on the PCB.

Q5. Are you using any specialized security chips or coprocessors?

Security chips / coprocessors integrate security at the chip level, providing trusted storage of device identity and authentication means. They also provide protection of device keys at rest and in use, and prevent unprivileged code from accessing security sensitive code. Without this, an attacker will be able to retrieve device keys, potentially allowing him/her to impersonate the device, steal data and inject fake data into the system.

Q6. Are you employing hardware-based immutable Root of Trust (RoT)?

Without a RoT, there is nothing to prevent an attacker having physical access to the device to take over it, introduce unauthorized code, steal data or even to form botnets.

Q7. Do you obfuscate the device's components?

While it is not recommended to use obfuscation as a sole security mechanism (and to rely on it), using obfuscation can make the life of an attacker more difficult and potentially discourage a certain class of attackers (such as "script kiddies"). Obfuscation can consist of erasing/masking IC names, not labelling the PCB, etc. Adding obfuscation has however a production cost and can also trigger the "challenge spirit" of some attackers (some attackers will then try to hack the device for the challenge of it, just to show they can do it). It is thus recommended to only use it for devices already presenting a high security level.

Software

Q8. Do you encrypt content stored on external memory?

An attacker with physical access to a device might read external memory on the board and access sensitive content such as personal data and/or cryptographic material. An attacker might also modify the content of the external memory to fake data for instance.

Q9. Is the cryptographic material unique for every device?

If the devices have cryptographic material used to authenticate the device and secure the data (for instance keys), is it important for this material to be unique per device (see Q28).

[1] See Practical Secure Hardware Design for Embedded Systems, by Joe Grand, for more information.

Q10. How is the cryptographic material generated?

It is important that the cryptographic material is generated in such a way it cannot be guessed by an attacker. In practice, it means using a TRNG or a PRNG[2] with proper seed. An example of what **not to do** would be for instance to generate the device's key simply by hashing the device's ID. In practice, to avoid not having enough randomness on the device, device-specific keys can be generated offline, during production.

Q11. Are you using a secure cipher suite (strong encryption algorithms and strong keys)?

An attacker could take advantage of a weak mode of operation to misuse the system (if for instance AES ECB is used, one could gain information, or try to perform replay attacks).

Q12. Do you have Over-The-Air (OTA) firmware update?

Providing an OTA firmware update mechanism allows to keep the devices up to date. It however needs to be implemented securely to prevent opening the door to attacks. The update must be transmitted securely, signed by an authorized trust entity and encrypted using proper encryption methods. The firmware must be checked by the device before proceeding to the update.

Q13. Is the firmware update automatic?

Without automatic update mechanism, devices' firmware will rapidly be outdated, allowing for known (and publicly available) vulnerabilities to be exploited by anyone, without requiring much knowledge.

Q14. Does the firmware contain any sensitive data (e.g. Hardcoded credentials)?

If it is the case, an attacker who gets his hand on a firmware file can retrieve the credentials and potentially compromise the ecosystem at scale.

Q15. Are there any mechanisms to prevent against weak, null, or blank credentials?

Such weak credentials allow an attacker to easily gain access to a device and/or service.

Q16. Do you have mechanisms to isolate privileged code, processes, and data from portions of the firmware that do not need to access them?

This prevents an attacker who can inject unauthorized code into a running "process" to access sections of the firmware that contain sensitive data (such as cryptographic material).

Communication

Q17. Is the information transmitted over serial line protected?

An attacker might want to access sensitive information and/or be able to fake the data sent by a device. Eavesdropping serial communications (such as communication between the

[2] True Random Number Generator and Pseudo Random Number Generator

microcontroller and a modem) requires very little knowledge and equipment, it is thus important to ensure that no sensitive information can be accessed/tampered with there.

Q18. Is the technology/protocol used for wireless communications secure in your use case?

Some protocols might have a secure configuration, but it might impossible to reach such a configuration in your use case: for instance, while Bluetooth can be secure, if there is no way to establish a secure key exchange during pairing (using a pin for instance), there will be a risk of Man in the Middle attack (MitM).

Q19. Are you implementing mutual authentication?

Without mutual authentication, an attacker can potentially act as a MitM and steal information or inject fake data into the system.

Q20. Are you exposing any credentials to internal or external traffic?

Sending credentials over an insecure communication channel (for instance GPRS[3]) could result in the credentials being intercepted by an attacker and further used to gain access to a backend server and/or service.

Infrastructure

Q21. Is the access to the backend infrastructure protected?

If possible, restricting access to the backend infrastructure is a good practice. For example, if devices always connect to a third-party communication provider's network to gain connectivity (Sigfox, or a mobile operator for instance), the backend infrastructure can be configured to only accept communications incoming from that network.

Q22. If you have an OTA firmware update mechanism, is the update server secure?

An insecure update server could result in device compromission at scale (an attacker uploading its own modified firmware for instance).

Q23. Do you have any mechanism to detect a compromised device?

A compromised device might behave outside the "expected range of operation". For instance, it could connect and send data at a different interval than the regular one, send data that should not be physically possible, etc. This can be detected in order to flag such devices or even disable their access.

Q24. Do you have any mechanism to detect a rogue device?

Attackers might have interest in introducing rogue devices to a system, i.e. unauthorized devices which pose as legitimate devices but might have enhanced and malicious capabilities.

[3] General Packet Radio Service

Q25. Is the backend infrastructure prepared to handle data flooding?

An attacker who compromised one or more devices could start sending data to the backend at a much higher rate than expected. If the data is accepted by the backend, depending on the configurations, this could lead to data loss or extra costs (when using databases from a third-party cloud provider for instance). Having DDoS protection and load balancer is important to prevent this as well.

Q26. Do you implement rate limiting?

Without rate limiting, an attacker who gained access to a device could misuse it outside of its planned operation range.

Generic considerations

Q27. What is the impact of a single compromised device on your system?

Assuming one device is compromised by an attacker (for instance, the attacker got their hands on a device and extracted the firmware and cryptographic material), this can have severe consequences on the whole ecosystem if some cryptographic material is shared.

Q28. Have you considered the operational consequences of one device being compromised? What about N devices?

While a single device being compromised might not have a big operational impact, a few devices could start to have an operational impact depending on the use case.

Q29. Are you collecting and storing only the minimum data required?

In the case of a device collecting data, it is important to minimize the data collected and retained (especially in the case of personal data).

Additional resources

https://www.enisa.europa.eu/publications/hardware-threat-landscape
https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot
https://github.com/OWASP/ASVS


SINTEF
Technology for a better society