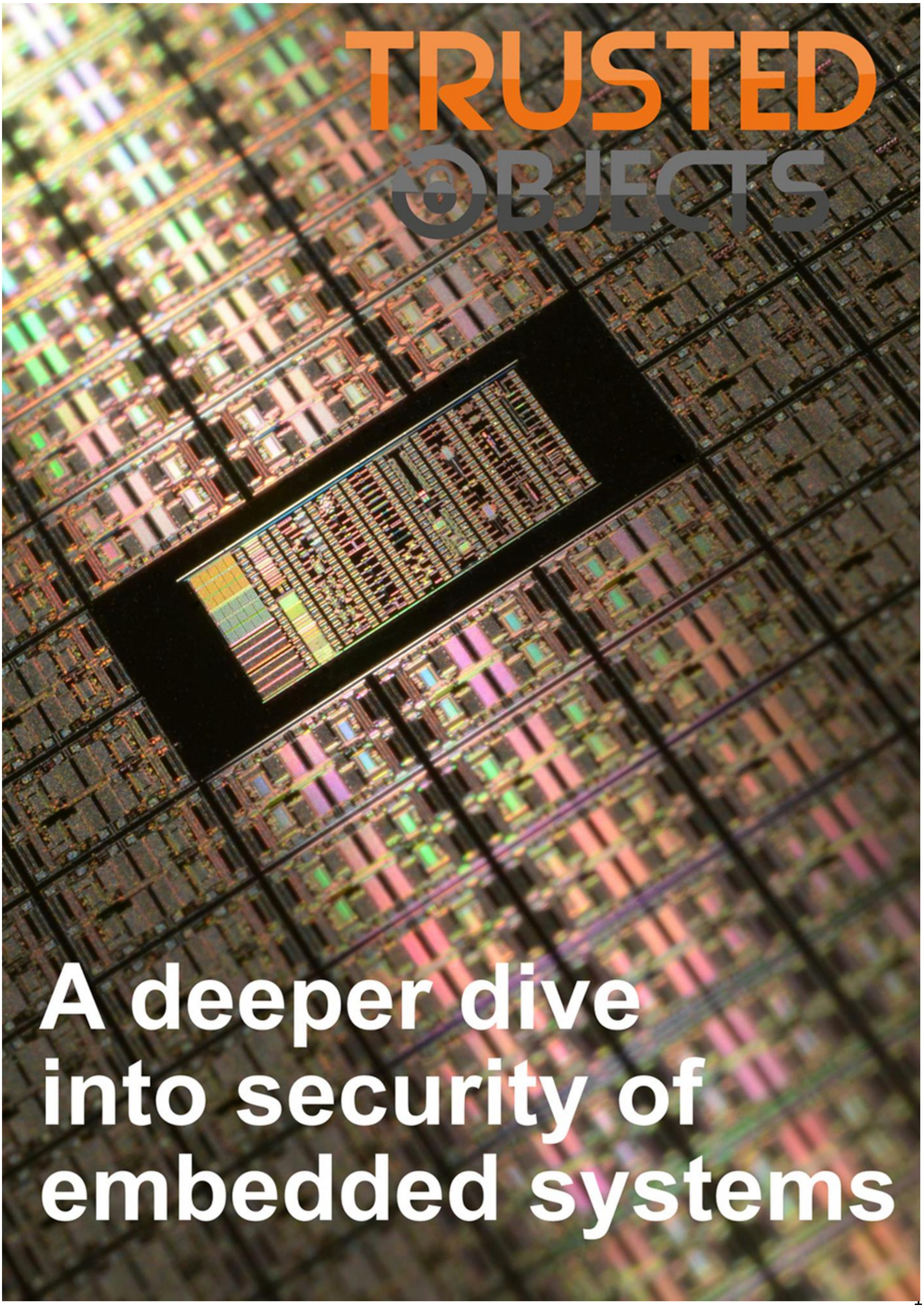
TRUSTED OBJECTS



A deeper dive into security of embedded systems

By:

Vincent Dupaquis – Software Security and Cryptography Expert at Trusted Objects

François Bonnal – Embedded Security, PhD at Trusted Objects

Henri-Michel Thébaud – Security Application Engineer at Trusted Objects

SUMMARY

1-Introduction

1.1 Scope and purpose of this white paper

The scope of this white paper is the security for embedded electronic systems and IoT systems, which are generally based on programmable microcontrollers. Examples are electronic consumer and industrial devices, IoT sensors, medical devices,…

The purpose is to stress the fact that although security countermeasures are necessary to protect embedded systems and IoT systems , they are unfortunately not sufficient to avoid surface attacks. Embedded systems and IoT systems are more and more exposed to a wider range of new security threats, and this trend will very probably accelerate.

To prevent damages from security attacks, Companies are taking measures to protect their assets, including more specifically their software IP. Unfortunately, in ecosystems where the supply chain is getting more complex, it is frequent that the ones deciding the security levels are not the ones that will be accountable for their choices.

Even when security measures have been duly selected and implemented, facts are showing that there are still some underlying vulnerabilities. On average, security experts will break security of more than 80% of implementations during their evaluation phase, for multiple reasons:

- Security attacks are getting easier to set-up, even by players who have limited technical skills and could use tools available on the web. It costs just a few dollars to launch massive DDoS attacks capable of generating up to 300Gb/s.
- Security countermeasures have their own limitations, and having an overreliance on those countermeasures could lead to potential hidden security risks.
- Security implementation matters! Technical challenges in implementing security could potentially lead to vulnerabilities exploited by hackers.

A good approach is to do a formal security evaluation with security experts. However, before taking this path, it will be efficient and cost effective to have a second view with a deeper dive into security. In most cases, it will highlight some vulnerabilities and will provide useful guidelines to improve the resistance of embedded systems against security attacks.

A deeper dive into security before security evaluation:

In this whitepaper, we will:

- describe the most frequently used security countermeasures.
- review the limitations of these countermeasures and explain why a deeper dive is recommended.
- share the views from our security experts.

The benefit of this deeper dive is to reduce exposure to security attacks without having to reconsider the whole security approach.

1.2 Security principles

Basic principles

It is widely accepted that security must rely on 3 basic principles:

- security by design (*and not after the facts*)
- end to end security (*at OT and IT levels*)
- security all along the product life.

The last one is equally important compared to the first two; we observe that several electronic industries are getting conscious about the security by design and end to end security, and are not considering the importance of security all along the product life. For instance, having a secure mechanism for firmware update over the air (OTA) will prevent a lot of security breaches.

Deeper dive

It would be great if a simple application of those basic principles will be enough to counter any potential security attack. Facts are showing that even by applying those principles, there are still remaining vulnerabilities exploited by hackers.

Embedded systems are all different and have their own specificities; on the other side, security requirements vary considerably depending on market, applications or risk management policies.

Considering that security must be scalable, and that no security scheme fits all, we recommend a deeper dive into security to ensure that the security schemes have been implemented in adequation with the system architecture. A strong security scheme which has not been properly implemented is simply useless.

We will explain in this white paper the reasons why these basic security principles are necessary and not sufficient.

1.3 Disclaimer

The information in this white paper provides general information and guidance about cybersecurity; it is not intended as legal advice nor should you consider it as such.

2-Why does a deeper dive into security make sense?

2.1 Security attacks on embedded systems are getting more frequent

There are several reasons that could explain why embedded and IoT systems are getting more vulnerable to security attacks:

- Systems complexity:
  Embedded and IoT systems are becoming more and more complex due to rich, broad and diverse ecosystems which could be interconnected with each other's. IoT ecosystems are an illustration of this trend; they include a wider range of technologies like sensors, gateways, networks, clouds with many different standards and limited regulations on security.

- Limited capacities in devices:
  Many embedded and IoT systems are based on programmable microcontrollers with limitations in processing power and memory storage. Several security countermeasures have not been designed based on those limitations. As a result, they require compromising between security and performance, and most of the time the decision is in favor of the last one.

- Human errors are always possible:
  The development of new technologies is accelerating, and we do not have enough background of previous threats to know enough about failures in protection. This is leading to an increase of human errors in life of a product: at the design stage, at manufacturing stage and during the implementation of security.

- Time to market and costs
  Generally, manufacturers shorten the launch time of products, putting higher priorities on volume of sales, and not always considering fundamental security best practices such as security by design. Security is often seen as an additional cost; this is why, in order to reduce costs, manufacturing companies are also limiting or ignoring security features in their devices. The result would be equipment that can never provide adequate protection.

2.2 Any countermeasure has its own limitations

Deciding a security strategy often means making compromises between risk, cost and time; the easier approach is to rely on legacy security mechanisms proposed by silicon and IP vendors, network providers or other third parties in the value chain. The issue with this approach is that there is no "one size fits all" security solution that can protect any embedded system.

The characteristics of each system is different and should be considered when deciding the security scheme; without those considerations, the security scheme could be much less efficient than expected, and will result in vulnerabilities.

Even when countermeasures have a robust and proven efficiency, their implementation process requires a high degree of expertise which could lead to gaps between the system architecture and the security scheme. Unfortunately, many companies still put off doing complex verification steps to ensure the security efficiency in the long run. They don't realize security risks until it's too late.

2.3 Vulnerabilities are coming from unsecure implementations

Cyber security technologies are not efficient against cyber attackers if they are not properly implemented. For instance, using encryption technologies is often perceived as a guarantee of strong security. However, encryption is useless if the encryption keys can be easily discovered, hardcoded or guessed.

LoRaWAN protocol is an illustration of information which could be misleading if the security is not properly implemented. The protocol is, by design, very secure, with mandatory authentication and encryption. In fact, researchers and ethical hackers have found the root keys usually used for encrypting communications between devices, gateways, and network servers are poorly protected most of the time. The result is that the keys are easily obtainable through different methods by malicious hackers who may use these keys to send false data and conduct denial of service attacks.

Another example of unsecure implementation comes from security by isolation mechanisms. Those mechanisms are widely adopted in embedded systems architecture. However, the complexity of implementing these mechanisms could result into mistakes and gaps with the architecture of the devices, which can then be exploited by hackers.

3-Frequently used countermeasures against security threats

3.1 JTAG

What is it?

The JTAG (also known as SWD) is a connection to the microcontroller which is used for in-circuit debugging, allowing to directly communicate with the memory registers, and also for programming devices. This interface is very well known and widely adopted.

What are the risks with JTAG, and what are the countermeasures ?

As it is a debug interface, JTAG can be used to dump memory, debug code or reprogram entirely or partially a device. For security experts, JTAG is the main entry door for attackers.

In order to prevent those attacks, JTAG has to be sealed before deploying products into the market and should never remain accessible on a developed product. The semiconductor industry is conscious about this potential weakness, and is proposing two different approaches:

1. For highly-secured products (such as Secure Element for Banking Cards or e-Passports), the JTAG has been removed.

2. For standard microcontrollers used in applications like Consumer, IoT, Automotive, the JTAG is left and can be sealed during the production phase. This sealing operation is performed using “fuses”, which can only be blown once, and never reset unless erasing the whole memory.

Why a deeper dive?

Having sealed the JTAG is a precautionary measure; however, it is as important to do a final verification at the very end of the production process, and ensure that it has effectively been sealed. Else, software IP may be exposed to attacks.

Even by applying this precaution and doing the final verification, the system is still exposed to security attacks. Sadly, hackers can bypass those fuses by using “perturbations” or “faults”, with tools that can be found on the net. For instance, useful information about breaking into a chip’s security can be found by following the link :

https://www.riscure.com/beauty-and-the-glitch

Recommendations from our experts:

Sealing the JTAG is necessary, and not sufficient. We recommend to setup a specific strategy, protecting IP against hacking, even in the case the JTAG protection has been circumvented. Considering an additional security layer on top of sealing the JTAG will ensure to be on a safer side. It can be achieved by using lightweight cryptographic primitives with a secure library software.

3.2 Bootloader

What is it?

A bootloader is a piece of code/program that runs before an operating system starts. A bootloader is also used to re-program a firmware running on an embedded device.

Some bootloaders allow to transfer the content encrypted and authenticated, and may prevent code dumping and malware injection in the microcontroller; we call them “secure bootloaders”.

There are usually two categories of bootloaders in an embedded device:

1. The bootloader installed by default by the microcontroller manufacturer : it is always present and cannot be removed or modified.

2. Software bootloaders, which are an alternative to replace and disable the manufacturer’s bootloader. They can provide better security features and offer a possibility to be updated and adapted to new threats.

What are the risks with bootloader ?

It is possible to break the code protection on microcontrollers by bypassing the bootloader. The following link hereafter shows for instance, how it is possible to break code protection on microcontrollers:

https://toothless.co/blog/bootloader-bypass-part1

In this example, the mechanism is supposed to offer different levels of protection, the highest one being to completely disable the possibility to modify or read the firmware or to access debug interface. However, the level of protection is checked at the start of the device and the poor implementation of this test allows disabling any level of CRP protection with a cheap fault injection setup. This attack does not require a specific technical expertise.

Why a deeper dive ?

The bootloader installed by default cannot be removed or updated. It is by construction a breach against one of the key principles of security which is the security all along the life cycle of a product. Even when this bootloader is disabled, it is a door open for hackers who can bypass it with cheap material and limited knowhow.

Recommendations from our experts:

Considering an additional security layer on top of a bootloader will ensure to be on a safer side. It can be achieved by using lightweight cryptographic primitives with a secure library software.

3.3 Security by separation

What is it?

Security by separation consists in isolating the execution of security critical operations in a specific zone.

Implementing security by separation in embedded systems could be challenging.

The concept of security by separation has been historically developed for smart phones, and several commercial implementations have been used over the years.

Embedded systems based on standard microcontrollers have different characteristics, and security by separation may reveal some limits by not being directly applicable to the ecosystem, or simply requiring too much code or data space.

Why a deeper dive?

A poor implementation of a security by separation will result in the presence of security flaws that are easily exploited by hackers. For instance, the security by separation mechanism does not bring sufficient protection against side-channel and faults injection vulnerabilities.

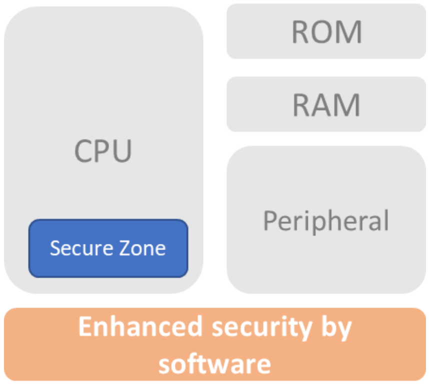
A lot of research has been conducted on the limitations of security by solation:

https://www.researchgate.net/publication/325874920_How_TrustZone_Could_Be_Bypassed_Side-Channel_Attacks_on_a_Modern_System-on-Chip

Recommendations from our experts:

The best practice is to perform a review of the implementation and check whether the implementation corresponds to the architecture. Such a review must be conducted by security experts; it will lead to additional costs, and some delay in device development must be anticipated.

The alternative is to harden the protection against physical attacks by adding another layer of security, in complement of the security by separation mechanism.



For instance, a secure library embedded into the microcontroller will dramatically enhance the resistance to side channel attacks. Even if the security patterns have not been correctly implemented, a secure library will enable security functions that will limit the exposure of security attacks such as side channel attacks.

3.4 Communication protocol in embedded systems

What is it?

The communication protocol is allowing a specific device to transmit information to another system. It plays an essential role in term of security and represents one of the main entry doors to device assets and software IP. Embedded systems use various communication protocols such as I2C, SPI, USB, UART…

What are the vulnerabilities and countermeasures?

Hackers can use vulnerabilities in the communication protocol to recover software and code IP. For instance, through fault injection attacks and using the protocol vulnerabilities, hackers can get access to the data transmitted through the protocol. An easy way to dump the code!

A good practice is to secure the communication protocol by making sure that the security of the data is preserved over the protocol.

Different methodologies and techniques can be applied: choosing standardized security protocols on top of the communication protocol (for instance TLS, DTLS…), or alternatively securing a proprietary protocol.

Regardless of the choice made, the confidentiality, authenticity and integrity of messages transmitted on any protocol must be achieved using a robust protocol based on proven cryptography primitives.

For instance, a proprietary protocol could be secured using an AES algorithm 128-bit keys. This symmetric encryption process requires a shared secret embedded in the device and used each time a message is encrypted.

Why a deeper dive?

At this stage, it is normal to think that a strong security level is achieved and that the communication protocol will not be used anymore to recover software and code IP.

Actually, the protocol security level based on cryptography primitives depends on the implementation of these primitives embedded in the micro-controller. Without a secure implementation of the cryptography, the protocol protection can be easily hacked and all data going through the communication interface can be compromised.

Attacking a cryptography algorithm based on an unsecure implementation can be done running physical and non-invasive attacks, for instance the side-channel attacks, in the sense that hacker do not need to interfere with the device. Hackers target the keys recovery when they are used during the cryptographic algorithm processing.

Unfortunately, such attacks are quite easy to carry out and they are well documented on the web with open-source software and low-cost hardware like Chipwhisperer.

ChipWhisperer is a NewAE Technology Inc's fully open-source (hardware, software, firmware, and documentation) toolchain for self-teaching about embedded hardware security. The ChipWhisperer-Lite is the low-cost hardware which cost is around 250$.

Recommendations from our experts:

That highlights how algorithms implementation aspects are important to achieve a robust security level and to avoid the code recovering through the communication interface. Defeating such attacks require a secure implementation.

Secure libraries based on secure implementation of cryptography algorithm and secure secrets storage with obfuscation mechanism, have proven to be efficient against side-channel attacks and reverse engineering. Moreover, those libraries are quite easy to implement.

3.5 Memory protection

What is it?

Memory protection is a mechanism which helps protecting parts of the embedded code against dumping.

The mechanism consists in an additional layer of memory protection, coupled to specific compiler settings (for instance, activating the "-mpure-code" option from GCC) and application organization, which allow (for instance) to block code read, and only allow its execution. Generally, these protections are set-up by blowing some fuses, the same approach which is used for JTAG or Bootloader.

Why a deeper dive?

Memory protections are theoretically quite efficient as they are executed at the hardware level. However, the implementation could be complex; it requires to dig deeper into the firmware architecture, and rework major parts of it. In some cases, it may result in a slower implementation and a bigger footprint of the code.

Memory protection techniques are also specific to each microcontroller supplier; it means that, depending of the code, the implementation could become challenging.

Recommendations from our experts:

Memory protection is a good solution, assuming the challenge to implement it is not too high. However, we recommend to use it as an additional layer of protection and not as the only defense against security attacks.

It is also important to note that security protection mechanisms are hardwired, and it will not be possible to upgrade or improve it. In a changing security landscape, the hacker's expertise is improving rapidly, and inherent security mechanism will offer poor resistance in the future if not enhanced and updated regularly.

4 Conclusion

For decades, the digital security industry has protected our critical assets and identities by applying security standards and best practices, by proceeding to security evaluations and by getting security certifications.

For many industrial applications, it will be difficult and expensive to follow the same path. On the other hand, blindly relying on security schemes without challenging the security robustness will lead to hidden risks and consequently create huge damages and financial losses.

Having a deeper dive into security is a way to ensure that there are no major security breaches in embedded systems.

Trusted Objects is an independent company, founded by digital security experts, and specialized in cybersecurity technologies for embedded systems. Our vision is that security-by-design, end-to-end security and life cycle management are the key principles to build a better and safer connected world.

Trusted Objects and its partners share a mission for helping our customers protect their sensitive data and critical intellectual property with easy-to-integrate, scalable and cost-effective industrial security solutions.

We invite the readers to do a deeper dive into their security, and get the benefits of a second view with our experts.

In our next white paper, we will speak about security at the manufacturing stage.

See you soon !

5-Glossary

- **CPU**: Central Processing Unit
- **MCU:** microcontroller unit
- **DDoS**: Distributed Denial of Service
- **IoT**: Internet of Things
- **Embedded System**: Electronic device based on MCU
- **SoC**: System on Chip
- **IP**: Intellectual Property
- **Software IP**: Intellectual Property in software form
- **IT & OT**: Information Technology, Operational Technology
- **TEE**: Trusted Executive Environment
- **JTAG**: Joint Test Action Group
- **Bootloader**: a software that is responsible for booting a computer
- **Secure bootloader**: a bootloader with security mechanisms
- **Side Channel attack**: a technique to extract secrets from a chip or a system, through measurement and analysis of physical parameters.
- **Fault injection attack**: a technique to break code paths by introducing faults