HighIntegritySystems

WITTENSTEIN

Increasing Security in Medical Devices

Issue 1.4 - May 31, 2018

WITTENSTEIN high integrity systems
Americas: +1 408 625 4712
ROTW: +44 1275 395 600
Email: sales@high**integrity**systems.com
Web: www.high**integrity**systems.com

Contents

List of Figures

List of Notation

BSP Board Support Package
COTS Commercial off-the-shelf
DAP Design Assurance Pack
DHF Design History File
MCU Microcontroller Unit
MPU Memory Protection Unit
MMU Memory Management Unit
RTOS Real Time Operating System
SIL Safety Integrity Level
SOUP Software of Unknown Provenance

CHAPTER 1 Introduction

Connected medical devices have many benefits. They offer opportunities for continuous monitoring, telemedicine and big data analytics to uncover hidden trends.

A quick search of the internet for medical device security breaches will result in multiple reports of vulnerabilities in insulin pumps, infusion pumps, cardiac defibrillators and pacemakers: with connectivity there is always a risk that bad actors could gain access to medical devices, with potential life or death consequences.

The bad actors' advantage is that they only need to find one successful attack vector, whereas the defender must defend all possible attack vectors. As attacks evolve and get increasingly novel, aggressive, sophisticated and frequent, defences must be continually refined, improved, strengthened and hardened.

It should be assumed that medical devices operate in a hostile environment, with medical device developers expecting constant attempts at intrusion. Therefore products should be hardened to meet this threat…. but how?

This white paper will address the first steps to take when developing security software for medical applications, through use of standards, the development life cycle, and common security mechanisms.

CHAPTER 2 Standards

There are limited guidelines and standards for developing secure systems. The FDA has issued several sets of guidance focusing on medical device cybersecurity, recommending manufacturers use the National Institute of Standards and Technology Framework for Improving Critical Infrastructure Cybersecurity.

A useful security standard that can be considered for medical device security is Common Criteria.

Common Criteria is a widely recognised international scheme used to assure security-enforcing products. It provides formal recognition that a developer's claims about the security features of their product are valid and have been independently tested against recognised criteria, to a formalised methodology.

Common Criteria requires a Security Target, which defines, amongst other things, the security objectives and the environment the device will work within. The system is then designed and verified according to a specific Evaluation Assurance Level or EAL, of which there are 7 altogether (see Fig 2-1). For each EAL there is guidance on the methodology - the design, verification and penetration testing should follow. The higher the EAL level, the greater the effort required, which should result in greater determinism and security for the device under test.

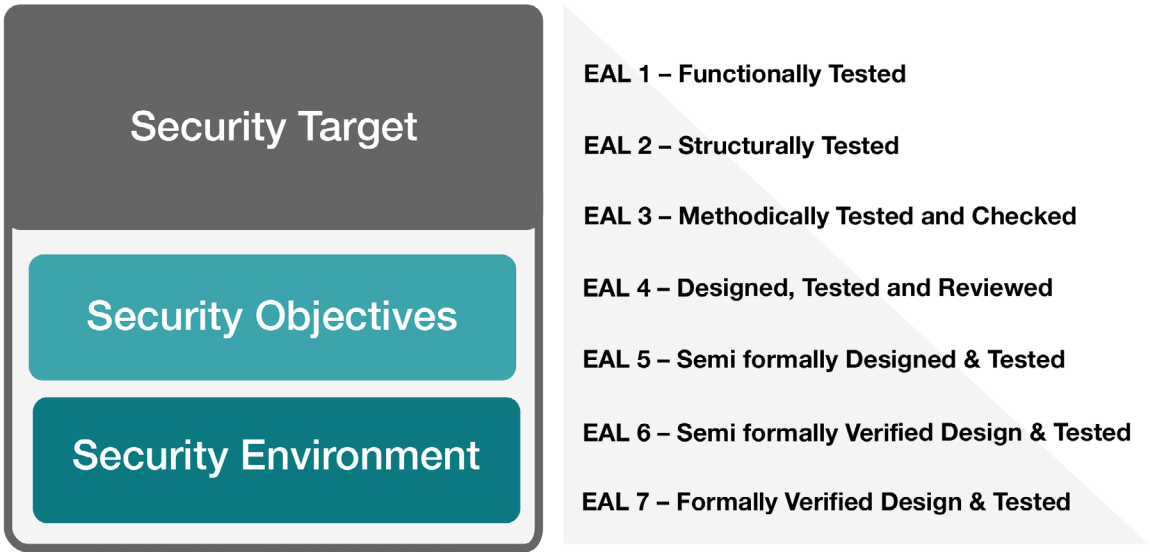

Figure 2-1 *Common Criteria*

Certifying a medical device to Common Criteria has its challenges. However the guidelines and methodology used by common criteria could be of use to medical device developers concerned about security, even if Common Criteria is not adopted in full.

CHAPTER 3 Safety Development Life Cycle

All medical device developers are familiar with safety, but for many, security is a new subject. The safety development lifecycle is a valuable tool that can be used to build in both safety and security.

Safety and security share many properties, however these similarities can be dangerous, and if handled incorrectly can result in the wrong design decisions being made. Figure 3-1 shows high level representation of a safety critical design life cycle, commonly referred to as the 'V model'. The next few sections provide examples of how this development life cycle can be altered to design in security as well.

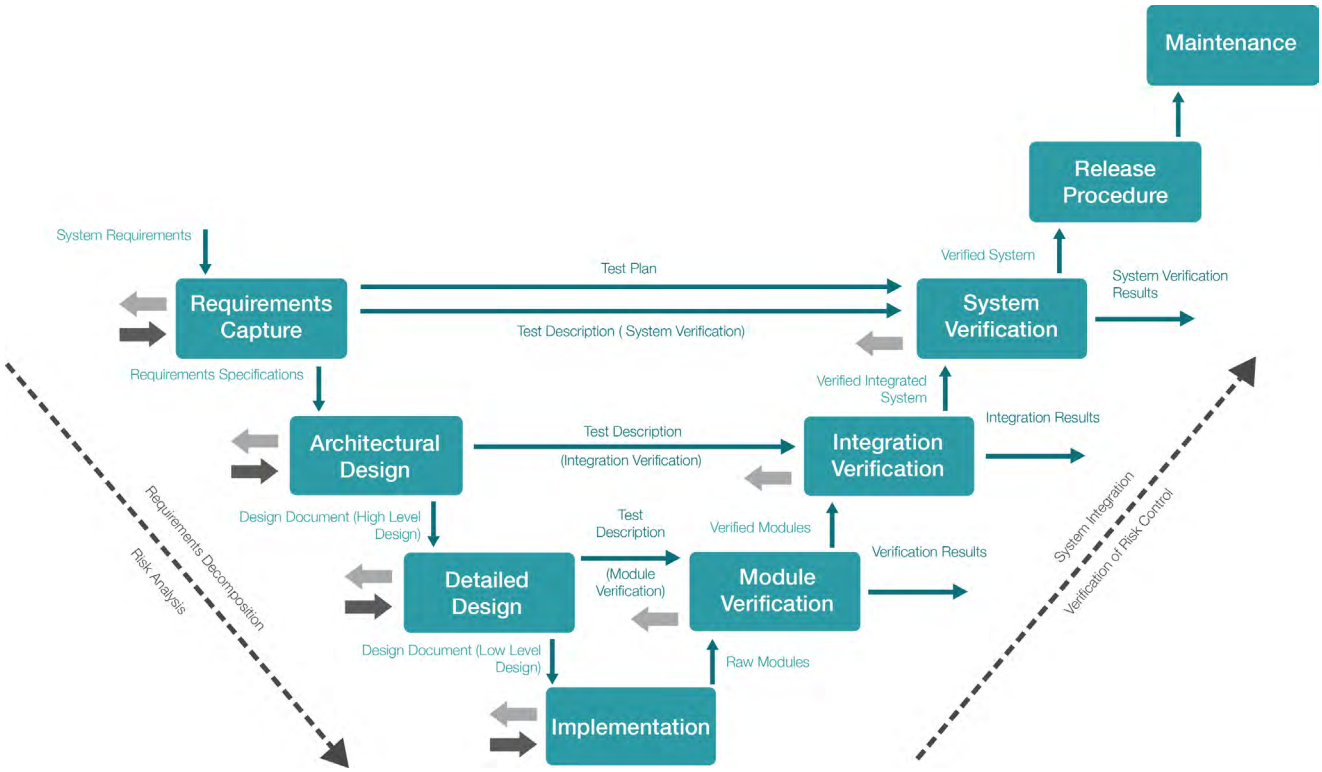

Figure 3-1 *The "V" Model of the Safety Development Life Cycle*

3.1 Requirements Capture

3.1.1 Attack Surface Analysis

Within the development life cycle, security and safety require similar, yet different processes. To identify safety requirements you need a highly structured technique such as a HAZOP Analysis, whereas to identify security requirements you need to ignore the limitations of the structured environment and consider all contexts that are possible in an Attack Surface Analysis.

The Attack Surface Analysis is used:

- To understand the risk areas in a medical device;
- To make developers and security specialists aware of what parts of the medical device are open to attack;
- To find ways of minimising this;
- And to notice when and how the Attack Surface changes and what this means from a risk perspective.

In some ways designing for safety is easier; the device works in a known context, and if the context moves outside of its defined operating parameters, then the system is placed in its safe state. Security is more difficult to ensure, as the attacker is attempting to manipulate the system context to gain access to it. The danger is that a context that was impossible to conceive during design is presented to the system, resulting in unwanted side effects which might leave the system open to attack.

Different as they are, safety and security cannot be considered in isolation. There are unwanted interactions between safety and security, and sometimes this can include unwelcome side effects. For example, an alarm system that automatically opens the doors to allow evacuation of a building could also be manipulated to allow an attacker to gain access to the building.

In a high integrity design life cycle the Attack Surface Analysis could be performed in parallel with the HAZOP, resulting in a set of security and safety requirements respectively. Safety requirements should be subjected to a security review and security requirements should be subjected to a safety review.

3.2 Implementation

3.2.1 Coding Standards

Most developers of safety critical software are familiar with the MISRA C coding standards. Its aims are to facilitate code safety, security, portability and reliability in the context of embedded systems, specifically those systems programmed in C by providing a restricted subset of a standardised C programming language that is deemed to be safer.

There is an alternative coding standard for secure programming called Cert C, from the Software Engineering Institute (SEI). Cert C details coding guidelines for those developing software that requires a degree of security. Cert C provides rules and recommendations which, when followed, result in less vulnerabilities for the bad actor to exploit.

There is a growing trend to use Cert C within secure systems.

3.3 Verification

3.3.1 Fuzz Testing

Testing for security requires a slightly different approach.

Fuzz testing is used to identify hard to predict security vulnerabilities within the software.

Fuzz testing involves providing invalid, unexpected, or random data as inputs to a software module or application. The module or application is then monitored for exceptions such as crashes, or failing built-in code assertions, or for finding potential memory leaks.

An effective fuzz test will use semi-valid inputs that are “valid enough” in that they are not directly rejected, but do create unexpected behaviours deeper in the application and are “invalid enough” to expose corner cases that have not been properly dealt with.

3.3.2 Penetration Testing

Penetration testing is used to evaluate the security of the medical device. The test is performed to identify both vulnerabilities, including the potential for unauthorised parties to gain access to the system's features and data as well as strengths, enabling a full risk assessment to be completed.

A penetration tester will identify a specific security objective, then will review all available information to construct tests that attempt to undermine the security objective. It may seem unfair that the tester has access to the design documents, but there's no guarantee that a bad actor won't also have access to the design documents.

3.4 Maintenance

Having software that is both safe and secure is an interesting maintenance challenge. Safety software takes a long time to develop and verify, is robust and reliable, and consequently is rarely updated. This is in direct contrast to the security threat, which is constantly evolving, with attacks growing in complexity as hackers learn and develop knowledge on how to exploit the software. This requires software to be regularly updated to counteract attacks.

The software design should take this into consideration, whereby the dependencies between the safety and security code are minimum, and the security code can be updated with little risk to the safety of the system.

CHAPTER 4 Common Medical Device Security Mechanisms

Now that vulnerabilities have been highlighted during the attack surface analysis, what can be done about them? This section focuses on addressing and mitigating typical security concerns that could be identified during an Attack Surface Analysis.

4.1 Root of Trust

Medical device security relies on a series of nested objectives. As these objectives build on each other, it's vital the initial objective is implemented in a highly robust way, which is very unlikely to be compromised.

In most instances the initial security objective of a medical device is to confirm that the device that has booted is the correct device, and that the software of the device has not been compromised. This is normally referred to as the root of trust.

The root of trust is typically implemented using a dedicated security module implemented entirely in hardware, or a combination of hardware and software integrated in a security coprocessor. This creates a secure mechanism for developers to lock down their code. Each time the system boots, the system validates the digital signature of the booting code. Only proven, authentic software can execute. Attacks that modify the system software are detected and blocked.

The actual code booted depends on the application. It could be that just a boot loader is initially authenticated and installed, and this boot loader will systematically authenticate and load other elements of the system. In other, typically smaller applications, it could be the complete application that is loaded.

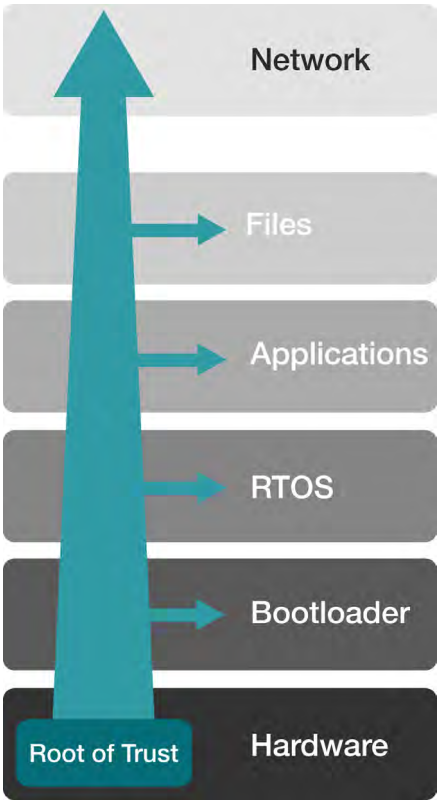

Figure 4-1 *Root of Trust*

4.2 The Digital Signature

The digital signature is a mathematical scheme for demonstrating the authenticity of digital data. It can provide authentication, non-repudiation and integrity. Authentication is where a valid digital signature gives a recipient reason to believe that the data was created by a known source. Non-repudiation is when the sender cannot deny having sent the message. Finally integrity is where it can be seen that the data was not altered in transit or during storage.

The digital signature is typically created using a private/public key cryptography, shown in figure 4-2. First, a hash value is calculated which uniquely represents the data being protected. This hash value is signed with the private key that identifies the source of the data. The reader of the data will calculate its own hash value for the data, and authenticate the digital signature stored in the data stream using a public key associated with the private key. If the result of the authentication process matches the hash value calculated for the data, then it can be assumed the data has not been altered, and it has been created by the owner of the private key.

Private/Public key cryptography provides a high degree of security as only the public key has to be distributed within the medical devices and does not need storing in a secure, private location, whereas the private key can be stored securely off line. The weakness lies within the public keys - if in the unfortunate event that the private keys are compromised, then a secure method would be required to update the public keys within the medical devices.

Please note, Private/Public key cryptography methodology does not encrypt the data that is being protected.

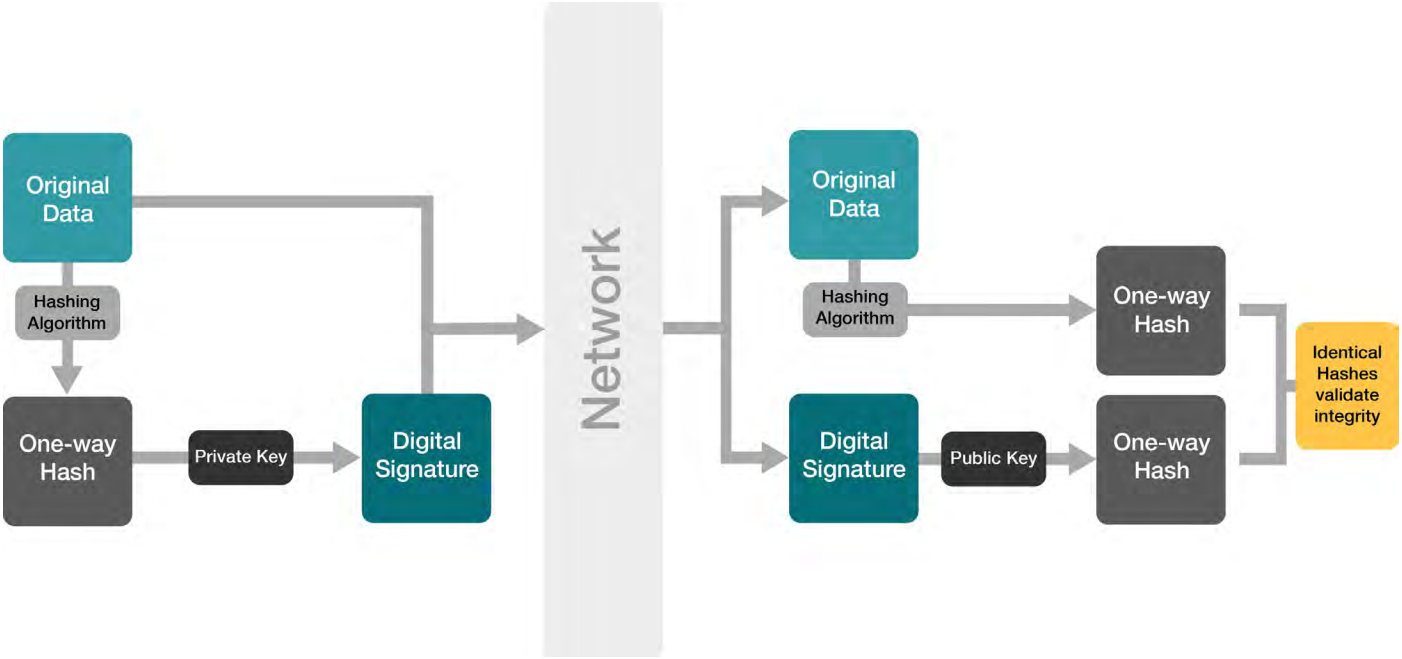

Figure 4-2 *The Digital Signature*

4.3 Creating a Greater Trusted System

Medical systems need to authenticate all connecting devices and sources of data, building a system whereby the true identity of all devices and data sources in the system can be validated.

For example, a medical device may have attachments used to administrate treatment. The medical device has to ensure that only data received from approved attachments can be used, and in doing so prevent the use of malicious, clone and counterfeit attachments.

Likewise if the medical device has internet connectivity. The medical device needs to demonstrate it is the true source of the data transmitted. This guards against 'Man in the Middle' attacks, where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other.

This can be achieved by adding a digital signature to all messages transmitted. If the receiving device can authenticate the digital signature, then the source of the message can be confirmed, and also its integrity.

Building a trusted system relies on each device in the chain being able to authenticate the devices connecting to it.

4.4. Encryption

Encryption is used to protect data at rest or in transit. Data at rest could include highly confidential patient records, or the code image. Encryption is used so that attackers can't discover the plaintext of a message, or stored data.

Encryption and a digital signature can be used in combination, seen in Figure 4-3. Typically data is encrypted, then signed with a digital signature. If the receiving device can authenticate the digital signature, then the source of the message can be confirmed, and also its integrity. Then the message or data object can be decrypted.

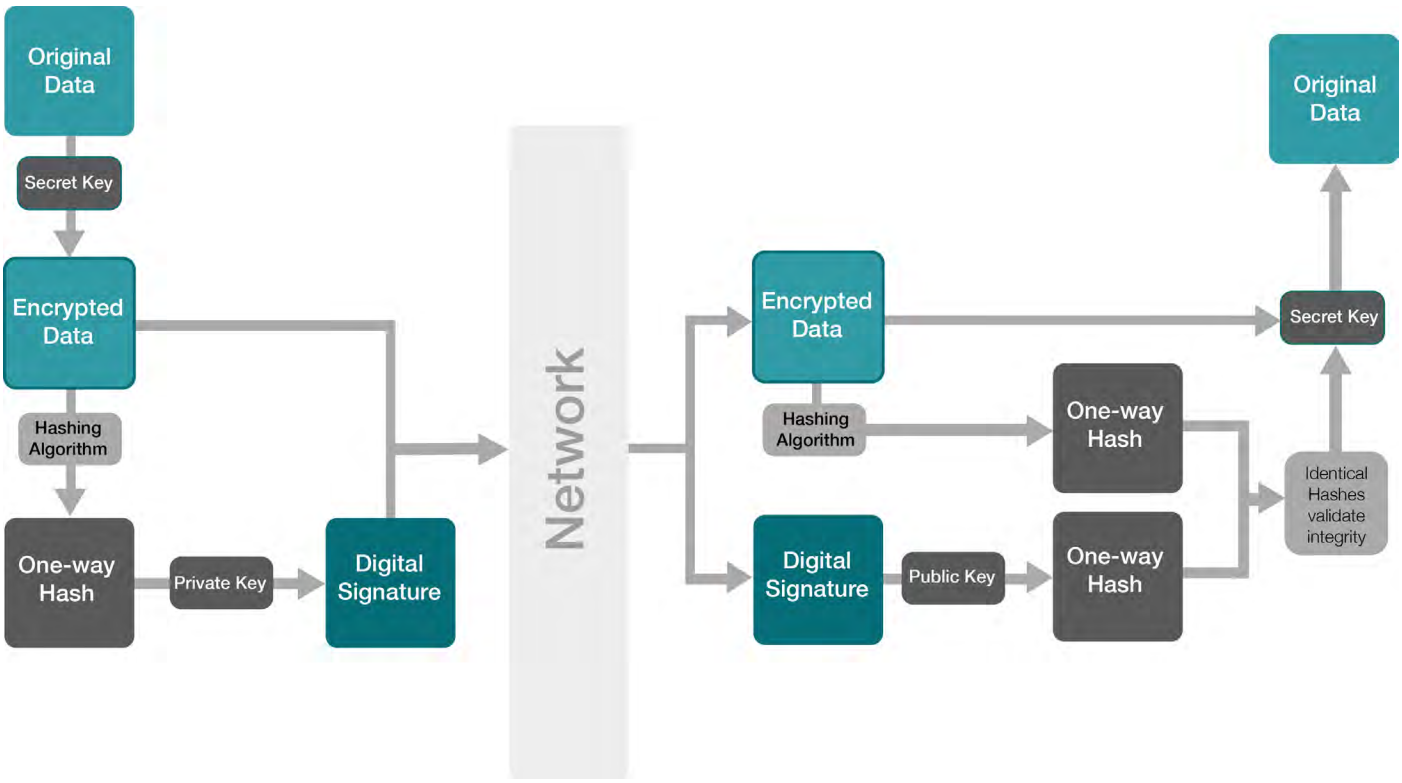

Figure 4-3 *Encryption*

4.5 Software Updates

Software updates and patches allow medical devices to stay current, but can also introduce vulnerabilities. Software updates must be delivered from an authenticated source, with an attached digital signature used to authenticate the code image, keeping systems in the field from being tricked into accepting malicious updates

Additional features are required to close rollback vulnerabilities. Rollback is an attack strategy in which the attacker attempts to undo security updates by tricking a system into executing an authentic, but old and vulnerable, version of system software.

The system will also need to confirm that the latest version of the software is valid for its configuration.

4.6 Reducing the Attack Surface

The smaller the attack surface, the less opportunity bad actors have of gaining control of the medical device.

All data streams into the system that don't authenticate and encrypt data must be disabled. This prevents bad actors circumventing the security measures already put in place and gaining easy access to the medical device via a back door. It may be quite easy to forget to disable a port used for monitoring and setting the internal status of the medical device used during its validation process.

Enabled JTAG ports offer an easy way into the system for bad actors that have acquired the medical device. JTAG is a common hardware interface that provides a way to communicate directly with a processor. JTAG is an extremely powerful interface to embedded devices, and provides access to almost all of the processor's functionality, allowing bad actors to do almost whatever they like. To guard against this type of attack, the JTAG interface should be removed from the board. It is best if the JTAG interface can be disabled completely.

Unused or undocumented code sections may offer a back door into the system that no one is aware of. Therefore it's important that all code segments can be traced back to a requirement, and that each requirement has a test case associated against it. The test case must test each and every path through the code.

As an example, for SAFE**RTOS**® we use Modified Condition/Decision Coverage, MC/DC for short, as a metric to demonstrate that all paths through the SAFE**RTOS** code have been verified. 100 percent MC/DC coverage means that there is no unused or undocumented code within the device itself and every path through the software has been tested.

On smaller devices, running from FLASH rather than RAM provides a more secure option.

4.7 Partitioning of Software

Building in partitioning makes it harder for bad actors to obtain full access to the medical device.

High security functionality - including key management - is typically implemented on a dedicated security module built entirely in hardware, or a combination of hardware and software, integrated in a security coprocessor. Access to the security module is achieved via a restricted interface. This achieves a high degree of partitioning.

Internal partitioning of the software using the processor's Memory Protection Unit (MPU) or Memory Management Unit (MMU) can make life harder for bad actors to gain control of the system. For example, SAFE**RTOS** has a task separation and isolation function that allocates each task its own memory regions and user permissions. This prevents Tasks from accessing memory outside of their allocated memory regions, and prevents bad actors from quickly gaining control of the system.

Hardware solution examples include TrustZone by ARM Zone, and Secure Shield by Synopsys. They both have a concept of secure and non-secure zones that are hardware separated, with non-secure software blocked from accessing secure resources directly. Software is either located within the secure world or the non-secure world. Switching between these two zones is device specific and achieved either by software or firmware.

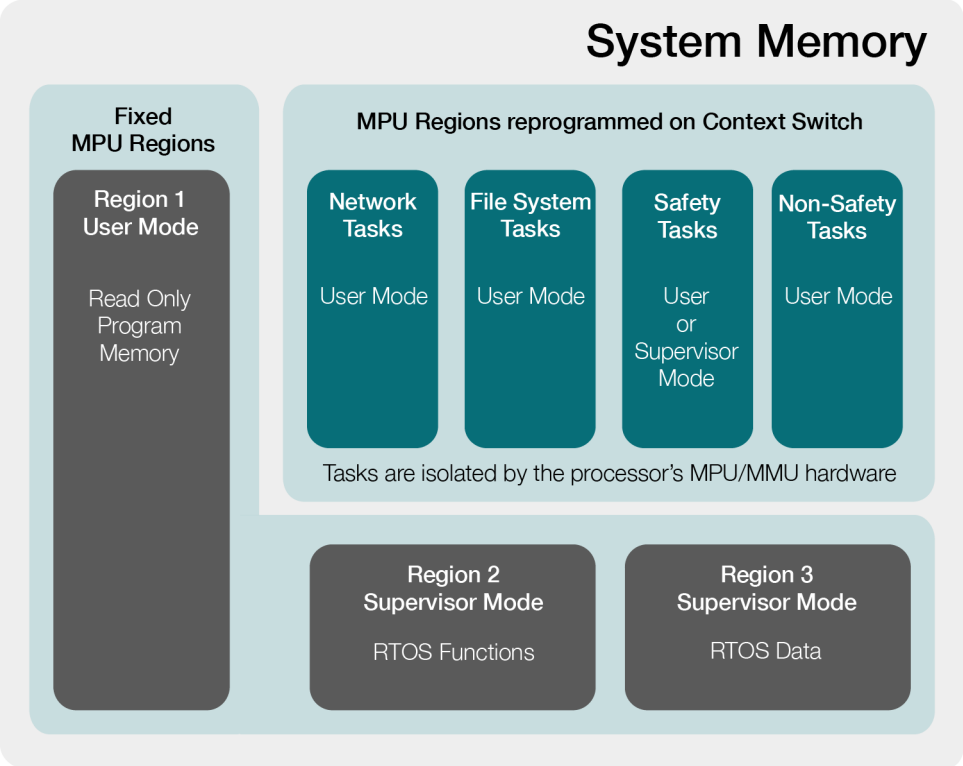

Figure 4-4 *Memory Protection Unit*

CHAPTER 5 Conclusion

This paper has looked at the first steps to take when developing security software for medical applications, through use of standards, the development life cycle, and common security mechanisms.

We've seen how security relies on a series of nested objectives that build upon each other. For this to be effective, security has to be considered from the outset of the design process, with constant review and monitoring across the lifetime of the product. Connected medical devices have many benefits, but the risks must be assessed and dealt with.

No one action will make a medical device secure; it should be assumed that medical devices operate in a hostile environment, with medical device developers expecting constant attempts at intrusion from bad actors. Only a holistic and thorough approach to security can reduce vulnerabilities in medical devices.

Contact Information

User feedback is essential to the continued maintenance and development of SAFERTOS. Please provide all software and documentation comments and suggestions to the most convenient contact point listed below.

Contact WITTENSTEIN high integrity systems

Address: WITTENSTEIN high integrity systems
Brown's Court, Long Ashton Business Park
Yanley Lane, Long Ashton
Bristol, BS41 9LB
England

Phone: +44 (0)1275 395 600
Fax: +44 (0)1275 393 630
Email: support@HighIntegritySystems.com

Website www.HighIntegritySystems.com