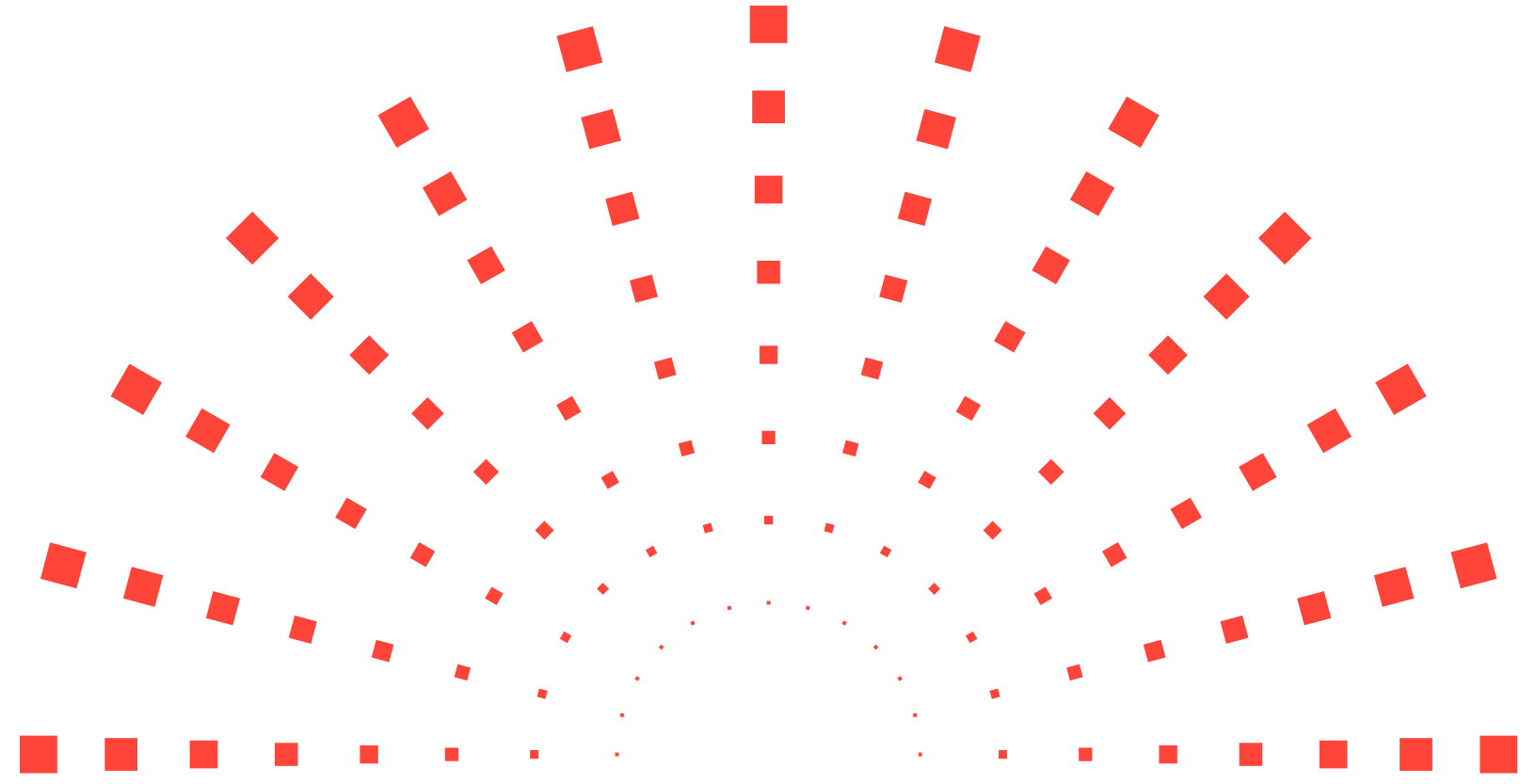
QNX

White Paper

Simplify Safe, Secure Medical Device Design

Accelerate Innovation While Ensuring Functional Safety



Abstract

With accelerating technology innovation, medical devices are becoming more complex and more connected. Consequently, as in most mission-critical systems, safety and security are increasingly vital requirements in medical device design. In recent years, the U.S. Food & Drug Administration (FDA) published an action plan designed to spur innovation towards greater medical device safety and cybersecurity. In this paper you learn about design challenges around medical device safety and security requirements as well as specific measures to deal with these challenges.

Introduction

Safety and security are two simple words that create complex requirements for mission-critical embedded systems, such as medical devices. Implementation takes great effort, in part because most device manufacturers are system integrators and do not build everything themselves. Adequate safety and cybersecurity therefore must be gauged for every component in the software bill of materials (SBOM).

Considerations for Safe and Secure Medical Device Design

Complexity and Connectivity

As technology innovation accelerates, medical equipment and devices become more sophisticated and connected.

FDA Focus

Recent actions and guidelines from the FDA focus on the topics of safety and cybersecurity.

Cybersecurity

Cybersecurity testing of all components is the responsibility of the medical product manufacturer.

Safety & Security

There is a growing need to improve the approach designers take to medical device safety and security.

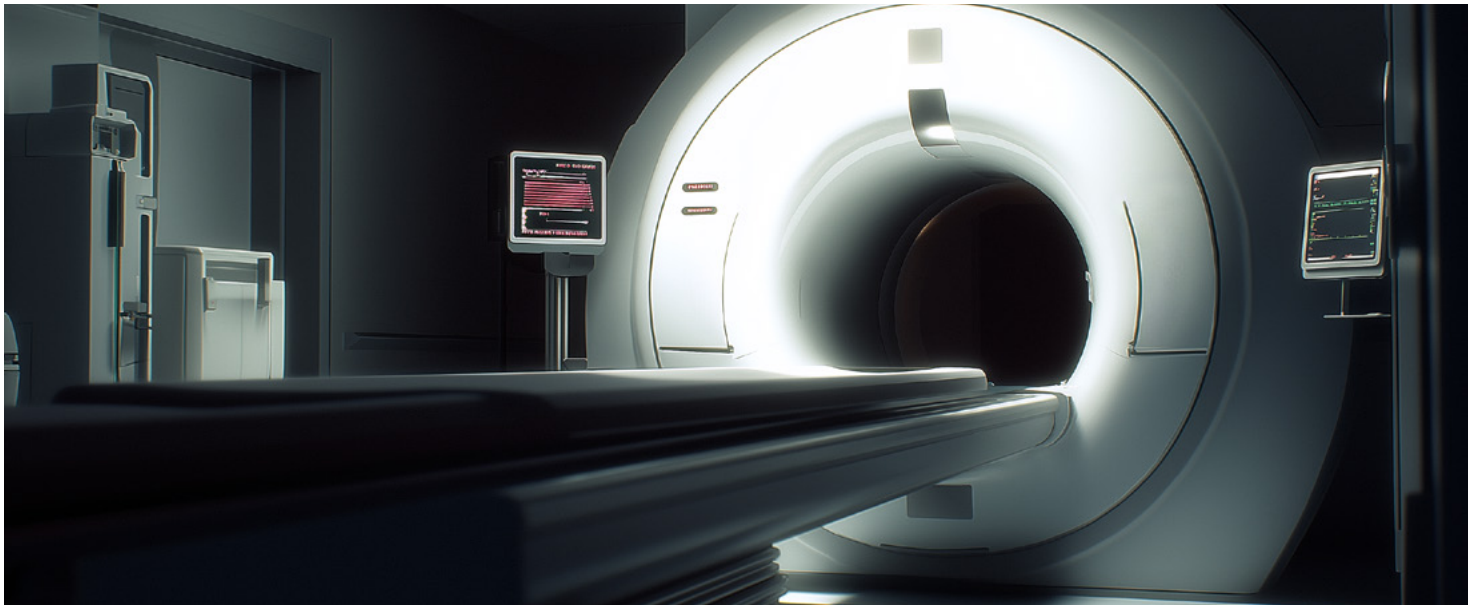
Scrutiny of Software

FDA places greater emphasis on the software in medical devices.

Trends for The Medical Market

The FDA regulates more than 6,500 medical device product categories. Recent changes made by the FDA have brought software to the forefront of medical device regulation. As connectivity becomes a standard feature for medical devices, cybersecurity requirements also surface. The FDA's cybersecurity guidance for medical device manufacturers reflects the market's awareness of lurking risks.

Device makers are required by the FDA to have a clear inventory of the software used in the device through a software bill of materials (SBOM), which comprises all software developed by the device maker and all software obtained off the shelf (OTS). The FDA considers cybersecurity and cybersecurity testing to be the responsibility of the medical product manufacturer. The manufacturer also bears responsibility for the safe and effective performance of the medical device. This puts a huge burden on medical device manufacturers to test the security of third-party software, use software designed for safety and security, build in mission-critical security features, and manage lifecycle security over the life of the device in the field.



Safety

Most medical device makers have a well-defined approach to functional safety. This approach is reflected in the device maker's corporate culture and internal processes. Although the FDA provides guidelines for device makers to follow, such as Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices and General Principles of Software Validation, it places the onus for device safety on the device manufacturer.

So, where should a medical device manufacturer start to ensure functional safety? **We recommend you address safety in each of the following four phases:**

Four Phases of Ensuring Functional Safety In Medical Devices

1. Find Hazards
2. Define Requirements (including RTOS requirements)
3. Select OTS Components
4. Submit for FDA Approval

1. Find Safety Hazards

Start By Identifying The Safety Hazards In The System

Many of the international standards on functional safety start with hazard identification for a safety-critical system. In fact, ISO 14971, **Medical Devices – Application of Risk Management to Medical Devices**, is an FDA-recognized standard for hazard and risk management.

As we look at the system holistically, we can try to identify all the system-level hazards.

Examples of System-Level Hazards

- The user interface of the device does not accept input from the user.
- The sensor on the device malfunctions and reports wrong data.
- The device does not perform the prescribed action A within time T after receiving the command.

All of these malfunctions cause a safety issue under some use scenarios. Let's consider the user interface of the device does not accept input from the user. If the device is designed to administer a medication to a patient and has an interface for the user to stop the medicine, then it is safety critical that this function does not break down. In contrast, a general malfunction does not impact anyone's safety.

If a malfunction can impact the safety of the user or a patient, it is a safety hazard.

When all software components are developed in-house by the device maker, then the hazard identification process is straightforward. However, many medical devices contain off-the-shelf hardware and software products. Hazards in these components can lead to a system-level hazard.

For example, a contract manufacturer could introduce a source of system-level hazard by cloning the device, which may be enabled by the insecure transport and control of encryption keys in production. The same hazard identification exercise can be applied to every component that the device maker procures for the system, including middleware and development tools.

2. Define Safety Requirements

After the hazards are identified, the next step is to find ways to mitigate the risks that could result from those hazards. This process usually starts with the definition of functional safety requirements.

Some of the hazards can be mitigated at the system level. The risk a logical error could occur in action A could be handled by the functional safety requirement the algorithm used to accomplish action A must be validated to ensure there is no logical error. If the device maker designs the software for action A, they could probably use a design verification method to perform the required validation for the algorithm. In this case, the requirement does not cascade down to other parts of the system.

In many cases, safety requirements cascade beyond the system level, such as action A must be free from interference from another action in the system. This hazard leads to safety requirements in the hardware, board support package (BSP), operating system, middleware modules and top-level application, as well as the communication and collaboration among all these pieces.

Let's see how a system-level risk translates into safety requirements for various components in the system.

Example: Device Does Not Perform the Prescribed Action A Within Time T After Receiving the Command

1. **Risk:** The hardware malfunctions after receiving the command.

   - **Safety requirement:** The hardware's power unit must have failure probability lower than <threshold>.

2. **Risk:** The operating system does not respond in time within time T after receiving the command.

   - **Safety requirement:** The operating system must have an upper bound for the response time less than T.

3. **Risk:** A logical error could occur in action A.

   - **Safety requirement:** The design of action A must be free from logical errors.

4. **Risk:** Action B could interfere with the proper execution of action A.

   - **Safety requirement:** Action A must be free from interference from another action in the system.

RTOS Requirements For Safety

Let's examine a specific case: safety requirements that cascade down from the system-level freedom from interference requirement used above.

The operating system needs mechanisms through which a device designer can achieve freedom from interference. Table 1 illustrates a few of the requirements taken from the hazard and risk analysis performed on the QNX® OS for Safety. If you use a different operating system, you may have requirements that look different from these but achieve the same end goal.

Freedom from interference is one of the most crucial yet challenging requirements for a safety-critical system. Well-known technical features, such as microkernel architecture and temporal partitioning, as well as newer favorites such as input output memory management unit (IOMMU), a component that brings the MMU-like mechanism to IO devices with direct memory access. Such IO devices include graphic processing units, USB connections and audio devices.

Run-Time Isolation

The code comprising the QNX OS for Safety shall be isolated at run time from faults, errors and failures in other components of the operating system.

Memory Protection

The QNX OS for Safety shall prevent code running as an application from writing to memory used exclusively by the QNX OS for Safety.

IOMMU

The application guidance provided for engineers deploying the QNX OS for Safety shall require the use of IOMMU-equivalent constraints on any hardware capable of direct memory access.

System Call Robustness

The QNX OS for Safety shall protect itself against improper system calls. Specifically, it must be impossible for any application code to cause a failure in the QNX OS for Safety by accidentally or maliciously passing a value in the parameter of a system call.

Table 1: How an Operating System Can Address the Safety Requirement Defined for the Real-Time Operating System (RTOS): Action A Must Be Free From Interference From Another Action in the System.

3. Select and Test OTS Components With Safety in Mind

✓ OTS is a Necessity

With the increasing complexity of medical devices, the use of OTS (off-the-shelf) components is a necessity.

✓ Safety Pedigree

Understand the safety pedigree of an OTS component, before choosing it.

✓ Analyze Binaries

Scan the complete software product for security vulnerabilities and craftsmanship

How to Select OTS Components

- The supplier's past track record in similar types of systems.
- Standards compliance—the FDA has multiple recognized consensus standards for safety and security listed in their guidelines.
- Product information.
- Supplier audit.
- Software composition analysis.

Given that top-level safety requirements can cascade down, you may wonder, "How do device makers find assurance that these safety requirements are met in an OTS component?" The answer is found in the diligence applied to the selection process and software composition analysis.

Checklist To Select OTS Components For Safety-Critical Systems

✓ Company Track Record

In mission-critical fields, the deployment of an OTS component in similar types of systems is often used as selection criteria. Strong adoption for the component in the past indicates a level of reliability.

✓ Information Readily Available About the Component

This information includes marketing material published on the supplier's website or through its sales channels as well as publicly accessible technical data and specifications. Only limited product information may be accessible prior to procuring the component. However, you will get more information when you buy the product. So the questions to ask are, "Does such material exist?" and "Has the supplier considered functional safety when building the product?" A close look at public product information can yield important clues.

✓ Proof of Standards Compliance

The FDA recognizes and references several safety and security standards in guidelines. The FDA even issues a dedicated guideline on the topic of governance for OTS components, **Guidance for Industry, FDA Reviewers and Compliance on Off-the-Shelf Software Use in Medical Devices.** IEC 62304 is one of the leading consensus standards for the software safety lifecycle. If a supplier shows compliance to a functional safety standard, it is a good indication that the OTS component is built with safety in mind.

✓ Supplier Audit

Probably the most thorough form of diligence is an audit of the supplier. Although not all suppliers support this type of engagement, it is a common practice in mission-critical industries. This type of activity can cover much ground, from the product lifecycle to actual product artifacts. An audit can provide an in-depth look at how the OTS component is developed and gives you the deepest level of insight.

✓ Software Composition Analysis

You can also inspect the binaries **through a software composition analysis solution such as BlackBerry Jarvis 2.0** to uncover software composition as well as vulnerabilities. By scanning, you might uncover details about the software that your supplier doesn't even know—and without having to access source code or supplier-provided material.

The Road to Simplicity

Select the right software platform—a deterministic, POSIX based, microkernel OS.

Meet reliability and safety compliance requirements.

Secure embedded software over its deployed lifecycle.

Bring products to market quicker, on budget, with quality.

QNX Safety Expertise

QNX® has developed in-depth safety expertise and technology through more than 40 years of industry experience. Our functional safety capabilities are validated and demonstrated through customer adoption of the **safety pre-certified QNX® OS 8.0** in hundreds of millions of devices, including medical products. The QNX OS for Safety is certified to IEC 62304 Class C, as well as other functional safety standards. Expediting safety certifications for mission-critical applications is one of our core competencies, and we help our customers with their product development and commercialization plans through the use of our products, safety-certification consulting, and professional services.

4. Submit For FDA Approval to Ensure Safety Standards Compliance

One of the easiest criteria to set for the OTS supplier is standard compliance and there is no lack of guidance from the FDA. Figure 1 illustrates the various standards for quality and safety as well as their relationships (extracted from the IEC 62304 standard).

The IEC 62304 standard defines the lifecycle requirements for medical device software. The set of processes, activities, and tasks described in this standard establishes a common framework for medical device software lifecycle processes.

As stated by the FDA, this standard is relevant to medical devices and is recognized for its scientific and technical merit. It is also known for its support of existing regulatory policies. If a component claim complies to the IEC 62304 standard, then you can safely assume that its development followed a well-established safety lifecycle. Other standards feed into IEC 62304, including ISO 14971, which defines the practice of hazard and risk analysis.

Having a pre-certification requirement for OTS components saves a lot of time and effort during the due diligence stage. It can also ensure a reasonable threshold for the quality and fitness of the component you receive.

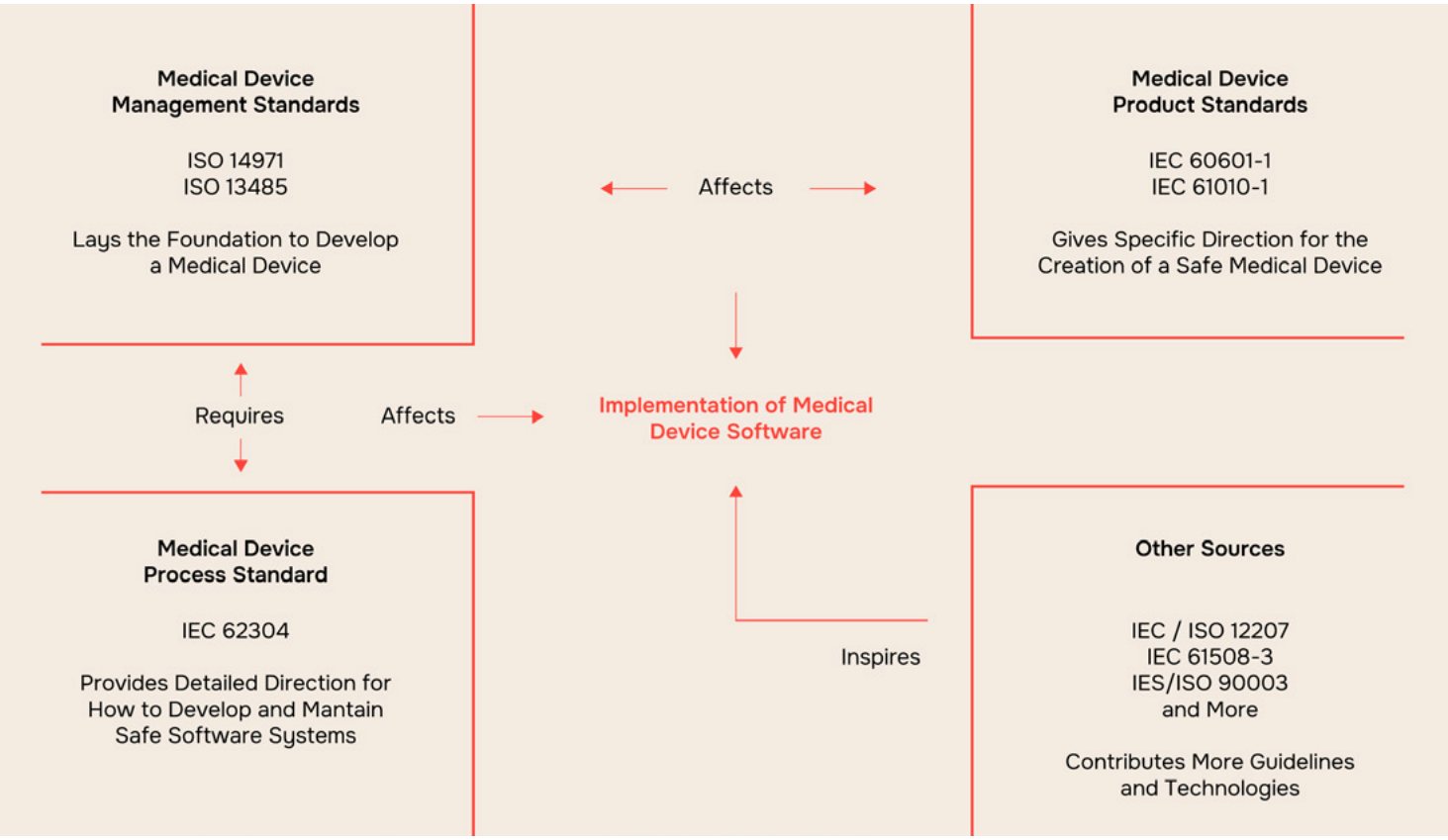


Figure 1:

Various standards for quality and safety as well as their relationships (extracted from the IEC 62304 standard).

Cybersecurity

Cybersecurity Is a Growing Concern

A more recent focal point for medical devices is cybersecurity. As software plays a bigger role in the function of medical devices, and connectivity becomes a standard feature, cybersecurity requirements are surfacing more and more. Recent US government activities reflect the market's awareness of the lurking risks.

In 2021, US President Biden issued an executive order that requires medical device makers and other software suppliers to US government entities to provide an SBOM. This reflects the FDA's plans to improve its approach towards medical device safety, including cybersecurity, and the FDA's draft guidance, Content of Premarket Submissions for Management of Cybersecurity in Medical Devices, which requires "a list that includes but is not limited to commercial, open source, and off-the-shelf software and hardware components that are or could become susceptible to vulnerabilities."

BlackBerry Jarvis 2.0 can help you comply with these regulations. This BlackBerry cybersecurity solution automates the enumeration of software bills of materials and can scan a software product for vulnerabilities and software craftsmanship from compiled binaries. BlackBerry Jarvis 2.0 has been cited as the most promising and robust binary analysis solution in an assessment conducted by The Aerospace Corporation on behalf of the United States Department of Defense (DoD).

Both the executive order and the FDA's draft guidance also call for device makers to include automated software update delivery systems to deliver critical security patches. The idea is to help device owners "better manage their networked assets and be aware of which devices in their inventory (or use) may be subject to vulnerabilities."

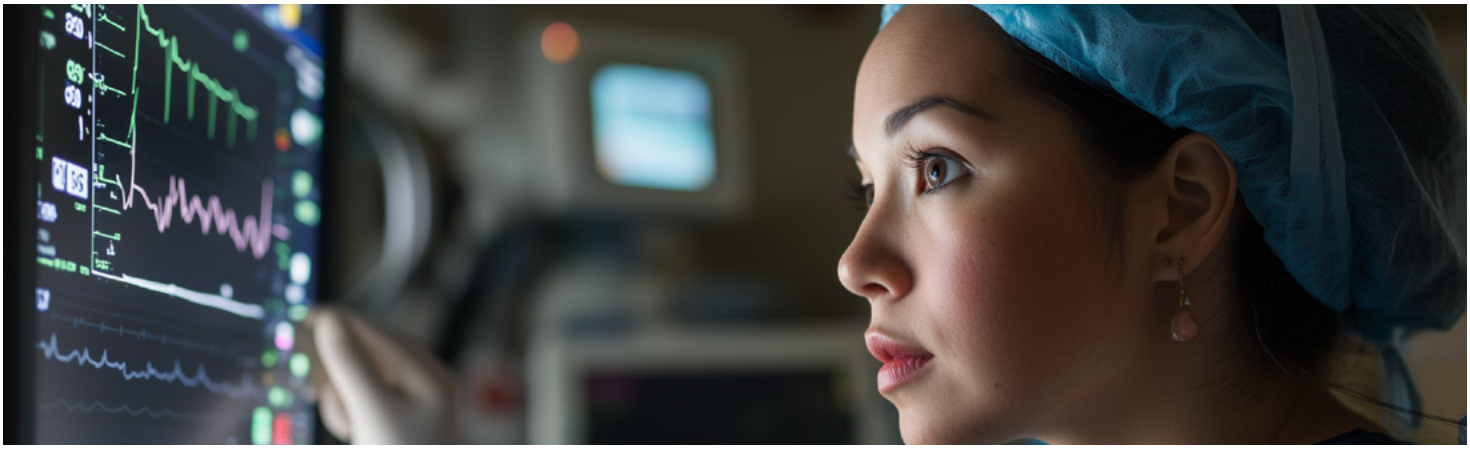
The FDA, which is the only US government agency responsible for the cybersecurity of medical devices, now considers medical device manufacturers to be responsible for the validation of all software design changes. This includes any computer software changes that are required to address cybersecurity vulnerabilities. The FDA states that it will not conduct pre-market security testing for medical products, as it considers cybersecurity testing to be the responsibility of the medical product manufacturer. The action plan also clarified that the medical device manufacturer chooses the software to use and thus bears the responsibility for the security as well as the safe and effective performance of the medical device.

Despite these cybersecurity guidelines, malicious attacks will likely grow as more medical equipment becomes connected and, therefore, vulnerable. The actions by the US aim to mitigate serious threats to connected medical devices, especially attacks that could disrupt the operation of critical monitors and drug delivery equipment.

Cybersecurity Is a Challenging Topic

Since cybersecurity is a newer focus than safety for medical devices, it has not accumulated the same level of knowledge, expertise and solution sets as safety. Cybersecurity for embedded devices calls for a different set of skills and solutions than IT security, which is better known. There is no consensus on a single dominating cybersecurity standard for embedded systems, for example.

However, if you search on the topic of software security, there is no shortage of answers. These answers range from models for a secure product lifecycle to myriad tools for scanning for product security issues and from databases with hundreds of thousands of security vulnerabilities to many types of security services. Unfortunately, if you are relatively new to cybersecurity, this flood of information is overwhelming and difficult to guide a real action plan.



Security Strategy and Tactics

Let's explore several widely accepted strategies and tactics for security, including cybersecurity. Some of these are longterm initiatives while others address immediate gaps. Your company may need a mix to reach sufficient levels of security.

First, if security has not been a focal point before, a good understanding of your security posture is needed. This type of assessment is best performed by security experts, ideally someone with domain knowledge for your industry. This activity takes stock of existing security mechanisms in place and identifies the major gaps in your internal process and product.

Despite the differences between safety and security, one similarity across the two disciplines is that they both require focus on internal process and specific product definition. Of all the strategies and tactics listed, the one that is most important but least clear is build a culture of security as the foundation. Let's examine how to build a safety culture.

The Role of Internal Process

Internal processes play an important role in building a culture of security. Although there are university courses on subjects such as encryption, few graduates are prepared to build security into their first product. Also, one person's approach to security may differ greatly from another's. A well-defined internal process for security is the necessary foundation for a security culture. Although some standards try to prescribe what this process should look like, such as ISO 21434, Road vehicles—Cybersecurity engineering, an effective process must suit the idiosyncrasies of the company and be sincerely endorsed and embraced by its executives and all its people.

The internal process could range from defining a higher-level framework like the complete security lifecycle to finer-granularity rules around vulnerability assessment as well as the tools and methods used to scan products for cybersecurity issues. It takes time for an internal process to become a natural part of work life and that is what it takes to build a culture.

When a good security culture is in place, other strategies and tactics follow naturally. For example, for identify vulnerabilities in deployed products, a good internal process would call out vulnerability assessment as a mandatory verification and validation (V&V) method. As another example, build protection mechanisms is a basic requirement for any security lifecycle. The two last items on the list are just reminders to get external help—using security consultants or reviewing a security standard—and to continue to manage security in deployed devices.

Cybersecurity Strategy and Tactics

Review your security posture.

Identify vulnerabilities in deployed products.

Build protection mechanisms in current-generation products.

Build a culture of security as the foundation.

Leverage external expertise where possible.

Adopt and adapt recognized process models in the market.

Architect your next-generation product with best-in-class security technologies and practices as well as defense-in-depth topologies.

Manage the lifecycle of security for the life of the product in the field.

Security Specification

Having covered internal process, let's look at security specification for the product. Recall that for functional safety, we started with system-level hazards and then examined each component to determine which component-level hazards could lead to a system-level hazard. That approach doesn't apply to security.

To identify the security risks that exist for a medical device, suppose a malicious hacker gains control of an infusion pump. The hacker may interfere with the normal operation of the device, such as administering the wrong dose of the medication, or not administering any medication when the user prompts it. The various places in the system where the attacker could get in to take malicious actions are called attack surfaces. These attack surfaces are like safety hazards. When analyzing a system for security, the goal is to identify the vulnerable places. The operating system is an attack surface because the operating system is the central brain. If a hacker can gain access to the operating system, such as with root privileges, then they basically have carte blanche to do whatever they please. That leads to operating system security specifications.

What do we do next once we have identified the attack surfaces? Just like in safety, security must be applied at the level of each component.

To protect attack surfaces, you need to have a good understanding of the components and figure out what makes them vulnerable. Let's consider the operating system again. Fortunately, the QNX operating system has architecture that is naturally more resilient than a monolithic operating system. In a monolithic operating system, everything resides in one large space. Critical components, such as the kernel and the drivers, share the same memory space. This provides an opening for hackers to get into the brains of the system from a vulnerable component. After gaining control of something critical, such as the instruction pointer, hackers can run any code they wish to exploit the system.

By contrast, in a microkernel system the kernel uses a different memory space, one that is not shared with other components, such as drivers. In fact, every component runs in its own dedicated memory space and cannot use the memory space of another component. This makes the hacker's goal much more difficult to achieve.

In a related manner, a microkernel architecture is also useful for meeting functional safety requirements as it provides freedom from interference, a mechanism required to build systems of mixed-criticality.

Security Solutions

Apart from inherent architecture, there are solutions that can be applied to software components to make them more secure. Process Manager Abilities in the QNX OS 8.0 illustrate this.

Abilities control a process's ability to perform certain operations. In the QNX microkernel, a process is the smallest container unit, which holds one or more threads that run and do work.

You may have heard of the fork command. It creates a new process (child process), which is an exact copy of the calling process (parent process) with a few differences, such as process ID, number of threads, and timer values. An ability governs the usage of fork. If you use a central security policy that only gives fork privileges to certain trusted processes, it prevents malicious actions such as a fork attack—a denial of service (DoS) attack that calls recursively until all system resources execute a command and the system becomes overloaded and unable to respond to any input.

Another example is the ability to map physical memory, which allows a process to invoke **mmap** to map a memory region into the address space of a process. The QNX OS 8.0 comes with a memory management unit (MMU), which hides the physical memory from most of the processes in the operating system and presents them with virtual memories instead. This alone is a very secure approach. However, sometimes a process needs to use physical memory, such as in the case of a process belonging to a hardware device driver. Through security policy, you can restrict which processes can map physical addresses and, in turn, mitigate security risks.

Protecting The OS

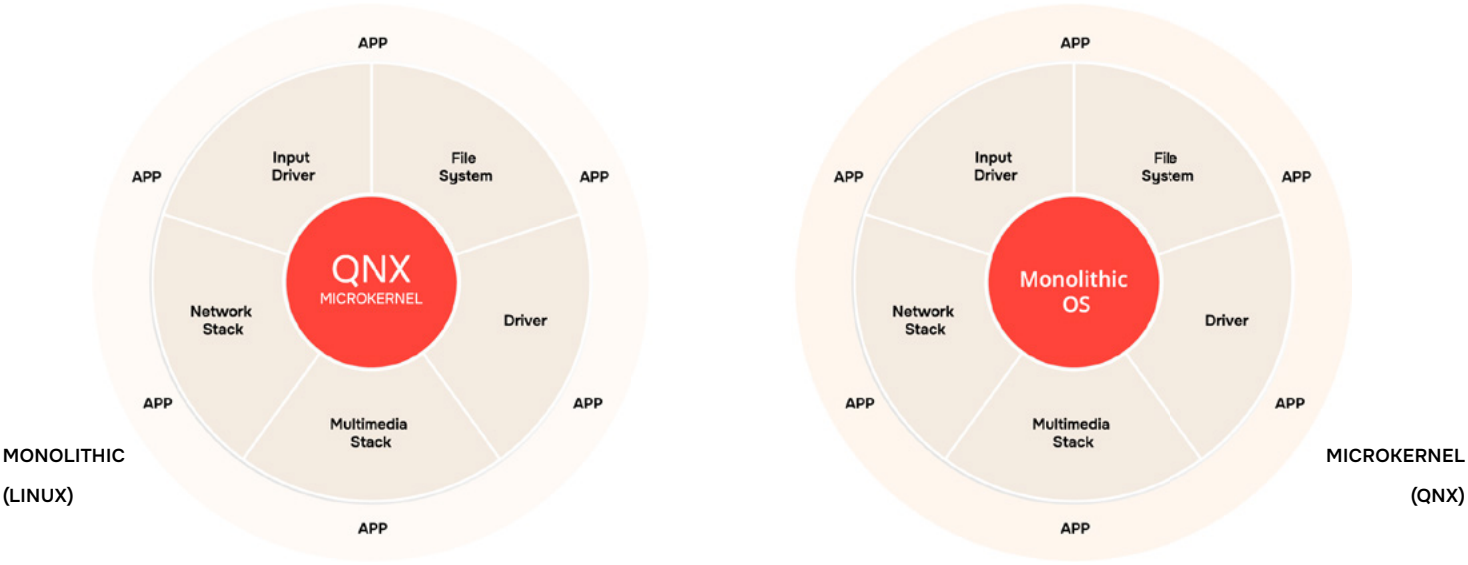


Figure 2: Protecting The OS

Examples: Process Manager Abilities

- Root and On-Root Abilities.
- More than 65 abilities to control various aspects of privileged operations.
- Abilities can be granted denied, locked or inherited.
- Fine-grain control of process privileges.

QNX Cybersecurity Expertise

QNX has developed in-depth cybersecurity expertise and technology through 45 years of industry experience. Our cybersecurity capabilities are validated and demonstrated through the adoption of QNX solutions in segments with top security demands, including all seven of the G7 governments. QNX solutions secure more than 500 million endpoints. We also hold a wide range of top security certifications, including NSA Classified Program approval, FedRAMP, and NATO Restricted. All this reflects the deeply entrenched security culture at QNX. Beyond secure solutions and many types of security services, QNX also provides expertise in medical device cybersecurity.

August 2025

About QNX

QNX, a division of BlackBerry Limited, enhances the human experience and amplifies technology-driven industries, providing a trusted foundation for software-defined businesses to thrive. The business leads the way in delivering safe and secure operating systems, hypervisors, middleware, solutions, and development tools, along with support and services delivered by trusted embedded software experts. QNX® technology has been deployed in the world's most critical embedded systems, including more than 255 million vehicles on the road today. QNX® software is trusted across industries including automotive, medical devices, industrial controls, robotics, commercial vehicles, rail, and aerospace and defense. Founded in 1980, QNX is headquartered in Ottawa, Canada.

Learn more at qnx.com →

©2025 BlackBerry Limited. Trademarks, including but not limited to BLACKBERRY and EMBLEM Design, QNX and the QNX logo design are the trademarks or registered trademarks of BlackBerry Limited, and the exclusive rights to such trademarks are expressly reserved. All other trademarks are the property of their respective owners. BlackBerry is not responsible for any third-party products or services.