BlackBerry | QNX

The Past, Present and Future of Cybersecurity for Embedded Systems

How evolving threats are changing the game for embedded device manufacturers

The only system that's safe from cyberthreats is one that's never powered on. While this is an overstatement meant to highlight the seriousness of security threats that surround things like our computers and phones, it's something to be considered as threats to embedded systems evolve. Today more than ever, ubiquitous connectivity, consolidation of functions and automation each play a major role in enabling new capabilities of software-driven devices, while introducing new threats. And while you can't stop progress, there are things you can do to limit the vulnerabilities that come with new and evolving systems.

A recent article in SecureWorld[1] highlighted the challenge of ubiquitous connectivity, noting that the Information Security Forum (ISF) has forecasted that through 2022 "new technologies will further invade every element of daily life with sensors, cameras and other devices embedded in homes, offices, factories and public spaces." The ISF also predicted that "a constant stream of data will flow between the digital and physical worlds, with attacks on the digital world directly impacting the physical and creating dire consequences for privacy, well-being and personal safety."

Gone are the days when embedded systems were generally isolated and hardware-driven, and exposing vulnerabilities was limited to the realm of academic experimentation or internal malfunction. Many of today's embedded systems are vulnerable to attacks, particularly when manufacturers have not designed their system to be resilient to attacks or lack a product strategy for cybersecurity management. And this is what keeps manufacturing leaders up at night.

Cyberthreats of the Past

01

In 1971, Bob Thomas created the first ever computer worm and made history in the process. That computer worm was not malicious. It simply bounced around between computers connected via a local switch and displayed "I'M THE CREEPER: CATCH ME IF YOU CAN" on their screens.[2]

The first real cyberattack occurred in November 1988 when Robert Morris created a computer worm that slowed down the Internet – for the time, significantly. Morris's motivation wasn't to cause damage, but simply to highlight security flaws in Unix Sendmail and password management. However, the worm did unintentionally cause failures in parts of the Internet, which lasted days and resulted in damages estimated at between $100,000 and $10,000,000.[3]

But it wasn't until 1999, when Microsoft Windows 98 was introduced, that security vendors started releasing anti-hacking software, marking the beginning of the cybersecurity industry. Ever since, governments, businesses, universities and, unfortunately, hackers have been looking more closely at the security of devices that are susceptible to being hacked. During this time, all the players, both attackers and defenders, focused primarily on computers and servers.

The cybersecurity of embedded systems didn't become an issue until more recently. For background, an embedded system is a controller that sits within a larger system, usually designed to perform a dedicated function. At one time, embedded systems were composed of a piece of hardware, a firmware component, and in some cases, an embedded operating system layered on top of one another as depicted in Figure 1 below.

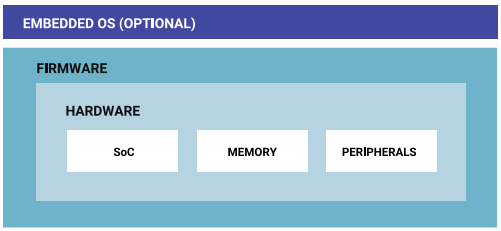


Figure 1:
Older embedded system architecture

At that time, many embedded systems were not designed to protect themselves from attacks. Priority was not placed on cybersecurity of embedded systems based on these assumptions:

- Embedded systems are not attractive targets to hackers.
- They are not vulnerable to attacks due to their isolation because they require complex authentication.
- In some cases, encryption provided adequate protection.

The Evolving Challenge of Protecting Embedded Systems

02

In today's market, embedded systems are becoming more advanced and contain many layers of software within their architecture. And in many cases, they perform mission critical functions in increasingly complex embedded systems. They are implemented across industries from aerospace and defense all the way to household appliances. There have been several well-documented attacks on embedded devices, ranging from vehicle anti-theft components, to control systems, to printers. In the case of printers, the attackers hijacked the printer, sending copies of documents to their own computer, in some cases getting their hands on confidential or secret information.[4]

Let's first look at the general architecture of embedded systems today. Figure 2 below shows two layers in addition to those shown in Figure 1. These are the middleware and the user application layers. Middleware is software that provides services to the user applications and is sometimes described as the software "glue". Some examples of middleware are databases, application servers and content management systems. User applications are programs that are designed for the end user. An example of a user application is an Internet browser or Android Auto running on a vehicle infotainment system.

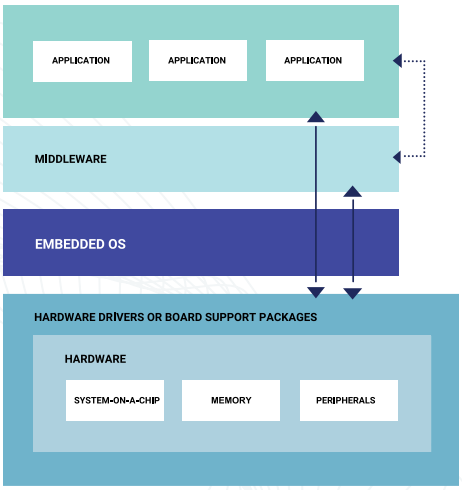


Figure 2:
Example Architecture of Modern Embedded Systems

As a result of the growing popularity and diversity of embedded systems and their connection to Internet of Things (IoT), attacks on these systems are increasing in frequency. The Internet of Things refers to system of connected computing devices, along with mechanical and digital machines. These devices can transfer data over a network without requiring human-to-human or human-to-computer interaction. IoT involves many technologies and has evolved to include a convergence of technologies ranging from analytics, to machine learning, to embedded systems. This discussion focusses on embedded systems.

How Systems Are Attacked

The embedded OS is often the target of attacks as it has the most control over the entire embedded system (hardware, middleware and user application layers). Some of the players attacking these systems can include: researchers, hackers, organized crime and nation states.

There are several ways someone can exploit a system, but the general method follows a process something like this:

1/ Get network access to the system.

2/ Understand the underlying software, hardware and embedded OS.

3/ Find a vulnerability in the host-based protection whether it's the hardware drivers, OS or middleware.

4/ Manipulate the software.

5/ Exploit the software or overall system.

Most embedded systems deploy, at a minimum, the following defense mechanisms.

- **Hook detection:** A means of detecting if something is attempting to alter or augment the behaviour of the software (embedded OS, drivers, middleware or applications)
- **Attestation:** A way to check if an unauthorized process has changed the software
- **Firmware/driver integrity verification:** A tool to detect any unauthorized changes to hardware drivers or firmware
- **Logic check sum:** A mechanism for detecting errors that might have been introduced during data transmission or storage
- **Encryption:** Encoding of data (stored or transmitted) in such a way that only authorized users can decode it
- **Authentication:** A mechanism for checking username and passwords and enforcing password policies (length, complexity, expiry, etc.)

Vulnerabilities by Type

Data for this graph was compiled using BlackBerry® Jarvis™, a cloud-based, binary static application security testing (SAST) platform to highlight the categories of vulnerabilities in embedded software within the automotive industry. It shows that insecure APIs contribute to approximately 70 percent of all categories of vulnerabilities found. Table 1 describes the functions or attributes that can lead to vulnerabilities within each category.

Note that buffer overflow is at the top of the list of exploits under the category "Insecure API." Buffer overflow warrants considerable attention as it can lead to remote code execution or system control by attackers.

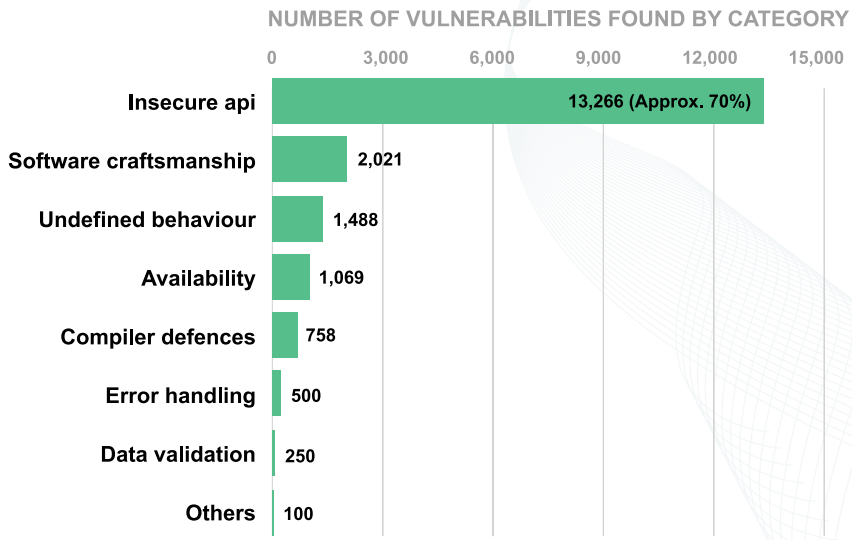


Figure 3:
Categories of vulnerabilities within embedded software, using automotive data. Analysis by BlackBerry Jarvis.

Category	Common functions or attributes leading to exploits
Insecure API	Functions that lead to buffer overflow exploits. Certain memory allocation functions that leads to vulnerabilities like memory leaks, insufficient memory allocation and use after free.
Software Craftsmanship	Functions that have unspecified, undefined and implementation-defined behavior associated with them (e.g., stdio.h). Unreachable code that indicates either a logic error or a binary that has been built without optimization. Functions with high cyclomatic complexity which are considered unstable.
Undefined Behavior	Uninitialized variables that have an indeterminate value and incorrect use of functions that can lead to undefined and inconsistent behavior.
Availability	Functions or code that can interrupt the operation of the system and its resources.
Compiler Defences	Absence of, or minimal, compiler defence mechanisms.
Error Handling	Absence of, or insufficient, error handling mechanisms.
Data Validation	Absence of, or insufficient, data validation mechanisms.
Other	Absence of, or insufficient, protocols, insecure protocols, information leakage, insufficient cryptography or lack of coding best practices.

Table 1:
Vulnerability Taxonomy Descriptions

The following graph depicts what Kaspersky Lab ICS CERT, 2018 was able to unearth in both industrial and IoT systems. It shows that vendors were able to close only approximately 10 percent of vulnerabilities (29 of 286) by the end of 2018. Approximately 50 percent of these identified vulnerabilities were found to have the potential to lead to remote code execution and give an attacker control of the device or could help attackers trigger a denial of service attack, making the equipment unusable.[5]

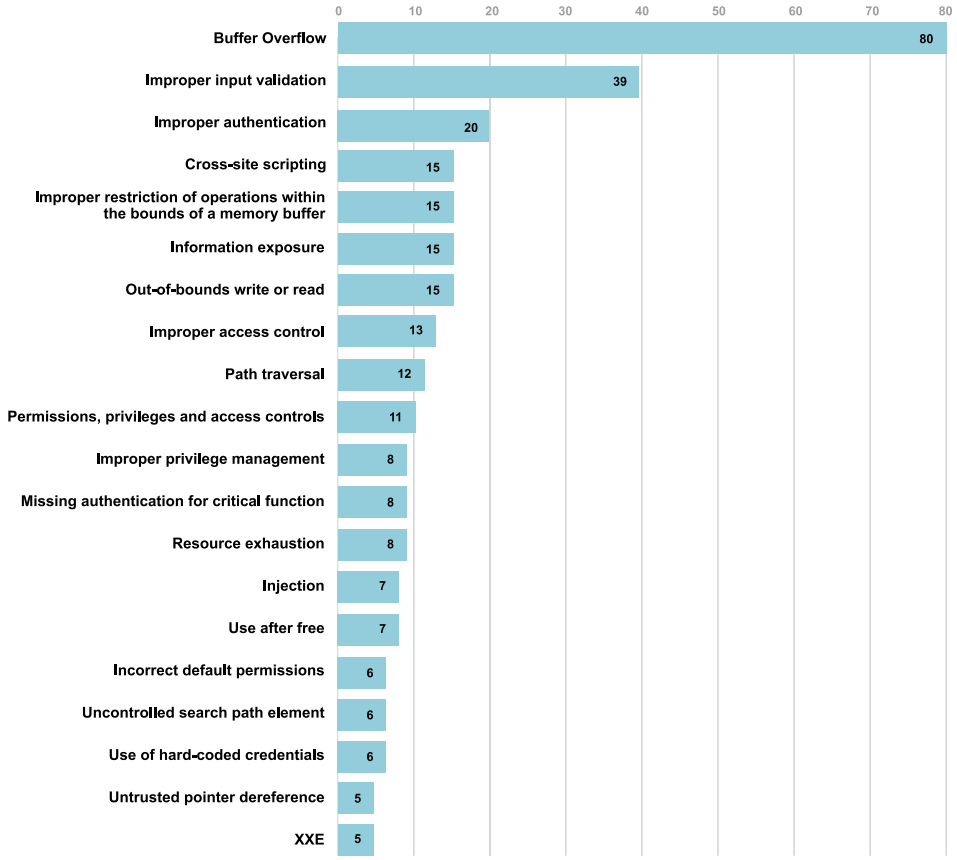


Figure 4:

The number of ICS vulnerabilities discovered by type.[5]

Most embedded software development is dominated by the C programming language. This is generally considered a very flexible but unsafe language[9]. Some of the known risks and vulnerabilities can be mitigated if you certify your OS, C libraries, math libraries and hypervisor safety system to the international standard most applicable for your system, such as the functional safety standards ISO 26262, IEC 61508, or others. Through the rigorous certification process of any of these standards, which includes detailed analysis against design, development and test, some of the known risks and vulnerabilities of the C programming language can be mitigated.

Memory Corruption Via Buffer Overflow

Memory corruption via buffer overflow is the most common vulnerability in the software on embedded systems as highlighted in the automotive and industrial vulnerability figures above. Buffer overflow attacks occur when an attacker modifies software, writing data to a memory buffer, making it overrun the buffer's limits and overwriting adjacent memory buffers. This can cause the software to execute arbitrary code and can possibly allow the attacker to take control of the system or cause it to crash. These types of attacks are extremely dangerous especially if your systems are performing critical tasks, for example, an industrial robot lifting heavy supplies while interacting with humans in a manufacturing facility, or an advanced driver assistance system helping navigate a vehicle through traffic. Being able to exploit a system via memory corruption can often be complex for attackers because it requires reverse engineering of hardware and/or software. In addition, having access to the unique software of embedded systems is challenging.

But in today's market, embedded systems are connected to the IoT and in some cases they use open source operating systems (e.g., Linux), which results in a substantial increase in attack surface. With open source OS, you need to design your own software countermeasures to protect your system, and to keep up with evolving threats. There's no trusted third party to provide constant security updates, or to help guide your security strategy. It is a scary thought.

Also, we must keep in mind that embedded devices have a significantly longer shelf life (10-20 years) compared to commercial devices like servers or computers (2-5 years) and updating embedded device software is extremely challenging. In fact, exploitation of embedded devices is considered inevitable by most system designers[7].

Mitigating Exploits

03

Today, there is a new hope for protecting embedded systems. These three exploit mitigations are now being implemented by commercial embedded OS providers, and are often called the Three Musketeers:

1/

Executable space protection (ESP):

A technique to mark specific memory regions as non-executable such that an attempt to execute machine code in those regions will cause an exception.

2/

Address space layout randomization:

A technique that involves randomly positioning the base address of an executable and the position of libraries, heap and stack in a process's address space. The random mixing means the attacker no longer knows where the required code is located. This way it makes it more challenging to exploit existing vulnerabilities.

3/

Stack canaries:

A technique where the OS can detect a stack buffer overflow before execution of malicious code can occur. This method works by placing a small integer, the value of which is randomly chosen at program start, in memory just before the stack return pointer. It makes exploiting buffer overflow extremely challenging because before the OS overwrites the new memory space, it checks the stack value and if it is not the same then the execution stops, and an exception is raised.

Unfortunately, not all embedded operating systems have all these security mechanisms in place. In fact, according to surveys by Ali Abbasi of the University of Twente, only about seven percent of the 30 most popular embedded operating systems include these mechanisms[8]. Fortunately for BlackBerry® QNX® users, all these mechanisms are all available in the QNX RTOS and QNX Hypervisor – both the safety certified and non-safety certified versions of the RTOS.

Getting Ready for New Threats

No one can predict what tomorrow will hold, let alone what will happen in the next five to 10 years. However, we can make some assumptions based on current and past trends. Figure 5 below shows the number of vulnerable products in different industries. With IoT now connecting embedded systems, malicious attacks via the Internet (web, email, user applications) are contributing to more than 30 percent of attacks and removable media attacks are contributing around 10 percent – these two represent almost half of total attack types across industries.

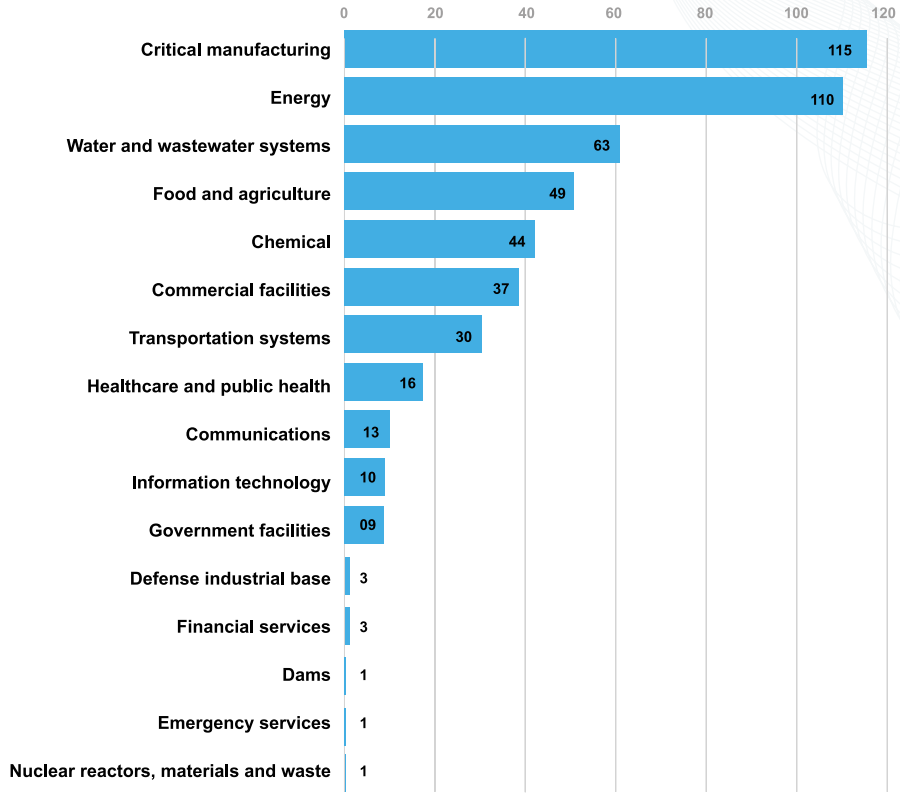


Figure 5:

Number of vulnerable products in different industries[6]

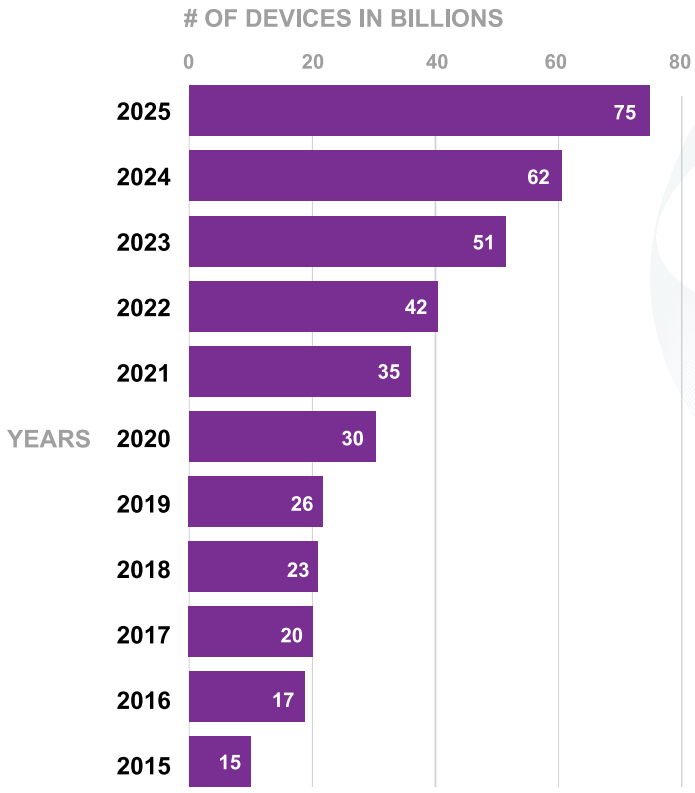
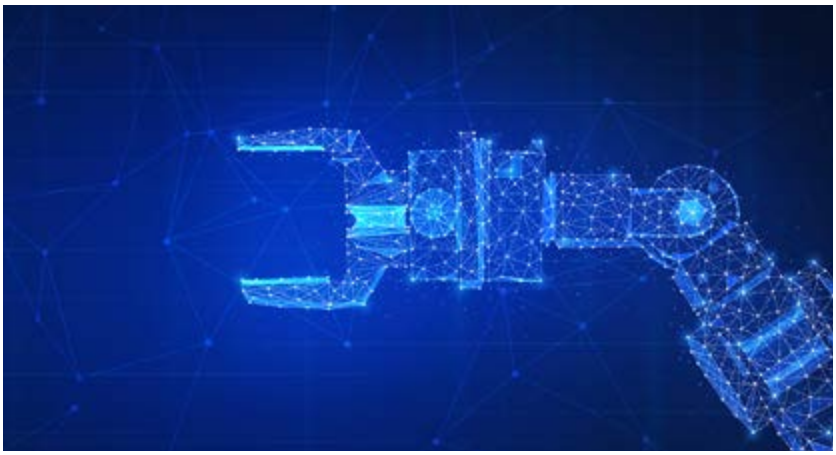


Figure 6:
Statistics for IoT connected devices (2015- 2025)[6]

Artificial Intelligence (AI) — The New Source of Threats

There is a new threat on its way, which could be considered one of the most significant we might ever see: Artificial Intelligence (AI). These are not fictional threats like Skynet, the artificial superintelligence system from the movie Terminator. Today's attacks on embedded systems make use of artificial intelligence that automate systems like those in robotics or automated driving systems. These attacks exist because there are fundamental limitations to AI algorithms that attackers can exploit to make the system fail or misbehave. These attacks are different from traditional cyberattacks because they are not the result of mistakes made by programmers or users. They are due to the shortcomings of AI. The algorithms that make AI systems work well, also contain imperfections – this is just a fact of mathematics. These limitations and imperfections create opportunities for cyberattacks.[7]



OS Security Design is Your Best Line of Defence

The security designs that embedded devices manufacturers need to architect and implement must ensure that all the layers (hardware, drivers, embedded OS, middleware and user applications) are protected from tampering, data theft, communication and malicious attacks. Unfortunately, there is no one-size fits all security design or solution that is recommended by the manufacturers of those products, or even security vendors but there are a few recommendations that we should follow as seen in the list below[4]:

- Secure boot
- Authentication
- Intrusion detection/ protection
- Embedded security management
- Secure updates
- Secure communication
- Security monitoring
- Data security
- Embedded firewalls
- Device tampering detection

Some of the security methods and provisions identified above are dependent on one another. For example, in order to have an embedded firewall or intrusion detection/protection, you need to determine how you will perform secure updates first, as both those items require frequent updates (e.g., virus definitions or signature files or security patches) in order to keep up with new attacks. You should also consider that secure updates in some cases, like drivers or the embedded OS, might require a system reboot. For safety-critical or other critical devices this is not ideal and should be done in a controlled environment. This isn't like a regular personal computer update in which you can save your data before you reboot your machine. Loss of data cannot be allowed to happen under any circumstances for critical embedded systems.

Cybersecurity Building Blocks

The key to safeguarding against attacks today and into the future is to consider the risks and costs of successful attacks as you develop your systems. These costs can range from economic, environmental, social, informational, personal privacy breaches and manufacturer brand credibility. The benefit of implementing a strong security solution far outweighs the cost of doing nothing. Attacks can compromise your systems and can keep your engineering operations, legal and other teams bust dealing with the repercussions. The risk and likelihood of these attacks and the associated attack vectors are real and increasing by the day.
As you start thinking about the elements that will go into your embedded system cybersecurity strategy, consider these building blocks:

- **Encryption:** The process of encoding information in such a way that only authorized users can access it.
- **Network communication:** A means of securing the communication between all the layers in embedded systems.
- **Lifecycle management:** Managing the entire cybersecurity lifecycle of a product from inception all the way to end of life.
- **Identity management:** A framework of policies and access lists for ensuring that the proper software or users utilizing the embedded system have the appropriate access to the required resources (hardware or software).
- **Threat defense:** The prevention or management of cybersecurity attacks.
- **Software or driver updates:** The practice of securely updating the software or driver, while ensuring minimal to no impact on the end user and overall embedded system functionality.

For another perspective on cybersecurity recommendations and best practices, you can dowload BlackBerry's 7-Pillar Recommendation for Automotive Cybersecurity white paper.

BlackBerry | QNX

Ensuring Security Now and Into the Future with BlackBerry QNX

BlackBerry QNX is leading the charge in placing cybersecurity at the center of development culture, while making use of cybersecurity tools, frameworks and architecting innovative security solutions in our embedded software products to build a comprehensive security ecosystem that suits any embedded system.

QNX® Neutrino® RTOS and QNX Hypervisor are based on a microkernel architecture and are safety certified – including the system C, C++ libraries and math libraries – to various standards (including ISO 26262 and IEC 61508). The QNX microkernel OS-based security can be more easily verified and assured compared to a monolithic OS and delivers the Three Musketeers of exploit mechanisms. BlackBerry QNX continues to work diligently in defining innovative security architectures to protect embedded systems against the future cybersecurity threats and risks.

References

[1] "Cyber Threat Forecast Through 2022," https://www.secureworldexpo.com/industry-news/isf-threat-horizon-report-2022

[2] "First Computer Virus". https://history-computer.com/Internet/Maturing/Thomas.html

[3] "What the world's first cyber attack has taught us about cybersecurity," https://www.weforum.org/agenda/2018/11/30-years-ago-the-world-s-first-cyberattack-set-the-stage-for-modern-cybersecurity-challenges

[4] "Security Requirements for embedded devices – what is really needed?" https://www.iconlabs.com/prod/security-requirements-embedded-devices-—-what-really-needed

[5] "20% of industrial control systems affected by critical vulnerabilities," https://www.bleepingcomputer.com/news/security/20-percent-of-industrial-control-systems-affected-by-critical-vulnerabilities/

[6] "IoT Connected devices installed base worldwide from 2015 to 2025". https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[7] "Attacking Artificial Intelligence: AI's Security Vulnerability and What Policymakers Can Do About It," https://www.belfercenter.org/publication/AttackingAI

[8] Technical talk by Ali Abbasi of the University of Twente on the security of RTOS found in PLC's and other devices, https://www.youtube.com/watch?v=rlic2A7VB80

[9] "Seventeen steps to safer C code," https://www.embedded.com/seventeen-steps-to-safer-c-code/

[10] "Heading off the inevitable hack attack," https://www.sme.org/technologies/articles/2017/october/heading-off-the-inevitable-hack-attack/