Safety and security for the internet of things

Jacques Brygier, Mehmet Oezer

Jacques Brygier - Directeur Général
Mehmet Özer – Solutions Architect

Safety and Security for the Internet of things

1 IoT Safety&Security challenges

Internet of Things (IoT) is a buzzword for the term “connected devices”, which we used in the last 15 years to describe the trend of connecting embedded devices. It is driven by the need to shift the usage of embedded devices to the next level of efficiency. By enabling seamless communication between all embedded devices, we capture more data about the ongoing process and are able to influence the process for optimization. The realization of seamless connectivity is favored by the broad availability of Ethernet and wireless communication. Even real-time requirements are covered by specific industrial Ethernet protocols, which are fast enough to drive high frequency control systems with a small amount of jitter.

With this seamless communication, functional safety and security of an embedded/IoT device become a challenge. A communication channel is always subject for vulnerability and can compromise the safety of the system because of a modified configuration. A security attack to a water heating system, which provides boiling water instead of tempered water, or a control valve for a chemical process, which is ‘out of control’ will definitely evolve to a safety problem.

The functional safety of a system ensures, that a system does not harm its environment. Safety is ensured by implementing a safety concept in software and/or hardware. On the software side, the underlying software platform (Operating System and Middleware) should support safety in its design and architecture, so that functional safety requirement can be more easily implemented.

Security concepts shall ensure, that the system is not harmed/attacked by its environment. As security problems can evolve to serious safety problems, security can also be seen as an indispensable safety concept. One aspect of security is to keep data confidential and ensure its integrity. If data confidentiality and integrity are not secured by appropriate strategies, a medical device may harm the patient with the wrong dosage. If data or the system is not available because of a Denial of Service (DoS) attack to the medical system, the medical device may not function at all.

With these scenarios in mind, we need to look for a design methodology, which bears safety and security in its architecture. In the functional safety domain, the use of a separation kernel is a well known solution for safety and security by design concept. In a nutshell, a separation kernel allows spatial and temporal separation between applications, by providing **separated partitions** for application execution and a concept for **sharing CPU time**. A thoroughly separated application concept by using a separation kernel guarantees non-interference, so that errors cannot propagate from one application to another.

From a safety perspective this partitioning mechanism is used to isolate applications of different criticality so that risk reduction can be applied for each application/partition independently. The application isolation capabilities are also suitable for system security by isolating critical data from non-critical data and allowing only controlled information flow between isolated partitions. Controlled information flow can be realized by:

- A white list policy for inter-partition-communication. That is, communication is generally prohibited and needs to be allowed explicitly, defined at system configuration, not at runtime.
- Using cryptography before exchanging data between partitions and with the outside world
- Applying a security policy for data communication so that communication is monitored and controlled

This paper uses the PikeOS Separation Kernel as an example implementation and explains how safety and security aspects are covered by the separation kernel architecture. An automotive use-case designed by Continental and an IoT gateway design based in the NXP/Freescale IoT gateway will be introduced by describing the chosen safety and security concept.

2 Separation Kernel - Safe and Secure Real-Time Virtualization

A well-known approach for separation is an Hypervisor, which is used in the IT domain to run an operating system on top of another operating system. Most Popular implementation like VirtualBox, VmWare etc. provide an execution environment to run a COTS operating system like Linux on another COTS OS. As the

afore-mentioned technologies were never architected with safety and security in mind, this separation does not allow enough safety and security for certification.

A separation kernel like PikeOS follows a slightly different approach. It uses a microkernel as the underlying operating system architecture and provides means for partitioning on top of the microkernel. The partitions can be used as an execution environment for:

- Bare metal applications
- Applications using the microkernel API, PikeOS Native.
- Using a standard API like POSIX.
- Using any run-time environment like Java, Ada including Ravenscar profile.
- Running an operating system like Linux, AUTOSAR, ARINC-653 in a partition …

The main advantage of this approach is, that the microkernel provides real-time capabilities and separation capabilities in the operating system design. Thus we have a Real-time Operating system and a hypervisor architected into one product.

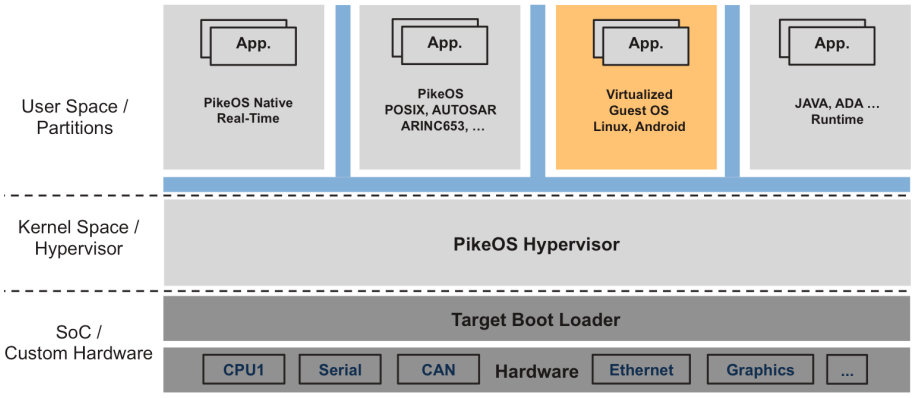

Figure 1: Separation Kernel architecture

The architectural concept of a separation kernel is based on separating resources, so that applications cannot interfere with each other. The available resources on computing hardware are the physical hardware components and the CPU execution time. The separation of physical resources is called spatial separation or resource partitioning, while the separation of the available execution time is known as temporal separation or time partitioning.

2.1.1 Divide-Et-Impera – Resource Partitioning

Spatial separation is achieved by means of resource partitioning, in which system resources such as memory, files, devices, secure communication channels and cores are statically assigned to different resource partitions. All applications run in the context of a resource partition and PikeOS ensures that during runtime an application has guaranteed access to the assigned resources of its partition and these resources are not accessible from applications belonging to other partitions.

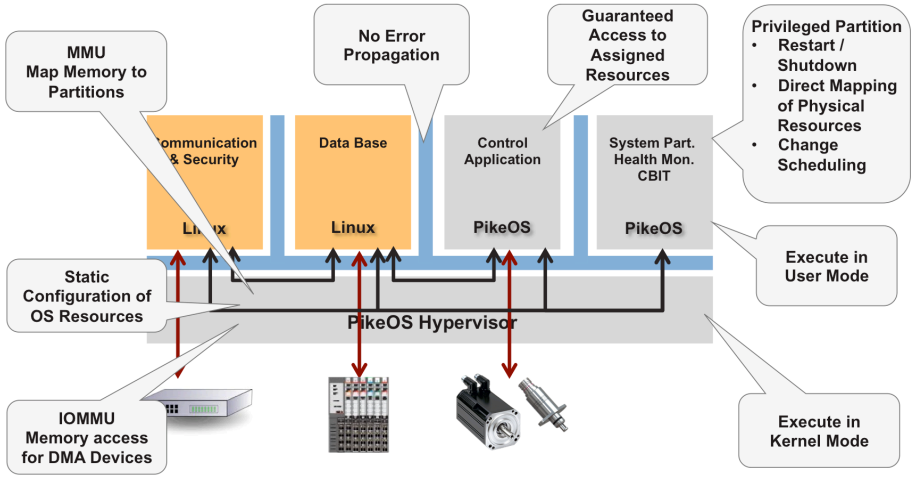

Figure 2: Resource Partitioning

Resource partitioning is enforced by using the MMU to control access to the available resources. Each hardware device is somehow represented by a physical address in order to access this device. Resource partitioning is realized by using the MMU to map a certain memory are into a partitions virtual memory space and allow or deny access to this RAM and I/O devices. The configuration of the MMU is done statically at system startup in the PikeOS microkernel and it is **not** modifiable at run-time, so that a hacker cannot modify the resource configuration later on.

DMA capable devices can bypass the MMU protection and access memory areas, which are reserved for the operating system or an application partition. To control the memory access of DMA bus master devices, the SoC provides an additional MMU device, which maps device-visible virtual addresses to physical addresses. The processors vendors name this mechanism differently so that NXP/Freescale (PowerPC) is talking about a Platform Management Unit (PAMU), ARM processors name this a System Memory Management Unit (SMMU) and Intel calls this VT-D on x86 architecture.

Last but not least the privilege level of the processor is used to configure partitions to run in user-space and the operating system to run in kernel-space. As user-space applications have no direct access to kernel-space, the operating system is safe against any misbehavior of user-space applications. In case of a separation kernel, partitions run in user-space (see Figure 1).

In summary, the separation makes sure that:

- A partition has guaranteed access to devices, which are configured only for the partition
- A partition has no access to a device, if it has not been configured for accessibility
- That errors cannot propagate from one partition to another partition

2.1.2 Time-Partitioning

Temporal separation is provided by means of Time Partitioning in which the CPU time is divided into time windows. The duration and order of these time windows are statically defined and enforced by the PikeOS Microkernel during runtime. Applications are assigned to one or more of such time windows and are considered for scheduling only when the associated time window is active. In this way the temporal behavior of a partitioned application is made independent from the rest of the system.

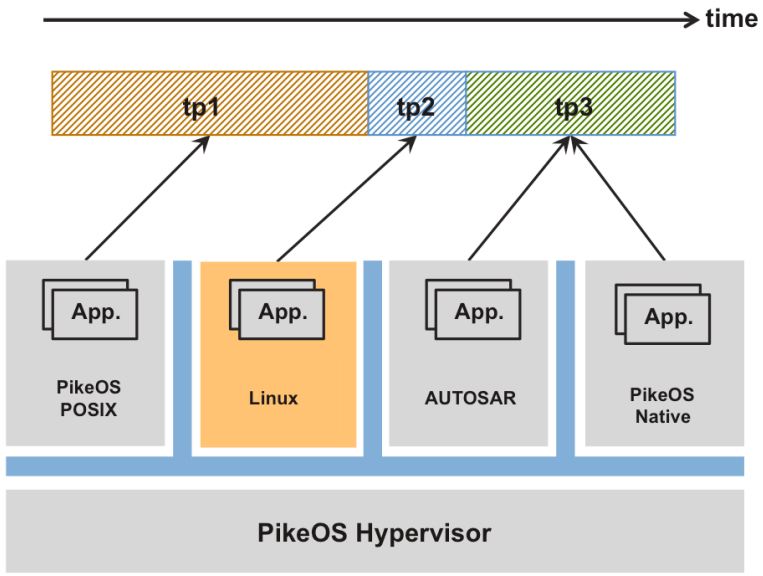

Figure 3: Time Partitioning

Priority based scheduling and a preemptive kernel design provides real-time capabilities to a microkernel, but this does not always guarantee a calculable deterministic behavior of the system. The more complex a system configuration is (multi-core is one of the major complexity issues), the more difficult it will be to determine a worst case execution time (WCET) for an application running in this configuration. The WCET

analysis is evident for the safety as it is the guaranteed maximum time for the system's response. Time partitioning demerges the complexity of the run-time behavior of a system by assigning a partition's applications to a dedicated time window. If this time-window is active, only the defined subset of applications is executed.

As pure time partition would not allow the handling of exceptions or the reuse of non-consumed time, adaptive time partitioning provides the required flexibility. The patented time-partition zero (tp0) concept of SYSGO defines an additional time partition (tp0), which is always overlapped to the current eligible time-partition. If an executable (thread) in tp0 has a higher priority than threads in the current active time-partition, than the high-priority thread in tp0 will be eligible to run. The opposite example is, if there is nothing to do in the active time partition, threads from tp0 are recognized for execution, thus allowing low priority threads assigned to tp0 to use not-used execution time.

Complex scenarios with multiple cores and multiple applications can have a complexity, which makes it impossible to calculate a WCET with the available means. The time partition based approach reduces the complexity of a system, so that the WCET is able to be calculated and stays is a reasonable range.

2.2 Separation Kernel Safety

An Application, which runs in a partitioned system, is not safe per se. The application safety is defined by a set of safety requirements, which need to be implemented in software and/or hardware. The important aspect of the separation Kernel is the possibility to design safety-critical software with the means provided by the separation Kernel. The major safety concept is the separation of application and the controlled information flow between the separated entities.

Industrial safety standards require the certification of software, which means that application code must be thoroughly tested and documented. These standards define several levels of criticality, which are measured by the severity of the hazards caused by application failure. In case of EN-50128 and IEC-61508 the criticality is ranked from Safety Integrity Level (SIL) 0 being the most basic up to SIL-4 being the most stringent level. The avionic safety standard DO-178B/C defines Design assurance levels ranked from E (basic) to A (strict). As separation kernels allow noninterfering application partitioning, the system can be designed with mixed criticality levels for each partition. That is, each partition can host an application, whose criticality to the system safety is different. This concept is market proven and has undergone several certifications according to industrial standards like IEC61508, EN50128, DO-178B and ISO26262.

Safety certification standards define strict guidelines for the development and deployment life cycle in order to achieve a conformity certificate. Beside the strict processes, extensive testing is required for each line of code. Test code and testing effort grows with the criticality level of the application. As the costs grow drastically with each additional line of code, there is a clear advantage to separated critical from non-critical application and keep the "Trusted Code Base" (TCB) for critical application components as small a possible.

Another important aspect for safety is the above-discussed WCET and thus the real-time behavior of the underlying operating system. Safety critical software requires a detailed timing analysis in order to prove that the software can react accurately in critical situations. The time partition concept demerges the complexity of the runtime behavior and eases the timing analysis significantly.

2.3 Separation Kernel Security

While safety is the attempt to protect the user from harm, resulting from malicious behavior of the system, security aims to protect the systems against malicious attacks from the humans, interfacing with the system. Security architects have the following aspect in mind, when designing a system:

- **Integrity:** Make sure, that the data cannot be altered without authorization
- **Confidentiality:** Make sure, that the data cannot be accesses by anybody who is not authorized to access the data
- **Authentication:** Prove the identity of a person who requests access to the system
- **Access control:** regulate the access to the system
- **Availability of resources:** Make sure that the system is available and is not hindered in operation

- **Non-Repudiation:** Have a proven concept so that nobody can deny his access to a device later on

In general-purpose platforms, these concepts are realized by using state of the art security technology like

- Firewall technology that monitors and controls the incoming and outgoing network traffic
- Authentication mechanisms to make sure that only authorized personal has access to the device
- Cryptography to encrypt data before transferring it over vulnerable transmission channels
- Monitoring and perimeter protection in order to identify and actively protect against intrusion

The usage of a separation kernel enriches these available security concepts with a fundamental security architecture, which we name as “security by separation and controlled information flow”.

“Security by separation” means, that sensitive data can be isolated into a partition and “Security by controlled information flow” means that access to the isolated sensitive data is controlled by the configuration of the communication means used for inter-partition-communication. The most secure approach is not to allow any communication, but this is not in the intention of the inventor. Communication from and into a partition with sensitive data can be secured by using/implementing a security policy manager, which captures all communication data and applies a security check (based on cryptography algorithm if required) on the origin and the content of the data.

Security certification standards such as IEC-15408 (Common Criteria) or the IEC-62443 specify the security functional and assurance requirements which have to be fulfilled by the device and thus by the underlying software platform. The IEC-15408 describes the security requirements for general-purpose equipment and the IEC-62443 targets Industrial Automation and Control Systems (IACS).

Common Criteria (IEC-15408) provides assurance that the process of specification, implementation and evaluation of a computer system (IoT devices are computer systems) has been conducted in accordance to the standard. An Evaluation process validates the security claims, which have been made on the device, so that the standard defines several Evaluation Assurance Levels (EAL), beginning from 1 being the most basic up to 7 being the most stringent level describing the depth and rigor of an evaluation.

To have a more generic certification, SYSGO has an ongoing project to certify PikeOS to CC EAL5+. It is important to know, that general purpose computing platforms (like Windows and Linux) have achieved EAL4 so far, but separation kernels like PikeOS can achieve EAL6.

2.4 Safe&Secure Automotive application

The high level of connectivity in a car makes it a perfect example of an IoT device. The Security of a car was heavily discussed in the press because a couple of automotive solutions were hacked in the past month. Specifically the hack of the Jeep Cherokee demonstrated, that a security leak could cause severe safety problems. Videos on the web show, that a hacker is able to steer the car and even operate the breaks.

The implementation shown in Figure 4 is a combination of a cockpit showing the cars speed and rotation speed. Motor status information is received over the CAN bus and displayed on a dedicated display which is integrated in the cockpit. Speed and rotation speed indicator reside in a dedicated partition. The car status is calculated and controlled in another partition.

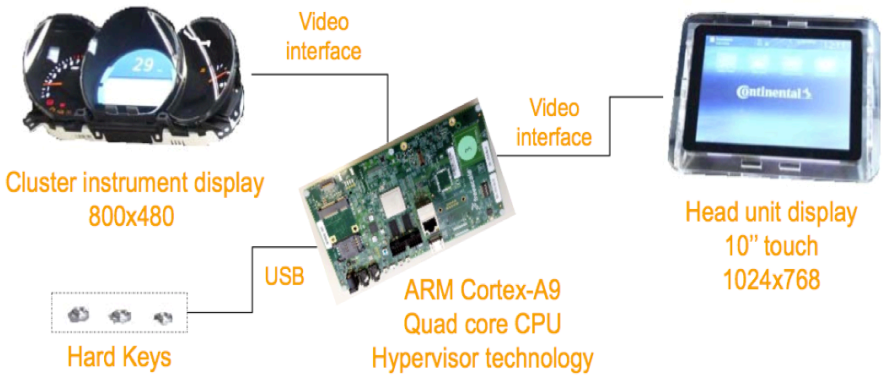

Figure 4: Continental Integrated Cockpit and Infotainment design based on PikeOS

The head unit display shows an Android, which runs in a dedicated PikeOS partition. Android is used to provide infotainment and navigation functionality. Browsing the Internet and downloading and installing Android apps offers the possibility to configure the head-unit according to the end-users preferences.

Android is the interface to the outside world and can be subject to attacks from the outside world. If a hacker is able to highjack the Android head-unit over a known vulnerability, than even PikeOS cannot prevent the access to the Android system. Starting from the Android partition the hacker will try to access physical devices in order to change system behavior. Or he may try to modify the operating system configuration in order to get control over the entire cockpit. Or, he may try to access other partitions (speed or car status) in order to modify functionality. Our Separation Kernel will deny all tries from the hacker to access I/O devices if they have not been assigned to the Android partition. Accessing the operating system from the Android partition will not be possible because the Android partition operates in user space and does not have the privileges to access the OS.

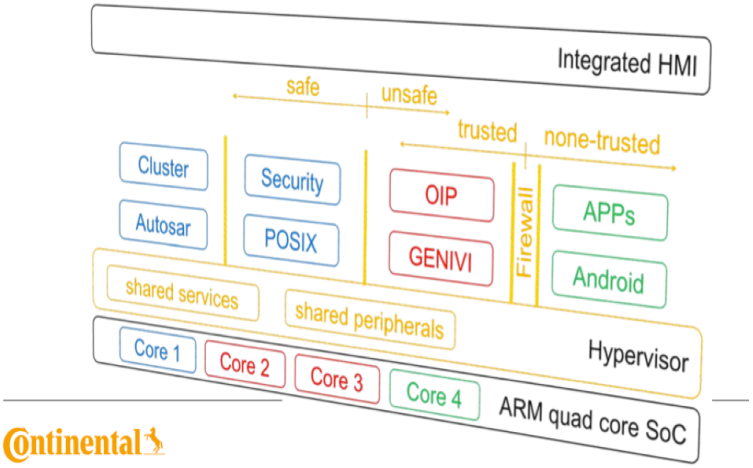

Figure 5: Integrated Cockpit and Infotainment architecture

2.5 Safe&Secure IoT Gateway

The NXP/Freescale Layerscape 1021A (further named as LS-1) is an ARM based SoC (see Figure 6). The I/O offered by the LS-1 is quite suitable for SOHO gateways. NXP/Freescale provides an IoT gateway reference design, which is supported with a PikeOS board support package (BSP) so that partitioning based safety and security can be implemented. The LS-1 uses the NXP/Freescale QorIQ framework so that secure-boot and secure-update can be implemented by using the Trust Architecture.

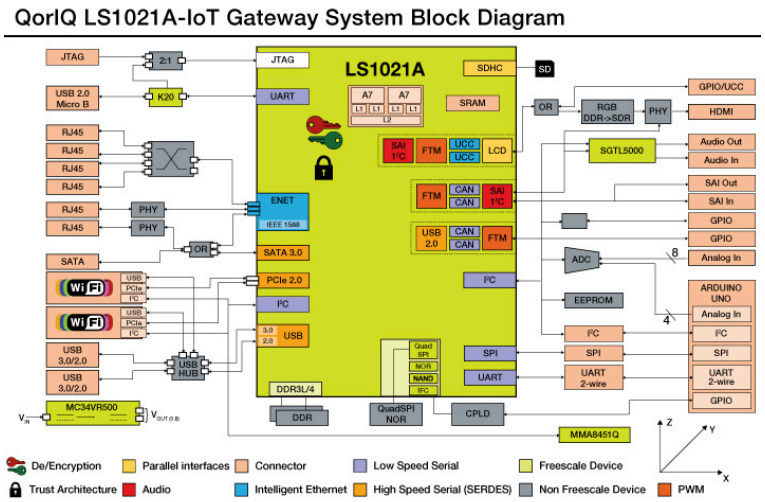

Figure 6: Freescale IoT Gateway Block Diagram

The implementation shown in Figure 7 was derived from the idea of a customer to offer a gateway, which offers the end-users the flexibility to customize their Linux domain with any open-source software downloaded from the Internet and provide home automation functionality. The service provider, who sells/leases the gateway to the end-user owns a dedicated Linux domain which allows him to offer services to the end-user by uploading the service software to his service-provider domain. The data of each Linux partition is separated by using a dedicated network, hosted by dedicated PikeOS partitions. The storage is managed by a PikeOS volume provider/manager. The volume manager provides a dedicated volume to each Linux domain with configured access rights. The two Ethernet interfaces are separated so that network problems do not propagate into the other partitions.

The final implementation offers direct IO access from the Linux-1 domain to the underlying hardware. Thus additional home-automation functionality can be implemented in the user-domain.

The shown architecture offers a certain level of security by isolating the data of the user domain and the service-provider domain so that a user cannot access the service provider's configuration and the service provider shall not access private user data. At a first glance, safety is not a demand for this kind of devices, but recognizing the fact that the gateway does also provide home automation functionality, the separation based security will ensure the safe operation of the home automation. All partitions can be updated independently, so that service and user applications can be updated on the fly without shutting down the entire system.

The idea to extend a gateway or router with home automation functionality in not new and major vendors in this market (like AVM, D-Link, etc.) have adopted this idea with their new generation of products. Their security concept is mainly based on using Linux security means and applying security updates. A PikeOS based security architecture can also make use of Linux security means and offers additional security by separation. If a hacker is able to highjack the Linux system over an unpatched vulnerability, the hacker will try to access physical devices in order to change system behavior. Or he may try to modify the operating system configuration in order to get control over the entire system. Or, he may try to access other application threads in order to steal data or modify functionality. But, as we have seen in chapter 2.4, the separation kernel will deny all tries from the hacker to access I/O devices and the operating system. Stealing data from another partition can be prohibited by securing the inter partition communication with a policy manager, which captures all communication data and applies a security check (based on cryptography algorithm if required) on the origin and the content of the data.

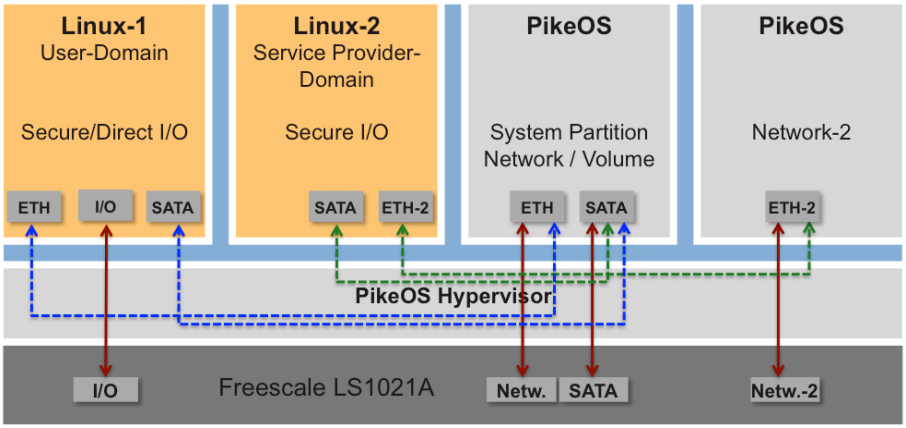

Figure 7: Safe and Secure IoT gateway architecture

3 Conclusion

IoT targets seamless communication between all devices from sensor level up to the enterprise level. These interconnection leads to a more vulnerable system design, which bears the risk to be a victim of a security attack. Specifically for IoT devices such as industrial control system, medical devices, transportation infrastructure, just to name a few, security attacks can compromise the systems safety by modifying the system configuration.

Security and safety is not a functionality, which you can just enable for a device. They need to be implemented following safety and security requirements and by sticking to the rules of industrial safety and security standards. Implementing safety and security becomes easier, if the underlying software platform provides an appropriate architecture for safe and secure design. An RTOS that is also an hypervisor based on a separation kernel like SYSGO PikeOS offers safety and security by separation and controlled information flow. Applications are isolated into partitions, which are protected against each other. The partitioning is non-interfering so that errors cannot propagate over partition boarders. The communication between partitions is configured and controlled so that access to safety and security critical data can be limited/prevented efficiently.

PikeOS has proven its safety and security concept by achieving a couple of safety and security certifications:

- IEC 61508 SIL-3
- EN-50128 SIL-4
- DO-178C DAL-A
- IEC 15408 EAL 5+ ongoing

Even more, PikeOS has achieved the worlds first EN 50128 SIL-4 certification on a multicore CPU.