White paper

Embedded Safety and Security Leveraging the PX5 RTOS

By Bill Lamie

Published January 25, 2023

Embedded Safety and Security leveraging the PX5 RTOS

By Bill Lamie, PX5

Today, safety and security for embedded devices are paramount. While distinct requirements, they have a high degree of overlap in the embedded PX5 RTOS context. For one, the PX5 RTOS can help in avoiding memory corruption, the most common source of safety and security issues in embedded devices. Beyond that, the PX5 RTOS safety and security features are part of a greater defense-in-depth solution that includes the PX5 RTOS, application software, device hardware, and other network/cloud entities and their configuration/settings.

Safety and security needs depend on the application. They ultimately encompass the attack surface as well as what is practical from a technological and business standpoint. Stated another way, embedded safety and security isn't a one-size-fits-all approach, but rather a deliberate risk-benefit analysis based on each use case.

Hardware Safety & Security Features

As for hardware safety and security, a wide variety of embedded processors feature an even greater variety of safety and security features, the most common of which include:

1. Anti-tampering. This hardware feature protects the firmware IP of the device from unauthorized access. It is generally outside the scope of the PX5 RTOS or application firmware but an important consideration when selecting hardware.

2. Lock-step execution. This hardware feature employs multiple processors executing the same code with the same data, the goal being that exact code execution is guaranteed. Such hardware is mostly found in safety critical applications. This is effectively invisible to the PX5 RTOS or most of the application firmware.

3. Anti-glitch. This hardware feature employs circuitry to prevent an attacker from causing abnormal program execution via manipulating power or other system signals. This too is generally outside the scope of the PX5 RTOS or application firmware but may be important when selecting hardware.

4. Execute only from flash. Most microcontrollers (MCUs) execute instructions from flash. Some of these MCUs can prohibit execution from RAM, which is recommended to help prevent dynamic insertion of malicious code in remote execution attacks. This sometimes relies on the application firmware to enable, but otherwise is invisible to the PX5 RTOS or the application.

5. Hardware stack limit. Some processors have a stack limit register that guards against memory corruption caused by stack overflow. This is generally implemented as an additional register and is set up by the PX5 RTOS on each thread context switch.

6. Hardware watchdog timer. Many processors have a non-maskable hardware watchdog timer. This feature acts as a failsafe. Under normal operation, application code resets the watchdog regularly and always before its expiration. During abnormal execution, the watchdog is likely not reset, thus leading to a non-maskable interrupt that halts the abnormal execution. Typically, applications will simply reset after a watchdog expiration.

7. True Random Number Generator (TRNG). Having a TRNG or even more basic Random Number Generator (RNG) in hardware is very beneficial. This is most important for networked devices, but it's also useful for the PX5 RTOS – especially for the unique PX5 RTOS Pointer/Data Verification (PDV) feature.

8. Memory Management Unit (MMU). Typically only available on larger, more powerful processors, this hardware feature enables access restrictions for memory regions. This is most often set up once by the application firmware after a reset to map and protect memory regions.

9. Memory Protection Unit (MPU). This hardware feature is similar to the MMU but found in smaller, more resource constrained devices. Again, this is typically set up by the application firmware after a reset to protect various memory regions. When there aren't stack limit registers, the MPU can set up a protected block at the top of each thread's stack in order to prevent stack overflow. This functionality would need to be accomplished inside the PX5 RTOS thread context switching logic.

10. Secure Element (SE) or Trusted Platform Module (TPM). For network-connected devices, having a SE or TPM for secure cryptographic functionality can greatly increase the network security of the device and is therefore highly recommended.

Of course, each of the hardware safety and security features mentioned have an associated cost in circuitry, power consumption, size, etc. These costs must go through the risk-benefit analysis mentioned previously.

Software Safety & Security Features

Software support is required to utilize the hardware safety and security features, e.g., setting up the hardware stack limit feature or the MPU to protect certain memory areas. The PX5 RTOS binding layer includes such support. The application firmware may also have logic to support hardware safety and security features.

PX5 RTOS Pointer/Data Verification (PDV)

For software-only safety and security measures, the PX5 RTOS provides patent-pending Pointer/Data Verification (PDV), a software-only technique to help detect corruption of important data like function pointers, function return addresses, internal system objects, allocated memory, etc. PDV uses the pointer or data value, the storage location of the verification code, and the unique run-time

identification provided to *px5_pthread_start* to create a verification code (fingerprint) for each important data element during its initialization. Before the important data is used, it is authenticated against the verification code. If corruption is detected, the application is alerted via the PX5 RTOS central error handing, at which point the application can respond to the memory corruption. PDV helps detect memory corruption early and greatly reduces the chance of unwanted execution associated with function pointer corruption.

Run-Time Stack Checking

Another software-only safety and security measure is run-time stack checking offered by the PX5 RTOS. Simply build the PX5 RTOS source with *PX5_STACK_CHECKING_ENABLE* and each thread's stack is checked throughout PX5 RTOS execution. The application may also utilize the *px5_pthread_stack_check* API to perform stack checking from the application C code.
If a stack corruption or overflow is detected, the run-time stack checking calls the central error handling function with a fatal error. If stack checking detects a stack that is at risk of overflow, the central error handling is called with the appropriate advisory error.

Application Best Practices

In addition to PDV and run-time stack checking, there are application *best practices* for enhanced safety and security. These include:

1. Harden the device firmware. The PX5 RTOS was implemented using Test Driven Development (TDD),, in which the tests are written before the code. Furthermore, by design, the PX5 RTOS code base must achieve 100% statement and 100% branch/decision coverage testing before each minor release. We recommend the application firmware take a similar approach: there is never enough testing. The better vetted the software is, the safer and more secure it is.

2. Leverage static analysis and related tools. In addition to hardening, leverage static analysis tools as well as penetration testing and fuzzing tools. These tools help find subtle issues in advance, which is often much easier than debugging fielded devices.

3. Use PX5 RTOS Pointer/Data Verification (PDV). The PX5 RTOS optionally (as determined by *px5.c* build options) uses PDV to verify function pointers, stack integrity, internal system objects, and allocated memory. Through API extensions, the application can also use PDV to verify its important function pointers and data. PVD enables early detection of memory corruption, greatly reducing the possibility of unwanted program execution, including malicious remote execution.

4. Use an adequate (or larger) stack size. Stack overflow is the leading cause of memory corruption in embedded systems. Each thread stack must have enough memory for its worst-case function call nesting – including all local variables in each function. If not, the stack may overflow into the memory directly preceding the stack. This problem can be mitigated by using hardware stack limit features or the MPU/MMU to guard the area directly above the stack. The PX5 RTOS stack checking features are also helpful in preventing stack overflow issues.

5. Use the PX5 RTOS run-time stack checking. The PX5 RTOS run-time stack checking is an easy way to detect (and correct) stack overflows.

6. Explicitly specify and check buffer sizes. In all functions where a buffer is supplied, the caller should explicitly provide the size of the buffer and the callee should explicitly check the size to avoid overrun. The PX5 RTOS does this internally and the application firmware should as well.

7. Be mindful of more likely areas of memory corruption. When a thread stack overflows, it generally corrupts the memory immediately *preceding* the thread stack memory (in most architectures, thread stacks grow toward lower addresses). In contrast, buffer overflows are more likely to corrupt memory immediately *following* the buffer. Knowing this, it might be safer to avoid placing critical data before stacks or immediately following buffers.

8. Use PDV to place markers before stacks and after data buffers to help detect memory corruption caused by stack and buffer overflows. Please see the *px5_pthread_pdv_** APIs for more details.

9. Be careful with function pointers. Function pointers provide an easy path to unwanted program execution. For example, it's not good practice to place function pointers inside of buffers since a buffer overflow could overwrite the function pointer. This is the easiest way for an attacker to initiate unwanted remote execution. Of course, of the PX5 RTOS PDV feature can verify application function pointers before they are called, which helps mitigate this issue.

10. Test, test, and test.

Summing Up

Safety and security of embedded systems is more of a spectrum of requirements based on the use case rather than one fixed set of hardware and software requirements. The PX5 RTOS provides useful tools to enhance your device's safety and security. However, it is merely one element of a defense-in-depth that includes hardware, software, and even IT/network security. Please contact us today to discuss further!

Enhance • Simplify • Unite

11440 West Bernardo Court • Suite 300
San Diego, CA 92127, USA

Phone: +1 (858) 753-1715
Email: info@px5rtos.com
Website: px5rtos.com