WIND™

Five Steps to Improving Security in Embedded Systems

WHEN IT MATTERS, IT RUNS ON WIND RIVER

EXECUTIVE SUMMARY

Headline-grabbing security breaches underscore the need for stronger protective measures in critical embedded systems, particularly those that control vital infrastructure, industrial operations, intelligence and defense networks, and even medical devices. While security breaches have been occurring for many years, in today's increasingly interconnected world, they are becoming more prevalent with escalating complexity challenges. How can embedded device developers balance the need for tighter security with competing business and market demands, especially given the realities of strict budget constraints and aggressive deadlines? This paper outlines five steps for building additional security assurance into embedded devices by considering the whole product life cycle—from design and inception, through development and testing, to delivery and maintenance.

TABLE OF CONTENTS

WAKE-UP CALL FOR INCREASED EMBEDDED SECURITY

In 2010, the Stuxnet worm became the first malware with the ability to break into industrial infrastructure and allow an attacker to take control of critical systems. This worm was designed to attack the centrifuges used in Iran's nuclear program, presumably setting the program back by years. Stuxnet could infiltrate programmable logic controllers (PLCs) and install malicious code on them. For example, the PLCs could be programmed to run a centrifuge into a destructive spin, overriding safety lockouts.

Security is not a new concern for certain classes of embedded systems. Military and government systems hold security as one of their primary design goals. What is emerging is the heightened awareness of commercial control systems and critical embedded systems that work in our key infrastructure. With the increasing connectivity of commercial embedded systems to desktops, laptops, mobile communication platforms, the Internet, and cloud architectures, many PLCs were not designed to ward off network intrusion; designers typically considered the private control system network to be secure. Stuxnet, for example, could attack through desktop or laptop computers hooked into internal networks that host the target control systems. Threats today are even more sophisticated and can come from indirect sources, forcing embedded designers to include more device safety and security requirements into their core product designs.

Striking an Appropriate Balance

Device security has become a high priority or a "must-have" capability. But it can add platform requirements that increase development complexity and, ultimately, cost. In addition, a wide number of vulnerable embedded systems with inadequate security strategies are already deployed in highly networked, interconnected environments. The demand for cloud computing increases device connectivity, which further complicates the security landscape because unsecure devices on the network introduce additional weak points for attack.

An appropriate balance of business requirements with security, functionality, and performance of the device in the intended market needs to be achieved. Securing a device must not cause its delivery to extend beyond the forecasted schedule or cost. Security is an important component of many devices. But it may be marginalized because some designers do not see it as a differentiating feature but rather as an expected capability.

Device designers should assess three things: 1) the importance of the assets on their products that need to be made secure, 2) the anticipated operational environment's threat exposure, and 3) the appropriate measures to be taken to protect the assets for mitigating the threats. In other words, a determination of an appropriate level of security needs to be established, one that is relevant for the device, its marketplace, and its intended deployment environment. Securing a device is a continuous effort that spans the entire life cycle of the product, from architectural design through deployment and end of life. Planning and budgeting for safety and security updates throughout the entire product life cycle, along with future threat protections, are essential for any connected device environment.

Increasing Complexity and Connectivity

Embedded devices have traditionally run in relative isolation and have therefore been protected from a wide range of security threats. Today's devices, however, are often connected to corporate networks, public clouds, or the Internet directly. Defense and intelligence networks also plan to embrace cloud capabilities and leverage commercial smartphone platforms while maintaining highly secure domains. This wide connectivity yields substantial gains in functionality and usability but, as discussed previously, also makes devices more vulnerable to attack, intrusion, and exploitation. This new "connected era" is elevating secure connectivity to an essential system requirement where it may not been a top concern in past projects. Often the embedded hardware and software from the previous-generation devices were not designed to enable secure connections or to include network security components, such as firewalls, intrusion protection, or other security-focused functionality. Additionally, developers cannot assume network environments will be private and protected, nor can they predict how their devices might be connected in the field. And they cannot predict the impact of future connected devices on their products.

The most efficient way to strike the correct balance between device capability and security is by defining and prioritizing the device security requirements with the rest of the system and its deployment environment, including the network environment. For maximum efficiency, this should be done early in the product life cycle.

Cyber-security and Information Assurance

In many cases embedded systems differ from enterprise IT systems in that they directly control processes and equipment that are part of key infrastructure. The key security concern is the possibility an enemy can break into the system and control devices remotely and shut them down, make them behave abnormally, or worse, cause equipment failure and destruction. In the enterprise world, security priorities are associated with information breaches and sensitive data exposure. However, for embedded device security, it's important to consider both the safe and correct behavior of the system and the data it stores or transmits. "Cyber-security" is a relatively new term and is used interchangeably with "information assurance." Information assurance, however, has a formal definition: "measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and nonrepudiation" (National Information Assurance Glossary, CNSS 4009).

Cyber-security is being used more generally in the press and is defined in the dictionary as "measures taken to protect a computer or computer system (as on the Internet) against unauthorized access or attack." In this paper, and to align with how the term is being used in the embedded systems market, cyber-security means protecting an embedded device from attack to ensure correct and safe operation in its operating environment while still protecting sensitive data (as applicable).

For information assurance designers, the focus is on protecting and preserving data that the devices store or transmit. This data is typically confidential and, in government and military applications, can be highly sensitive, such as top secret data. The embedded device must be designed so that it is very difficult for an attacker to gain access to it, thus maintaining the integrity of stored or transmitted data. This protection should guard against a wide range of threats, from external network-based attacks to physical access to the device.

No manufacturer wants its devices to be easily disrupted by attacks or easily vulnerable to losing sensitive data. Many classes of devices do not handle sensitive data, yet their correct and safe operation is paramount. As machine-to-machine (M2M) communication expands in our infrastructure (e.g., the smart electrical distribution grid, cloud computing environments), automation devices become safety critical and must properly handle confidential information in a complex, connected environment.

GUIDELINES FOR IMPROVING EMBEDDED SYSTEMS SECURITY

Embedded systems designers should consider the following five high-level steps when addressing security. This is not a prescriptive methodology, yet it is intended to highlight an approach that looks at embedded security as a series of development life cycle decisions.

Step 1: Conduct an End-to-End Threat Assessment

Improving the security of an embedded device starts with identifying the potential threats. These threats must be evaluated in the context of the device manufacturer, operators (if the device is provisioned in such a way), and end users, including their usage

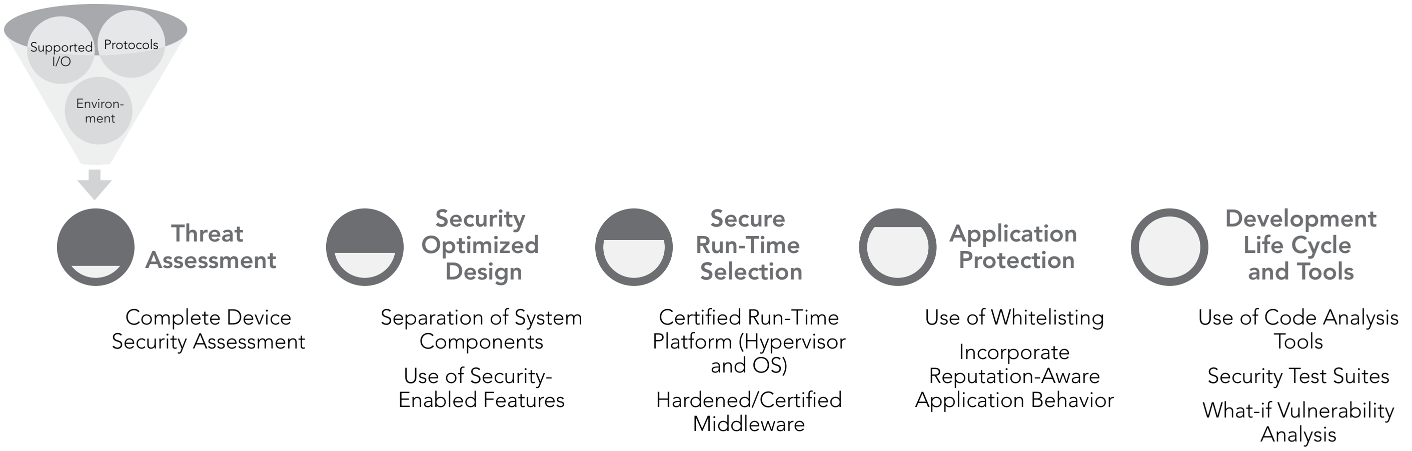
Supported I/O
Protocols
Environ-ment
Threat Assessment
Complete Device Security Assessment
Security Optimized Design
Separation of System Components
Use of Security-Enabled Features
Secure Run-Time Selection
Certified Run-Time Platform (Hypervisor and OS)
Hardened/Certified Middleware
Application Protection
Use of Whitelisting
Incorporate Reputation-Aware Application Behavior
Development Life Cycle and Tools
Use of Code Analysis Tools
Security Test Suites
What-if Vulnerability Analysis

Figure 1: Five steps to improving security in embedded systems

environment. Threats are described in terms of an attack vector (how the attack is perpetrated on the device) and the vulnerability it exploits (the weakness or fault in the hardware or software that allows the attack to exploit the device). Examples of attack vectors include a wired Ethernet connection on the device used for communication, and common services such as web (HTTP), FTP, SSH, and debug agents. Examples of vulnerabilities include weak or default passwords, coding errors such as not protecting against or checking for stack or buffer overflows, or even design errors such as how the device powers up. The difficult part of security design is predicting and preventing the attack vectors and vulnerabilities in advance.

A device needs to be evaluated within a large scope if a security threat assessment is to be successful. Indeed, many current security threats result from thinking a device won't be used in a certain way. Stuxnet exploited PLCs that were on the same network as infected desktop and laptop computers. Although not initially connected to the Internet, it's likely that private control system networks will have diagnostic or development computers connected at some time. In evaluating security threats, it is important to consider the larger picture: Operators (e.g., wireless network providers) and end users (e.g., electrical grid controllers) are part of the equation.

In end-to-end threat assessment of embedded devices the following needs to be considered, at a minimum:

- **A complete product life cycle analysis needs to be performed:** This analysis must include developers, manufacturers, operators, distributors, retailers, and end consumers to capture the total usage impact on device security. It is important to determine the priority of information assurance and cyber-security.
- **Possible entry paths for attacks into the system need to be defined and described:** Possible paths to consider include network access as well as other potential paths. For example, is it possible to compromise a device via physical access such as USB or serial ports? Once the entry points are ascertained, the attack possibilities need to be evaluated. For example, is the device vulnerable from web access via TCP/IP port 80? Does the device use a firewall? If not, what TCP/UDP ports are open? Similarly, for physical machine access, can the device be booted from an attached USB device? An analysis is required of the permutations possible from physical access plus system vulnerabilities.
- **A risk matrix needs to be built:** Because of the vast number of permutations possible, a risk assessment needs to be performed. What is the probability of an attack via each channel? What is the impact of exploitation via this channel?
- **A mitigation strategy needs to be created based on the priority list:** For example, partitioning a system into higher segments of higher and lower security might be a sound strategy. In some cases, this can lead to more complex architectural and design requirements for the device but can dramatically reduce testing and update costs later in the product life cycle.
- **Creation of a design specification that includes security needs to be done based on the previous assessments:** This is part of the overall system design but should be treated with high priority. An overall plan for designing, implementing, testing, and maintaining security features and mitigations needs to be part of any existing or new development plan.

Step 2: Leverage Existing Advanced Security Designs

A number of technologies and design methodologies have evolved to address the ever increasing threats to connected devices. An increasingly important paradigm for advanced security designs is using proven commercial off-the-shelf (COTS) system components that can improve device security while controlling costs. Examples of such security components include embedded virtualization, operating system partitioning, and middleware, and partitioning these components into virtual run-time environments for increased separation and abstraction.

Virtualization is gaining popularity in embedded systems because it enables multiple operating systems to run on a shared hardware platform. This provides flexibility in system design and allows designers to get more out of their hardware than with single-OS systems. In addition, it can provide a foundation for partitioning a device's operations into virtual execution environments, which can enable the separation of concerns and facilitates deploying higher criticality components with less sensitive code on a shared platform.

Application partitioning is a useful technique for separating safety-critical or security-critical portions from the less important system functions. For example, a multi-OS system might partition a real-time operating system (RTOS) from the general purpose operating system (GPOS) such as Microsoft Windows or Linux.

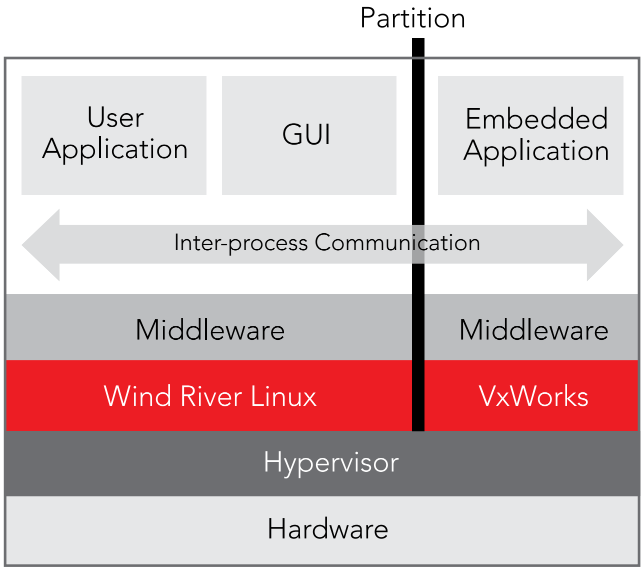

Figure 2: Example of a partitioned architecture using a hypervisor

Vulnerabilities and attacks on the GPOS would then have minimal impact on the RTOS partition, thus keeping the device operational. Figure 2 shows an example of such a system.

Component reuse includes more than just reusing your own code in system design; at a higher level it means using existing components that have already met quality and security requirements. Use of COTS software is an example of component reuse at a high level—a proven way to reduce costs, time-to-market, and most importantly, budget and schedule risks. Many COTS products have gone through extensive security testing, validation, and certification and can be excellent candidates for reuse opportunities in next-generation device software design. Designers can immediately leverage the proven security features and artifacts of these products, features that would be difficult to develop and support commercially for a single product development effort.

Step 3: Select an Appropriate Run-Time Platform

Selection of an appropriate commercial run-time platform for an embedded system is a key consideration. Implementing a system with components that have COTS security evidence can increase the security and reduce the cost of development of the overall platform. There can be additional benefits of using COTS software components instead of roll-your own (RYO) code or self-ported and maintained open source code.

Some COTS run-time platform components that can help in developing a secure device include the following (see Figure 3):

- **Hardware support layer:** Run-time platforms such as hypervisors and operating systems require low-level hardware support layers that contain device drivers for the specific hardware devices. Relying on commercial hardware support is a critical first step in leveraging off-the-shelf components, but these should be acquired through known and trusted suppliers. Commercial offerings, such as board support packages (BSPs), are optimized for the target hardware, come with technical support and maintenance, and in some cases have certification evidence to support insertion into safety and security-critical environments.
- **Embedded virtualization:** To provide additional system integrity, an embedded hypervisor can provide virtualization to enable sophisticated partitioning schemes, multi-OS capabilities, and support of multi-core and other processor architectures. Running on top of a commercial hardware support layer, an embedded virtualization component with commercial certification evidence can accelerate and improve the security of devices with mixed-criticality partitions.
- **Real-time operating system:** Many embedded systems require small footprint, strict timing constraints, or safety/security certification; for these, a real-time operating system is the operating system of choice. The following should be considered in RTOS selection:
  - **Secure default configuration:** Is the RTOS available in a secure default configuration? For example, are nonessential services turned off and network ports blocked?
  - **Secure communications:** Does the operating system support services, such as a network stack, that implement and meet secure communications standards? Does the RTOS include encryption capabilities? Does the RTOS stack service allow for use of secure sockets, virtual private network (VPN), IPsec, and so on?
  - **Certification:** Does the RTOS need to have COTS safety and security certification that's applicable to your implementation? Has the stack been tested and passed security tests and verifications?
  - **Security response:** Does the vendor treat security issues with a high priority? Is there a security response team and process in place?

- **Embedded general purpose OS:** Standard Linux or Microsoft Windows general purpose operating systems are not considered extremely secure. They are best used in a partitioned environment, such as with an embedded hypervisor, that can provide additional security features. Running GPOSes in a partitioned environment enables their execution in isolated environments, separate from other, possibly more critical, parts of the system.
- **Middleware:** Depending on the system requirements, the middleware layer can include basic and advanced functionality in networking, wireless support, and communication security (IPsec, IKE, VPN support, etc.) as well as other services such as audio, video, and graphics. The network middleware selection is particularly important for supporting network security via encryption libraries and communication protocols such as SSL and IPsec. Additionally, intrusion protection can be enhanced via firewall and use authorization support at this level. Using certified middleware, such as products certified under Wurldtech's Achilles Certification program, is prudent for the building blocks of embedded systems. The selection of human interface middleware services (e.g., audio, video, and graphics) must be done with a security focus; otherwise these could lead to gaps in the overall system security and introduce vulnerabilities.
- **Virtual system simulation:** System, board, and processor-level simulation can accelerate software and hardware integration, providing ubiquitous platform availability, especially in embedded development life cycles where reference hardware may not be available. A system simulation tool enables sophisticated debugging, testing, and analysis techniques that hardware cannot provide.
- **Tools:** Tools for testing, debugging, and static analysis play a critical role in identifying improper design and preventing unsecure code from entering the project early in the life cycle. Test management and simulation tools greatly increase embedded development productivity.

Step 4: Secure the Applications

Modern embedded systems are much more than the traditional dedicated devices with a single mode of operation. Embedded systems now often host multiple applications and typically have their capabilities augmented through software and hardware updates and upgrades over the life of the device. As with desktop and server applications, it becomes critical that embedded software applications have secure capabilities because they are likely to be the target of malicious code or data breaches.

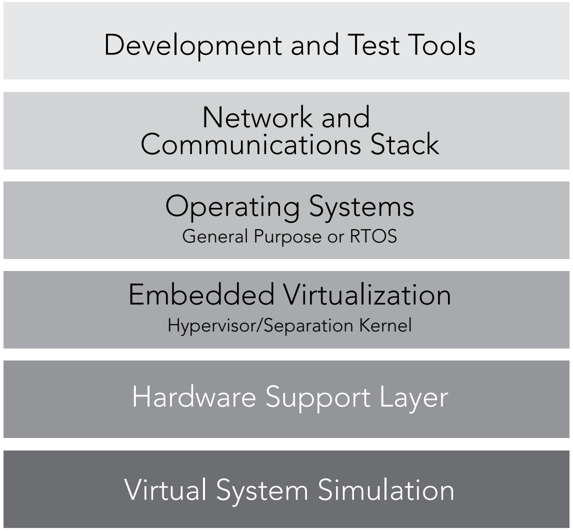
Development and Test Tools
Network and Communications Stack
Operating Systems
General Purpose or RTOS
Embedded Virtualization
Hypervisor/Separation Kernel
Hardware Support Layer
Virtual System Simulation

Figure 3: Embedded development and run-time solution stack

Updatable or upgradable embedded devices can use a technique called "whitelisting" to improve security. Whitelisting allows devices to only accept applications for download that are known to be safe for execution. Any software not on the whitelist will not be installed and will be rejected by the system. Blacklisting, a related technique that provides a list of known malware and viruses, is used by systems to reject downloading or installing any software that is on the blacklist. Because blacklists are much larger than whitelists and change much more rapidly, embedded devices typically lack sufficient computing resources to handle on-the-fly blacklisting and likely can't support the frequent updates, data storage, and network connectivity required. This makes whitelisting an attractive and effective technique for embedded systems.

Another technique that can be used to increase security involves evaluating the reputation of the source of data. When data arrives from sources with no reputation or a bad reputation, the application can determine what steps are required to verify the data's integrity or to ignore or block it altogether. Use of reputation awareness can also enhance system performance: When data is transmitted by trusted sources, applications can accelerate security screening and process the data more quickly.

Step 5: Adopt Comprehensive Life Cycle Support

Security constantly evolves as threats change over time. As a device becomes popular (Stuxnet targeted a popular PLC device) or exists in the market longer, it can become more susceptible to attack. Many devices in the past were not designed to be field programmable or to accept updates without significant modifications. Those days are gone. Devices today must be field upgradeable not only to change and improve functionality but to resolve future security issues.

Including security planning in the life cycle management of your device is critical. Moreover, it is important that organizations respond rapidly to security vulnerabilities as they arise. Equally important is that your COTS suppliers match your security responsiveness.

At a minimum, embedded systems designers and developers must adopt the following product life cycle design aspects:

- **Integrate security into the entire product life cycle:** Security must be designed into embedded products and systems from the very beginning and be a foundation of the entire system life cycle. A secure architecture cannot be bolted on at some future date.
- **Architect for change and keeping your product current:** Devices and systems require the ability to be updated, patched, and modified over time. It is critical to design continuous security monitoring and updating into product support plans.
- **Design and test for security:** Security vulnerabilities are a class of software requirement deficiencies—in design or implementation—and the earlier they are caught in the product development life cycle, the less costly it is to fix them and harden a system against attack. Security testing must involve defining the boundaries of a system and determining methods of exploiting weak defenses along these boundaries. Techniques such as fuzz testing or penetration testing, which simulate attack vectors used by malicious hackers, can also be effective tools. Management and automation of security testing, as well as use of simulation tools, can greatly increase embedded development productivity and allows for more comprehensive examination of products than ad hoc testing.
- **Assign a high priority to security vulnerabilities and defects:** Security needs to be considered a high priority not only in design but also during support and maintenance. Vulnerabilities, when discovered by the security community, become public knowledge quickly. Companies need to respond and correct their products with agility as these unplanned vulnerabilities arise.
- **Create a security response team:** This team is needed to address vulnerabilities, draft responses, communicate internally and externally, plan for potential product updates, and manage the delivery of those changes. A security response team is usually cross-functional, for example, including software and hardware development, software quality assurance, customer support, product management, and technical publications.

CONCLUSION

With these five steps, companies can make fundamental progress toward minimizing their risk and exposure to security threats with their highly connected embedded products. Designing security into embedded systems today is a core necessity that requires an increased commitment by all aspects and levels of a company to achieve the end-to-end high assurance coverage demanded in our increasingly connected world.

Wind River is a global leader in delivering software for the Internet of Things. The company's technology is found in more than 2 billion devices, backed by world-class professional services and customer support. Wind River delivers the software and expertise that enable the innovation and deployment of safe, secure, and reliable intelligent systems.