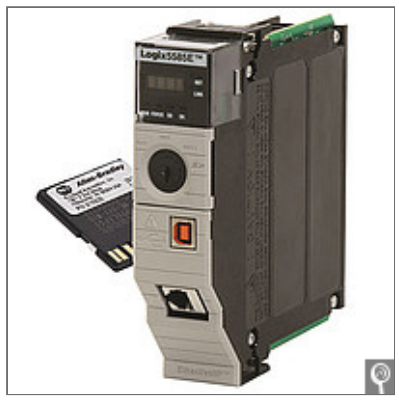
Safety, security, and source code for industrial embedded systems: No shortcuts

BRANDON LEWIS, TECHNOLOGY EDITOR

1 Comment - Leave a Comment



For English-speaking embedded engineers the term “safety,” as in functional safety, is used to imply the reliable and deterministic operation of electronic systems, particularly those capable of harming human beings. The term security, on the other hand, is used to describe safeguards present in an electronic system to protect it and its data against software-borne, network-borne, or physical vulnerabilities.

For German-speaking embedded engineers the word “sicherheit” is used to describe both. While the lack of distinction can be confusing at times, as more embedded devices connect to the Internet of Things (IoT), the two are becoming increasingly intertwined: at their root, both focus on the integrity of the code.

While security industry professionals tend to be fixated on encryption algorithms and the hypothetical possibilities of side-channel attacks, what engineers in the embedded world have known for some time is that bad code is much more likely to expose a system. To this end, coding standards and practices such as MISRA C and CERT C that were developed to help engineers achieve functional safety requirements can also be beneficial to the overall security of connected systems, as Dave Hughes, CEO of HCC Embedded explains.

“There are practically no examples of algorithms busted in a way that can be used en masse,” says Hughes. “The only real breaks in the security of networked devices come from two sources. The first source is when somebody’s gained access to the password database, as in the Sony disaster. The other type of case is when there is a fundamental error in the source code, which allows the hacker to access lots and lots of different devices from a third location en masse.

“The errors that have occurred due to bad coding mistakes, and I guess Heartbleed was the biggest of those, are potentially catastrophic and can allow somebody remotely to access millions of devices, millions of people’s data, and that data can be of varying quality,” Hughes continues. “Carnegie Melon developed the CERT C standard specifically targeted at writing secure C code, and estimated that 90 percent of hacks were the result of poor coding. If you include the actual number of people or devices hacked, it would be a much larger number than 90 percent. CERT C targets people trying to write code properly. So by far the biggest factor in trying to secure a device is to write good quality code.”

HCC Embedded provides verifiable software components such as networking stacks, file systems, and bootloaders developed using the MISRA C:2004 standard, along with their artifacts, for safety- and security-critical industrial systems. However, Hughes cautions that the endeavor of creating quality code is far greater than simply using coding standards. It is the result of sound development methodologies that encompass verification, validation, and specification of both system requirements and the development process itself, driven by qualified individuals from product management down through systems and application engineering.

This, of course, is a challenge for industrial systems, the design of which is often based on a multi-vendor development supply chain. The issue of managing a development methodology across multiple stakeholders is where traditional notions of security can be applied in the context of protecting source and execution code, according to Marcellus Buchheit, President and CEO of Wibu-Systems USA Inc.

“In the embedded world we have a system integrator that installs an embedded system, and knows what libraries and applications need to be added. The operator or user never touches this,” says Buchheit. “If someone unfamiliar installs something it’s a problem because it probably endangers the stability of the whole system.

“When you have a complex system, like a roller coaster, and have 50 PLCs or more running in the system, a company like Rockwell just sells the PLCs. Then you have another company that installs roller coasters, who know these PLCs very well,” Buchheit continues. “But now you have the issue that a maintenance person comes and has to change something. The maintenance person knows well what to do, but the roller coaster manufacturer still wants to know what has changed when the maintenance person modifies the PLC parameters. The maintenance company needs limited access to the roller coaster, but only for a limited amount of time.

The roller coaster example illustrates not only the multi-vendor ecosystem associated with many industrial systems, but also the need to protect valuable intellectual property (IP) within source code and the content of the source code itself through cryptographic security measures that lock down a system.

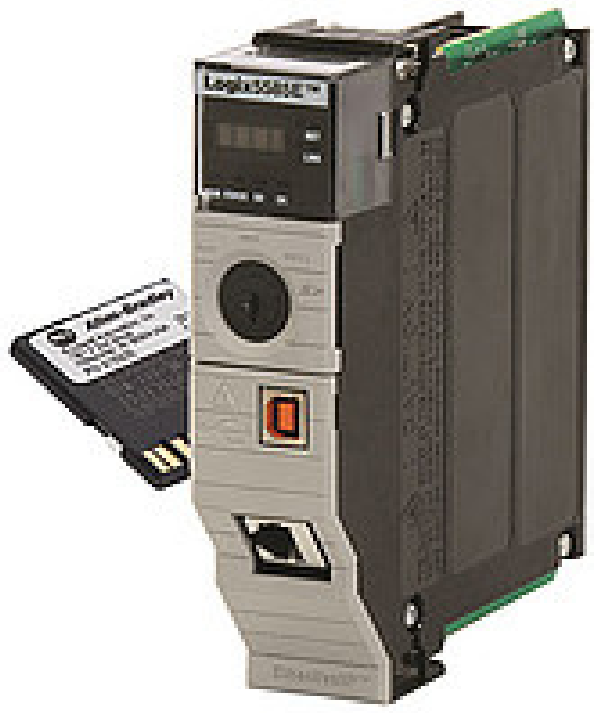
“On these systems you definitely need code that is secure, but also code that when updated can not be infected or altered to perform malicious attacks,” Buchheit says. “On the one side is IP protection, because the roller coaster manufacturer doesn’t want someone else to analyze the PLCs and build the same roller coaster for a lower price because they know what’s running on the PLCs.

“The other thing is security and safety,” he says. “Stuxnet affected this binary code so that while it was transferred from the memory card in the PLC it was modified by this virus, and what you saw was that the source code that was running later was something different than what had been running in the PLC previously, and attacked these centrifuges and destroyed them.”

Wibu-Systems’ solution for providing IP protection alongside safety and security for industrial systems (connected or not) can be found in the Rockwell Software Studio 5000 Logix Designer application, which is based on tamper-proof CodeMeter secure elements and software licensing technology. It generates software licenses that use cryptographic keys to encrypt and decrypt source code and executables, which allows industrial manufacturers to control what code can be accessed, who can accesss it, at what time, and for how long.

“The Rockwell Software Studio 5000 is a source code protector. The system converts binary source code and uploads it to a PLC,” Buchheit explains. “To access that PLC source code, a maintenance person would need a key (or specific license code) that is valid for one time use or maybe a week, attach it, view the source, make modifications, and then after that the key cannot be used again. The manufacturer can not only control when changes can be made, but they can also control what source code can be modified and what can’t, what rights the person has to do something, so each service person could have different rights.

“The Rockwell Software Studio 5000 system encrypts the data into this system, and with the same license key is able to decrypt it. A virus like Stuxnet would be unable to address the PLCs binary code, so it would not be able to repeat the Stuxnet attack. It’s authenticated code between the development system and the execution system in the PLC,” he adds.



[Figure 1 | The Rockwell Software Studio 5000 includes source code protection, execution (application code), and web-based entitlement (licensing) to safeguard IP and defend against malicious attacks. Shown here is an Allen-Bradely ControlLogix 5580 controller from Rockwell Automation with an SD card running Rockwell Software Studio 5000 Logix Designer.]

Industrial cyber security: Investment required

Inevitably some IoT devices in the consumer sector will lack some of the fundamental coding practices and security measures described here, largely due to tight margins and short deployment lifecycles. However, for safety-critical industrial systems, such measures are now a requirement.

“Historically we’ve seen lots of companies making devices with software components that would quite shock you, simply because you’d expect them to invest much more heavily in verifying those elements,” says Hughes. “But for reasons that I don’t understand, they haven’t had to do that. We would like them to invest in developing verifiably strong software components because, as engineers, we like the idea of producing fault-free software. But we can’t do that unless companies are willing to invest in software developed within robust and standardized frameworks (IEC 61508, DO178B etc.) This is a significant upfront investment in the product – but the results are proven across many industries – particularly aerospace and industrial safety where the regulations have been very strict for years.”

“There are no shortcuts. If you take a shortcut, your system will be weaker,” he says. “It’s a game of probabilities.”

THIS ARTICLE WAS PUBLISHED ON MAY 4th, 2017.