Comment

Impact of cybersecurity regulations on embedded networks

Worldwide, regulatory bodies publish rules and regulations concerning the cybersecurity of electronic devices. Some of the most stringent are issued by the EU, such as the Cybersecurity Act (CSA) and the Cyber Resilience Act (CRA), which all manufacturers of electronic embedded network devices must follow.

According to the issued regulations, the manufacturers of electronic devices using embedded networks or fieldbus networks have the following main options:

1. Apply "security by design": Ensure that all company's products, processes, and personnel adhere to this principle. This approach simplifies the risk assessment because everything required is already in place, and one only needs to document it.
2. Selective compliance: Identify niches where certain security rules do not apply to a specific product. This method results in lengthy risk assessments as each exception must be justified by a security expert.

Option 1 may seem more costly and effort-intensive, as it requires adaptations to all processes, including development and maintenance procedures. To note is that cybersecurity is not free; the CRA estimates that compliance will cost EU companies around 29 billion Euros. However, with Option 2, one might need to re-evaluate the situation for every product, release, and market continually. This could result in multiple software versions and different risk assessments for different markets.

While some manufacturers might temporarily avoid implementing a "security by design" scheme, most will eventually need to adopt it to sell products across various markets and application use cases. This is especially true, if a company is a supplier of individual components with a CAN or other network interface. A product could be a hard sell if you would need to state: "Only use this product in isolated networks with no access points to other networks."

Looking at the CRA, there are various consequences for business processes, such as security incident reporting, but this document will focus on the main technical aspects. In summary, "security by design" means building layers of security; protecting a single aspect or communication method does not qualify as secure by design.

Applying "security by design"

Here is a list of the major technical consequences of applying "security by design" to the development of CAN-based devices and systems. This is just a summary, and each point may involve a whole set of further decisions and consequences.

1. ***System design, hardware (HW), and software (SW) development***
   - Requires full version tracking and (peer) review: Document who made what changes, when, and why.
   - No single, rogue employee can introduce a vulnerability.
   - All HW components and SW libraries should be "security hardened." Document any security limitations.
2. ***System network layout and design***
   - Limit physical exposure of network cabling. This is the only chance to protect the system from the ultimate denial of service attack: cutting wires.
   - Define and review all gateways and bridges to different networks. Practice network separation and implement network zones using firewalls.
   - Plan for foreseeable changes: anticipate additional gateways or access points being added later.

3. ***Key management and infrastructure***
   - Introduce a key infrastructure, either a public key infrastructure (PKI) or a pre-shared keys database.
   - Use multiple keys for different purposes or layers.
   - Some keys can be local to a machine, while others are needed externally for maintenance or diagnostics.
4. ***Secure software/firmware updates and version management***
   - Carefully select update paths and methods.
   - Only allow authenticated software/firmware to be installed.
   - Fully document the history of software updates applied to each device.
5. ***Secure data storage***
   - Protect stored configurations or data in non-volatile or external memory with both encryption and authentication.
6. ***Secure communication using non-cryptographic methods***
   - Implement injection detection where possible.
   - Monitor communication behavior, especially for anomalies in timing in strictly time-based systems.
7. ***Secure single channel communication between two entities (1:1)***
   - Use protocols like datagram transport layer security (DTLS), transport layer security with pre-shared key (TLS-PSK), or optimized variations.
8. ***Secure group communication between multiple entities (N:M)***
   - Apply appropriate security measures for group communication.
9. ***Secure higher-layer protocol communication***
   - Authenticate powerful network management commands and essential system configurations.
10. ***Application-level security measures***
    - Consider implementing additional security measures at the application level.
    - Log which user installed which system configurations.

The CiA SIG HLP cybersecurity (special interest group for higher-layer protocol cybersecurity) focuses on addressing all security measures above the data link layer. The goal of this group is to tackle most of the points listed above and to publish documents detailing best practices for handling these security challenges.

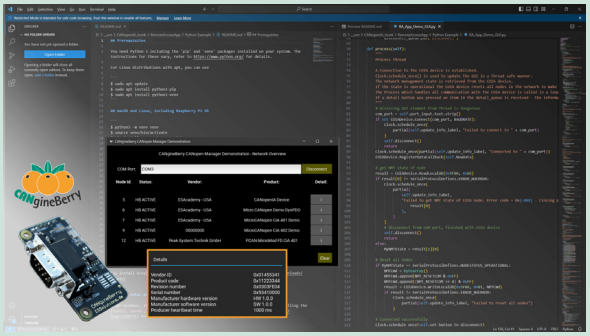
The SIG aims to develop guidelines and protocols for implementing security measures for higher-layer protocols in embedded networks. The group also provides guidance and frameworks to help manufacturers comply with regulations such as the Cybersecurity Act (CSA) and the Cyber Resilience Act (CRA).

If you are interested in joining the effort, consider joining the SIG HLP cybersecurity. Contact secretary@can-cia.org to join. ◀

Figure 1: Use multiple keys for different purposes or layers (Source: EmSA)(Source: EmSA)

Firmware update for CANgineberry

Embedded Systems Academy (EmSA) updated its CANopen Manager software for the CANgineberry. The included Python example is aimed at simplifying network management for developers working across Linux, Windows, and macOS. This example uses the provided CANopen Manager firmware to visualize and manage devices on a CANopen network.



Python-based graphical user interface screen (Source: EmSA)

Unlike basic CAN interfaces (only passing through CAN frames), the CANgineberry handles the entire CANopen NMT (network management) manager functionality. This allows the module to automatically scan the network, identify new devices, and configure the appropriate process data objects (PDOs) without requiring manual intervention. Through this automated functionality, users can directly access the object dictionary entries for both the CANopen NMT manager and the connected CANopen devices, reducing the need for writing a complex code.

The Python example showcases these features through a graphical user interface (GUI). Once connected to a CANgineberry, the script accesses detected CANopen devices, retrieves their details on demand, and presents them in an intuitive display. The Python-based GUI enables the configuration and monitoring of CANopen networks and gives instant control over CANopen devices, informs the manufacturer. The Python example is available for download at: https://cangineberry.com. *of*

Author

Olaf Pfeiffer
EmSA (Embedded Systems Academy)
info@esacademy.com
www.esacademy.de