Security architecture for CAN

Partly financed by European and Belgian governmental research projects, the Sancus approach is an ongoing initiative to develop a security solution for CAN-based embedded control systems.

Today, networked electronic control units (ECU) manage road vehicles. They interpret the sensor readings and operate the actuators to control the car's behavior and safety. They intervene for braking, steering, light switching, actuate airbags, and optimize the powertrain operation. The crux is: all these networks are connected and open to the outside world, which renders them vulnerable to malicious interferences. To establish an effective mitigation against such attacks, Imec has devised the Sancus security architecture for networked embedded devices.

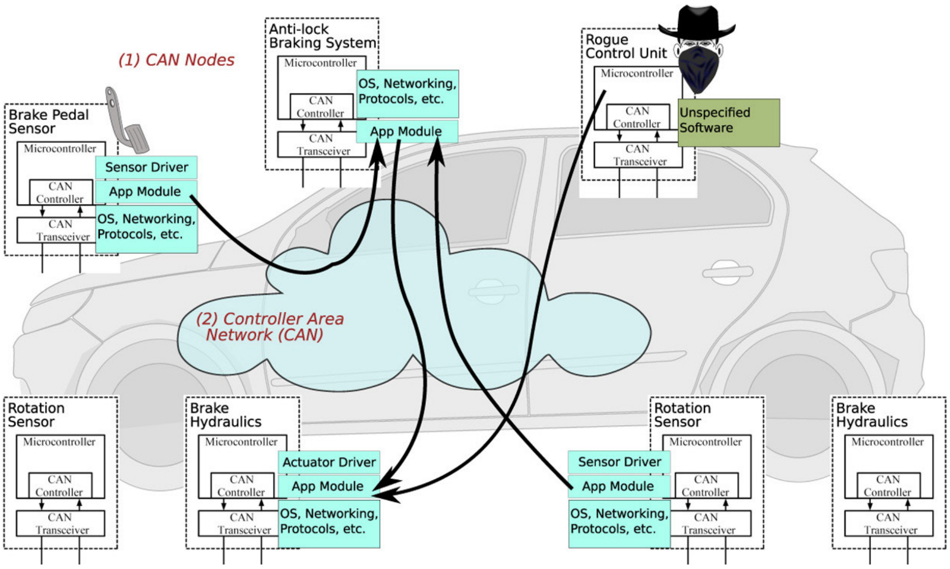

Figure 1: Networked control system with multiple ECUs and gateway with no protection against message injection or replay attacks (Photo: Imec)

Sancus has been designed and implemented by Imec's Distrinet and Cosic (both located at KU Leuven) – two research groups well known for their work on security matters. The development is supported partly by the Intel Lab's University Research Office. It was also partially funded by the Research Fund KU Leuven, by the European FP7 Project Nessos, and by the Belgian Cybercrime Centre of Excellence (B-CCentre).

Sancus was carefully laid out to fit the usual automotive electronics environments, and is intended as a general solution to secure not just vehicles, be they smart or autonomous, but also for other critical infrastructures, such as medical equipment, smart buildings, or power grids. To ensure that the results of the Sancus initiative can be verified and reproduced, the hardware design and software of our prototype have been made publicly available. Hardware designs, source files, as well as binary packages and documentations can be found at https://distrinet.cs.kuleuven.be/software/sancus.

CAN: island of smart electronics

Complex industrial equipment is monitored and steered by networks of sensors, actuators, and control processors that continuously exchange essential up-to-date messages. In automobiles, this real-time interaction usually is organized via CAN (Controller Area Network). The problem here is: CAN was laid out about 30 years ago as a closed network with no consideration of obvious access points for intruders. The CAN lower-layers offer a convenient way to integrate the growing number of heterogeneous sensors and control processors, which send and receive reliable and timely messages without any central computer. Most important: CAN networks connect the rotation sensors in the wheels with the anti-lock braking system (ABS) and the drivetrain.

Traffic infrastructure: opening up to the world

In high-end cars, the infotainment and navigation systems are hooked up to both, the CAN network and to external public networks. The infotainment equipment communicates via the driver's mobile phone or headset and they receive software updates from their vendors. With information provided by the CAN network, it is possible to turn up the music volume when driving faster or upon entering rough terrain. Autonomous vehicles take this a step further: they will communicate with the traffic infrastructure to steer and protect the car.

So, suddenly a car's CAN network provides a number of potential entry points for malicious intruders. Communication with the outside is done via Bluetooth or IP networks, some of which may connect to the Internet. And the Internet, if anything, is a highly non-trusted network. The CAN interface and it's hardware and software components were not designed to operate in such an unsafe environment. CAN ▷

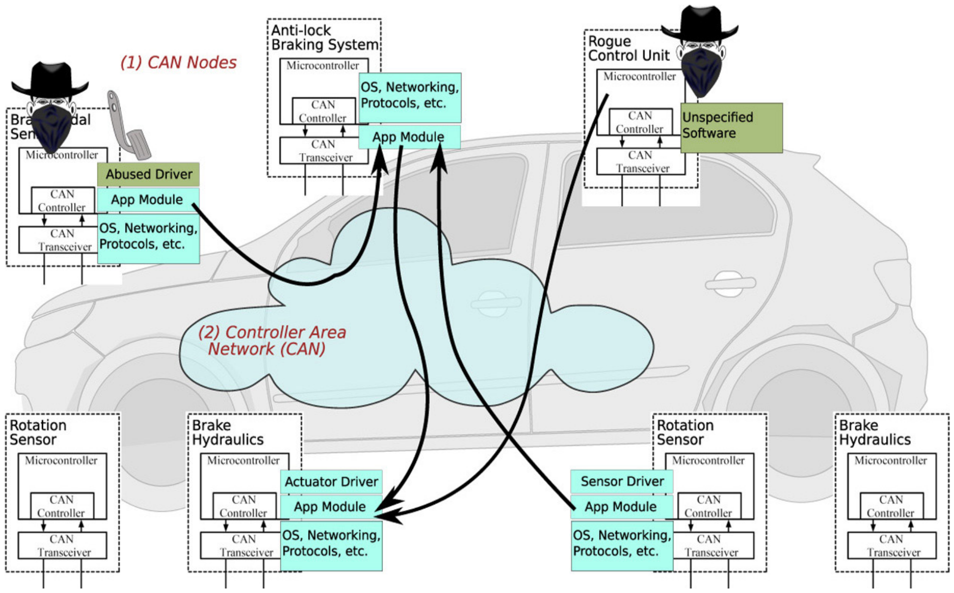

Figure 2: Classical CAN allows for two main attack scenarios. Firstly, in the absence of strong authentication mechanisms, a rogue ECU can inject message. Secondly, the lack of security mechanisms on many light-weight ECUs leverages software vulnerabilities as attack vectors. Even in the presence of authentication and encryption, attackers are able to control software directly, bypassing securing mechanisms (Photo: Imec)

offers no actual form of authentication or authorization. If a syntactically correct CAN message arrives at the car's brake system, it just assumes that the message is legitimate and stems from a trusted source.

Moreover, car network processors are designed to be very small and inexpensive, just good enough for their task, and consuming as little power as possible. They usually run tiny operating systems and some communication and control applications. They don't feature memory protection or an isolated sandbox to run processes in. Every application, also an application that shouldn't be there, is able to access and rewrite the complete processor memory.

All in all, this is a considerable risk and – an untenable situation. Reportedly, researchers were able to remotely control a car by hacking its Wifi or Bluetooth gateways. Also, in a high-stakes case in Ukraine, it was demonstrated that electricity grids could be taken over and manipulated by attackers. Imec's researchers were able to hack pacemakers, eavesdropping on the devices and injecting potentially harmful commands.

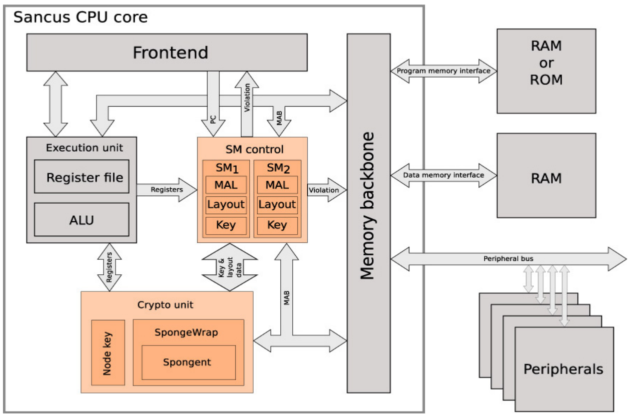

Figure 3: Block diagram of the Sancus CPU core (Photo: Imec)

This is not to say that such attacks are easy: They require a high level of sophistication, ingenuity, and patience. But in the case of highly sensitive road traffic environments, because of the sheer number of electronically identical cars involved, an attacker who manages to find a way into one system, poses a real threat to the security of a great number of other systems.

Establishing safe processing harbors

For all these incidents and scenarios there is no commercial mitigation available today. In contrast to higher-end processors in laptop computers and smartphones, the automotive control chips are small and resource-constrained. They lack the security features that are standard on other processors, such as various privilege levels and memory segmentations. Yet, replacing all embedded processors in cars with high-end systems is not an option, due to cost, complexity and power consumption.

Therefore, at Imec, we have initiated a research endeavor to design a new secure architecture that is suitable to secure today's embedded systems. It covers the CAN networks in cars, and also industrial control systems in manufacturing, or even very small IoT (Internet of Things) devices. Such security systems have to be low on complexity and cost – a definite requirement in regard to the envisioned applications.

We started out with a lightweight micro-controller and extended its design, adding a secure memory management and a crypto unit optimized for low-power consumption. The result is a processor that is not much larger and doesn't consume much more energy (about six percent). But it is able to isolate the critical network software and it creates a kind of a safe harbor for it. With this isolation concept, the software cannot be compromised. Its trusted computing base is restricted to the hardware on which it runs. Barring vulnerabilities in a protected application itself, no software, be it an application or operating system, running on the same processor or on an outside process, can override the security checks and read or overwrite the protected runtime state.

Knowing whom to trust

But even if the processor that controls the brakes of a car can no longer be hacked, it will still obey any brake command, even if issued by an illegitimate source. Therefore, we have limited the range of trusted message sources to those that can authenticate themselves as legitimate. Thus a brake command should only come from a trusted processor, which itself cannot be hacked, and from an authenticated software component. So the CAN network is now made up of small unbreakable applications that mutually authenticates and trusts each other.

In an automobile, such an embedded system must be able to be contacted from the outside, for instance by a software provider that wants to install updates, or, in a more general way, for communicating with the surrounding traffic infrastructure. Therefore, Sancus provides secure communication and remote attestation. Any outside party can send or receive messages to and from a specific software module on a specific node, while making sure that this is the correct module (authenticity), it has not been changed (integrity), and its status is correct (freshness).

Demo at ITF and future work

In May, we have demonstrated Sancus at our Technology Forum in Antwerp either in an automotive scenario and as a smart metering solution. Sancus is conceived as novel security architecture for resource-constrained, extensible embedded network systems. It provides remote attestation and ▷

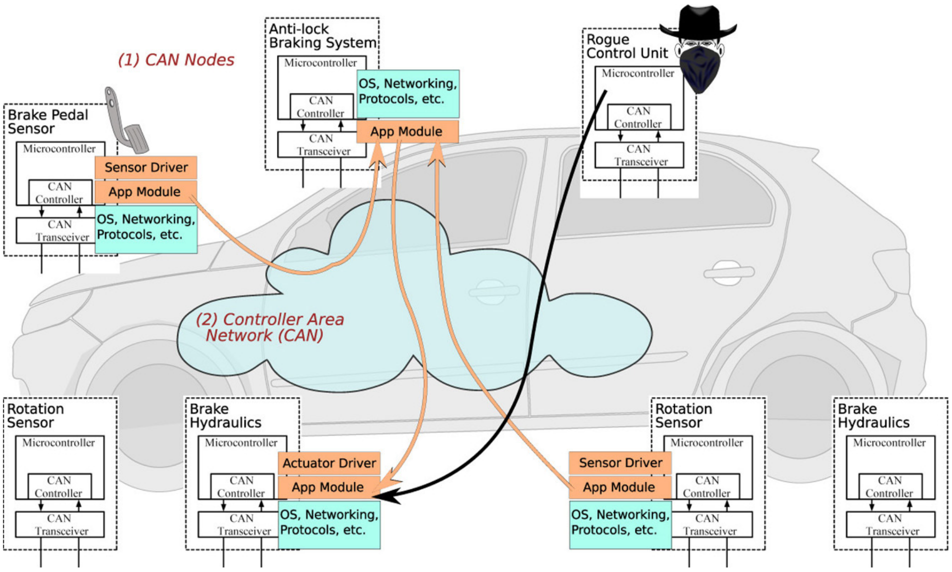

Figure 4: Legitimate nodes of the network run secure applications with Sancus' protection (orange), which matually authenticate each other and are protected against code-abuse attacks; rugue nodes cannot interfere with security (but may harm availability), node takeover is very diffecult, if not imposible (Photo: Imec)

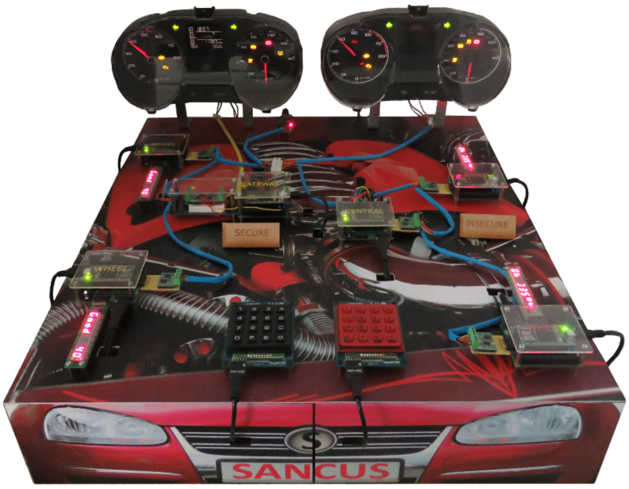

Figure 5: Sancus prototype implementation (Source: Imec)

strong integrity, as well as authenticity guarantees within a minimal layout of a trusted computing base. Sancus consists of the specially extended microprocessor, the dedicated software running in the safe harbors, and a C compiler that generates the Sancus-secured code.

To be precise, Sancus still is an ongoing project, and the researchers need still resolve a number of issues to be included in Sancus. One of these issues is to ensure the availability and real-time function of the network. We can now guarantee that any messages that arrive in a module are legitimate. But we cannot yet ensure that they will arrive at their intended destination nodes. It would still be possible for an attacker to drop malicious messages – which our solution of course would detect. And in most cases this would probably not lead to dangerous situations, as the receiving node would raise an error flag and halt the system in a safe way. But this is of course inconvenient.

A second issue is safe operation of the secure software modules. Without formal design methodologies and inherently safe programming languages, these modules show vulnerabilities that may lead to unsafe operating situations. But due to the small isolated modules of trusted code, it should be possible to design these in a more formal, fault-free way. The researchers are still looking for collaboration partners to develop suitable hardware/software solutions. ◀

Author

Jan Tobias Muehlberg
Imec
jantobias.muehlberg@kuleuven.be
distrinet.cs.kuleuven.be/software/sancus