CHALMERS
UNIVERSITY OF TECHNOLOGY

Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms

Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms

Thomas Rosenstatter
Department of Computer Science and Engineering
Chalmers University of Technology
Email: thomas.rosenstatter@chalmers.se

Tomas Olovsson
Department of Computer Science and Engineering
Chalmers University of Technology
Email: tomas.olovsson@chalmers.se

***Abstract*—Modern vehicles are becoming targets and need to be secured throughout their lifetime. There exist several risk assessment models which can be used to derive security levels that describe to what extent components, functions and messages (signals), need to be protected. These models provide methods to gather application specific security requirements based on identified threat and item combinations that need to be coped with. However, a standardized mapping between security levels and required mandatory security mechanisms and design rules is currently missing. We address this problem first by suggesting that the risk assessment process should result in five security levels, similar to the functional safety standard ISO 26262. Second, we identify suitable security mechanisms and design rules for automotive system design and associate them with appropriate security levels. Our proposed methodology is as much as possible aligned with ISO 26262 and we believe that it should therefore be realistic to deploy in existing organizations.**

I. Introduction

Computer and network security became more and more important as the number of interconnected devices spread. We now face similar challenges in the ongoing transition towards the Internet of things, industry 4.0 and smart connected vehicles. Constraints, such as low computational power, low energy consumption and real time reaction, are major challenges that limit the range of applicable security mechanisms and design rules. Back in 2011, Checkoway et al. provided a detailed analysis of the attack surface of vehicles including attacks via the Tire Pressure Monitoring System and the media player [1]. Furthermore, Miller and Valasek have demonstrated several vulnerabilities that lead to attacks against vehicles that could be performed remotely in 2015 [2].

There is currently no standard similar to the functional safety standard ISO 26262 [3] for security in the automotive domain. The ISO work item ISO/SAE AWI 21434 *Road Vehicles – Cybersecurity engineering* is currently under development and is planned to address threat modeling and risk assessment. Proposed security models, such as Microsoft's STRIDE threat model [4], SAE J3061 [5] and the HEAVENS model [6], describe how to identify items that need to be secured, perform the threat analysis and result in security levels for components and functions in a vehicle. HEAVENS additionally describes a method for deriving item specific requirements. Currently missing in proposed models is how to select required security mechanisms for each component and function based on the security level when following the risk assessment process. In this paper, we address this problem by first suggesting that the risk assessment process should result in *five* security levels, similar to the Automotive Safety Integrity Levels (ASILs) and, when needed, a set of specific security requirements for that particular function. Fig. 1 provides an overview and puts our work into context.

We believe the proposed methodology can show the way towards a possible future automotive security standard similar to ISO 26262 used in automotive safety. The proposed methodology is realistic to deploy in existing organizations, as it is aligned with safety design methodologies which are already in place. In addition, having distinct requirements for security design allows the classification of components, including third-party designs, and enables us to know how robust and secure the design is and to what extend it can be trusted by other components.

We emphasize that we are focusing on mechanisms to be deployed in vehicles rather than traditional information systems, which is covered by the ISO 27000 family [7]. The contributions of this paper are as follows:

- We propose a set of suitable mandatory security mechanisms and design rules for the automotive domain.
- We suggest a framework for mandatory security mechanisms that should be required for specific security levels.
- We motivate the proposed method with an automotive use case.

II. Background and Related Work

The Cybersecurity Guidebook for Cyber-Physical Vehicle Systems, SAE J3061 [5], provides guidance for threat identification and assessment, and guidance for security in the development process. HEAVENS [6] and EVITA [8] are two Threat Analysis and Risk Assessment (TARA) models also referred to in SAE J3061, which result in security levels and methods for how to specify security requirements for identified threats. Later in this paper, we apply the HEAVENS model to our automotive use case, to identify security relevant

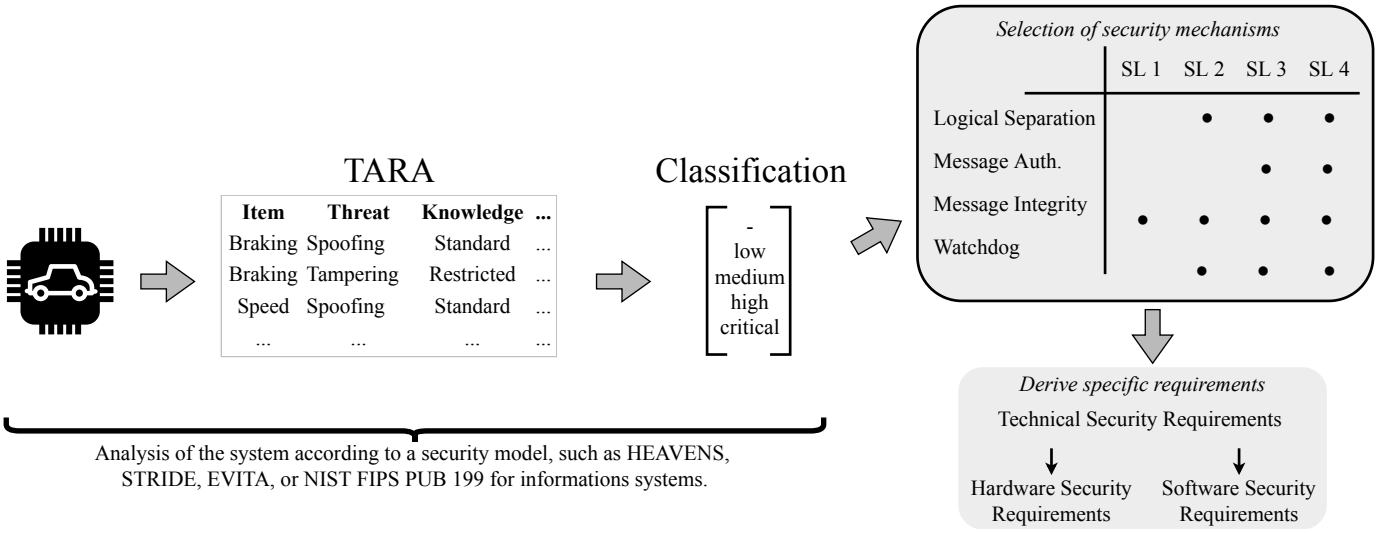


Fig. 1: Overview of the steps from performing Threat Analysis and Risk Assessment (TARA) to requirement engineering.

items, i.e., functions, components and messages (signals), and classify their security needs. The resulting security levels constitute of six elements representing the security attributes mitigating the STRIDE threats: *integrity*, *authenticity*, *non-repudiation*, *confidentiality*, *availability*, and *authorization*.

IEC 62443 [9], a security standard for industrial communication networks, performs a similar mapping between system requirements and security levels. IEC 62443 is not entirely applicable for the automotive domain due to the focus on the human operation of industrial automation systems.

The NIST FIPS PUB 199 [10], *Standards for Security Categorization of Federal Information and Information Systems*, describes a security classification for information systems using *confidentiality*, *integrity*, and *availability* with each having three levels, low, moderate and high. Additionally, the NIST special publication SP 800-53 [11], *Security and Privacy Controls for Federal Information Systems and Organizations*, contains security mechanisms for information systems. The work in the *Connected Vehicles Pilot Development Phase 1 - Security Management Operating Concept – New York City* [12] follows both, the NIST FIPS PUB 199 and NIST SP 800-53, and identifies safety and privacy requirements for specific usage scenarios and further divides these requirements in information flow and device classes. As many of the NIST SP 800-53 security controls are not appropriate for the automotive domain, the identified security requirements in this project do not cover the requirements for the automotive domain in depth.

The United Nations Economic Commission for Europe (UNECE) has started a task force on cybersecurity and over-the-air issues in [13]. The version from 27/11/2017 consists of a reference architecture, a list of security principles and security controls to mitigate certain threats. We have taken security mechanisms and requirements from all these works into account when selecting relevant mechanisms applicable for the automotive domain.

III. Security Mechanisms and Design Rules

Security mechanisms are used to mitigate threats and to minimize the risk of security attributes of an item being violated. Table I shows the mapping between the STRIDE threats and the corresponding security attributes, including an explanation from HEAVENS [6, p.6]. In Table I we additionally associate the STRIDE threats with item types to emphasize the threats violating the security attributes related to these types. For example, mechanisms that mitigate all types of threats are necessary in order to properly protect messages or signals – the information flow.

The benefits of having a direct relation between STRIDE and security attributes is that any threat assessment model can be used to derive required mandatory security requirements provided a mapping to STRIDE exists. For example, a mapping of the CIA model in [10] to STRIDE is also possible, however, the desired granularity decreases due to the aggregation of security attributes. A proposed mapping between security levels and identified mechanisms is provided in Table II. For some identified design rules and mechanisms, it is favorable to divide them into different classes represented as numbers instead of dots in the table describing the requirements in more detail. For instance, the requirement *AU.3 Verify authenticity of firmware/functions on start* has class 1 *on demand verification of modules* and a stricter class 2 *secure boot*. Furthermore, we highlight that these security mechanisms are mandatory, nevertheless, a mechanism may be disregarded when properly argued.

The following sections describe the six security attributes and list the corresponding security mechanisms and design rules. The security levels the mechanisms are associated with range from 0 to 4, where level 0 demands no security and level 1 and upwards require increased security needs.

TABLE I: Mapping of STRIDE threats to security attributes and types of items that need to be protected: information flow, message (msg), firmware (fw), and hardware (hw)

STRIDE Threat (from [4])	Security Attribute* (from [5], [6])	Explanation (from [6])	Item Type
Spoofing	Authenticity	Attackers pretend to be someone or something else	MSG, FW, HW
Tampering	Integrity	Attackers change data in transit or in a data store	MSG, FW, HW
Repudiation	Non-repudiation	Attackers perform actions that cannot be traced back to them	MSG, FW, HW
Information disclosure	Confidentiality	Attackers get access to data (e. g., in transit or in a data store)	MSG, FW
Denial of Service	Availability	Attackers interrupt a system's legitimate operation	MSG, FW
Elevation of privilege	Authorization	Attackers perform actions they are not authorized to perform	MSG, FW, HW

* Unlike SAE J3061 [5], we include the attribute freshness in non-repudiation and omit privacy, as we believe that privacy needs to be addressed separately.

A. Integrity

Integrity is a property that ensures that data, like messages and firmware, have not been altered due to random errors during transmission or by a malicious node. Message Authentication Codes (MACs) can be used to verify the integrity of data. We propose the use of MACs with securely stored pre-shared keys as the required minimum to provide message integrity. Verifying the integrity of software during boot or when upgrading also requires the use of cryptographic hashes along with secret keys. In order to decrease the time overhead the verification of the firmware takes, the firmware may be partitioned into modules and the verification of the integrity may only be performed for modules currently needed.

IN.1 Message Authentication Code (MAC) with pre-shared key.
IN.2 Cryptographic hash function with pre-shared key to verify *firmware* integrity when upgrading.
IN.3 Cryptographic hash function with pre-shared key to verify *firmware/function* integrity on boot.
IN.4 Physical protection against tampering.
IN.5 Detection of physical tampering.

B. Authenticity

Authenticity guarantees the origin of data, thus mitigates the possibility to spoof data, i.e., messages and firmware. MACs combined with authentication of nodes and session keys can be used to provide authenticity and integrity of messages. Authenticity of firmware received via over-the-air updates or through a diagnostics device may be achieved with digital signatures and physically unclonable functions may be used to ensure the authenticity of hardware the Electronic Control Unit (ECU) communicates with.

AU.1 Message Authentication Codes (MACs) with session keys to provide message integrity and authenticity.
AU.2 Verify authenticity of firmware when upgrading using digital signatures.
AU.3 Verify authenticity of firmware/functions on boot using digital signatures.
AU.4 Verify authenticity of hardware, e. g., ECUs and diagnostics devices.

C. Non-repudiation

In order to cope with repudiation of messages, it is necessary to include counters or timestamps in messages to be authenticated to guarantee freshness. Moreover, it is needed to store logs as evidence of performed transactions and the use of digital signatures to provide proof about the source of the message.

NR.1 Freshness mechanism, e. g., adding a counter or timestamp to message to be authenticated.
NR.2 Audit logging.
NR.3 Use of digital signatures for messages (signals).

D. Confidentiality

Encryption provides confidentiality of data. Some microcontrollers can include embedded Hardware Security Modules (HSMs) to provide hardware accelerated encryption and a secure storage of cryptographic keys. Hardware acceleration can be crucial when transmitting and receiving time-critical authenticated and/or encrypted messages. In case decryption is only required for firmware updates, it might be sufficient to rely on software-based methods. Some ECUs might also not have the resources to decrypt the firmware and to perform the update. In such cases it has to be decided whether it is sufficient that the gateway ECU decrypts and verifies the firmware update.

CO.1 Encryption of *messages* when transmitted over the network.
CO.2 Encryption of *firmware* when transmitted over the network.

E. Availability

Denial-of-Service (DoS) attacks can target the network and computational resources of an ECU. The effectiveness of DoS

attacks aiming at the network strongly depend on the network architecture. The Controller Area Network (CAN) bus, for instance, is due to its nature highly vulnerable to DoS attacks (see [1]), whereas Flexray (ISO 17458-1:2013) is based on time-triggered control for accessing the physical medium. As example, this time-triggered control approach already limits the extent of the attack to the time slots the compromised node is allowed to send. Other targets of DoS attacks can be memory and I/O resources. Watchdog timers can protect against some DoS attacks targeting the computational resources.

AV.1 Limiting access to network resources - Quality of Service (QoS).
AV.2 Use of watchdog timers.

F. Authorization and Access Control

Authorization to perform certain tasks or to send specific messages can be enforced individually or by introducing roles on many levels, e. g., network, host, and function-level. Restricting communication between ECUs can be controlled by separating the ECUs into different domains and connecting these domains via gateways that use whitelists. A more radical approach is complete isolation of a specific network segment, which may be feasible in only a few cases. Host-based access control on the other hand can be used to restrict the types of messages and their frequency an ECU is allowed to send. Examples of logical separation are encryption of network traffic, use of Virtual Local Area Networks (VLANs) and virtualization techniques using hypervisors or containers. Domain isolation on the other hand, requires separated physical networks, total isolation from other functions and processes, dedicated memory, and guaranteed computational resources.

AC.1 Whitelisting of messages (signals) on gateways.
AC.2 Whitelisting of messages (signals) on ECUs.
AC.3 Access control on function level.
AC.4 Detection and logging of intrusions.
AC.5 Logical separation.
AC.6 Domain isolation.

G. Other Requirements

Some security design rules or guidelines have to be fulfilled regardless of the security level due to compliance with laws and regulations or strongly depend on the application rather than a specific security level. Compliance to safe and secure coding guidelines, such as MISRA C Guidelines [14] or SEI CERT C Coding Standard [15], are the base for safe, secure and reliable software. More design rules are for instance least privilege and fail-safe defaults, which have been already listed by Saltzer and Schroeder back in 1975 in [16]. Logging of errors is essential when deploying Intrusion Detection Systems (IDSs), additionally, all violations risen by security mechanisms should be logged. We strongly recommend to take the requirements below into consideration when designing the system.

OR.1 Fail in known state (safe defaults).
OR.2 Input Validation.
OR.3 Operate with least set of privileges that are necessary.
OR.4 Compliance to secure coding guidelines.
OR.5 Secure logging of errors, data modification and updates.

IV. Use Case and Attack Model

The reference architecture for the automotive system we will use in our discussion is shown in Fig. 2 and was developed in the HoliSec project [17]. It illustrates a highly simplified version of ECUs and their functionality where the Vehicle ECU is responsible for the park brake status, warning light and gear requests. The OBD-II port is provided by the Edge Node GW which further interconnects networks to the Infotainment unit, CAN bus ECUs, Ethernet nodes, and the Driver Control unit. The Driver Control unit additionally interconnects sensors, such as radar and cameras, the Vehicle ECU, and the Driver Display.

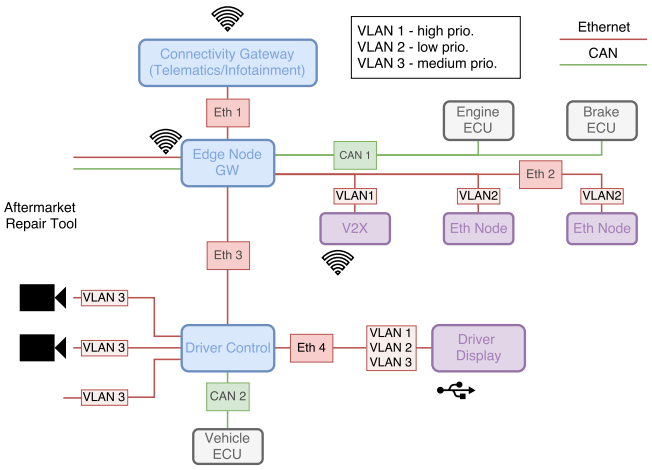


Fig. 2: HoliSec reference architecture of an in-vehicle network.

A. Use Case

Fig. 3a illustrates the functions involved when implementing Cruise Control (CC) functionality and what messages (signals) these functions exchange. Function Cruise Control is located in the Vehicle ECU. It receives target speeds from the driver and sends a *VehicleSpeedCommand* to the function Speed Control which is responsible for maintaining the target speed and therefore also listens to *VehicleSpeed* broadcasts from the Vehicle Speed function. In order to maintain the target speed, Speed Control always broadcasts a *BrakeCommand* on the CAN bus. Function Braking listens to *BrakeCommand* and *VehicleSpeed* to determine the best braking strategy to be applied to the engine and, if needed, also engages the foundational brakes. Function Braking broadcasts *BrakeEngagedPercentage* to inform other functions, such as Light Control, about its actions.

Furthermore, Fig. 3a shows the security levels obtained from a HEAVENS TARA analysis, additionally, we added the security demands on the messages to the source component that is broadcasting information. The results from applying

TABLE II: Mapping between security mechanisms and security levels.

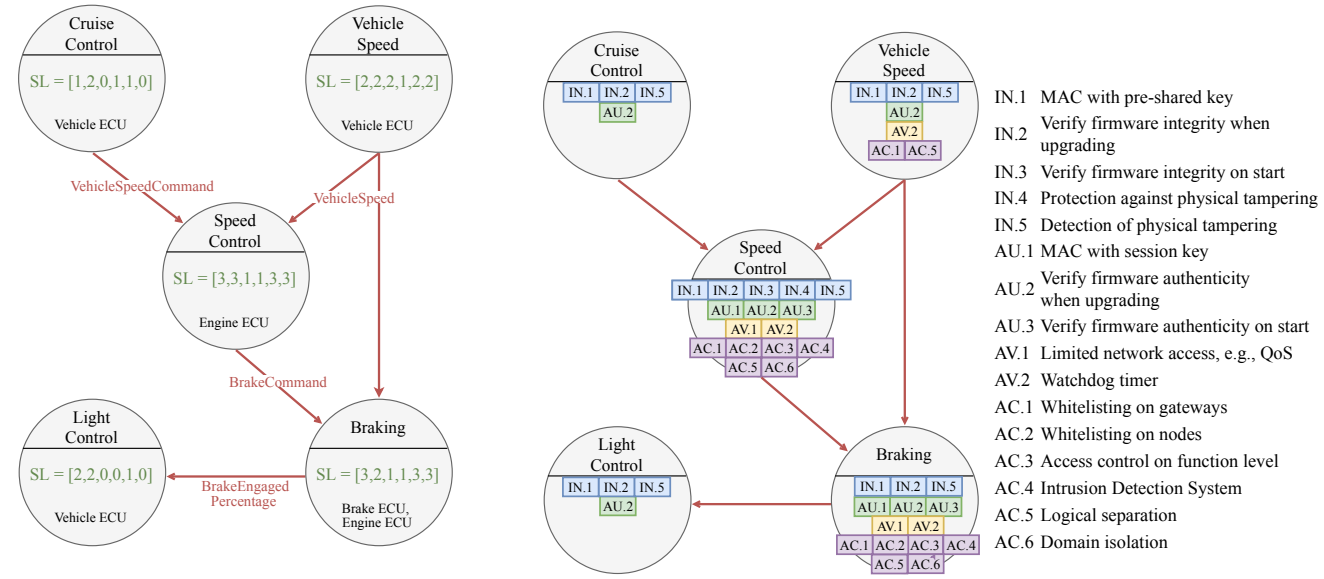
		SL 0	SL 1	SL 2	SL 3	SL 4
Integrity	IN.1 [MSG] Message Authentication Code (MAC) with pre-shared key			•	•	•
	IN.2 [FW] Verify cryptographic hash of firmware when upgrading		•	•	•	•
	IN.3 [FW] Verify cryptographic hash of firmware/functions on boot				•	•
	IN.4 [HW] Physical protection against tampering				•	•
	IN.5 [HW] Detection of physical tampering		•	•	•	•
Authenticity	AU.1 [MSG] Message Authentication Code (MAC) with session key				•	•
	AU.2 [FW] Verify authenticity of firmware when upgrading using digital signatures[a]		1	1	2	2
	AU.3 [FW] Verify authenticity of firmware/functions on boot using digital signatures[a]				1	2
	AU.4 [HW] Verify hardware authenticity					•
Non-repudiation	NR.1 [MSG] Freshness using counter or timestamp in authenticated message				•	•
	NR.2 [MSG] Audit logging				•	•
	NR.3 [MSG] Use of digital signatures for messages (signals)					•
Confidentiality	CO.1 [MSG] Encryption of messages				•	•
	CO.2 [FW] Encryption of firmware during transmission[a]				1	2
Availability	AV.1 [MSG] Limited network access – Quality of Service				•	•
	AV.2 [FW] Watchdog timer			•	•	•
Authorization and Access Control	AC.1 [MSG] Whitelisting of messages (signals) on gateways		•	•	•	•
	AC.2 [MSG] Whitelisting of messages (signals) on nodes				•	•
	AC.3 [MSG] Access control on function level				•	•
	AC.4 [MSG] Deployment of Intrusion Detection Systems				•	•
	AC.5 [MSG, FW, HW] Logical separation[a]			1	1	2
	AC.6 [MSG, FW, HW] Domain isolation				•	•
Other requirements[b]	OR.1 Fail in known state					
	OR.2 Information Input Validation					
	OR.3 Operate with least set of privileges that are necessary					
	OR.4 Compliance to secure coding guidelines					
	OR.5 Secure Logging					

[a] The numbers imply the class of a mechanism – higher numbers imply higher demands.
[b] These requirements have not been mapped to security levels as they are either required by laws and regulations, should be considered when developing secure systems or strongly depend on the application.

threat assessment techniques may vary for this example, however, the use case is used for illustration purposes to discuss possible security mechanisms in an environment consisting of different functions with different security levels communicating with each other. The security levels shown correspond to the abbreviations of the security attributes from Section III: $SL = [\mathrm{AU}, \mathrm{IN}, \mathrm{NR}, \mathrm{CO}, \mathrm{AV}, \mathrm{AC}]$.

B. Possible Attacks

There are many ways to attack this system, for example injection of faulty/wrong messages and modifications on the functions themselves. We will focus on the two target functions Speed Control and Braking to motivate the results of the TARA analysis and the resulting security levels shown in Fig. 3a. A modification or injection of faulty/wrong *VehicleSpeedCommands* or *VehicleSpeed* messages can cause the Speed Control to think that the vehicle is driving at low speed which results in a high acceleration. Modifications of the Speed Control function in the ECU can lead to dangerous situations if the *BrakeCommand* or the maximum acceleration are being altered. An attack on the Braking function is also dangerous if messages the function listens to are modified which entails a wrong brake behavior and may cause an



(a) Function view and resulting security levels from TARA analysis.

(b) Required security mechanisms per Function.

Fig. 3: Function view of the Cruise Control (CC) use case.

accident. Moreover, modifications of the function itself can have severe outcomes.

C. Required Mechanisms

Once each function has a security classification, it is possible to find the mandatory security mechanisms according to Table II. The required mandatory security mechanisms for this use case are illustrated in Fig. 3b, which shows the required mechanisms as boxes in each function.

It can be seen that functions with security levels up to level 2 only need to fulfill basic security requirements, such as verifying the firmware integrity when upgrading, whereas the functions Speed Control and Braking have higher security demands.

D. Applying the Framework

Mechanisms need to be deployed on the ECU and on the network or bus. Mechanism *AC.1 Whitelisting on gateways* needs to be deployed on the Edge Node GW and the Driver Control since they act as gateway between the Vehicle ECU and the Engine ECU. Also note that, if the source function requires *AU.1 MAC with session key*, the receiving nodes need to fulfill AU.1 for these messages. *AV.1 Limited network access* requires a change in the underlying network technology, as CAN itself does not provide means to limit the network traffic per sender. In this case one may argue that alternative security measures on the gateway, such as firewall-like functionality to handle DoS attacks or physical separation, in combination with *IN.4, IN.5, AU.2 and AU.3* are sufficient for this specific case. Another example is mechanism *AC.5 Logical separation*, it may be sufficient to use VLANs for class 1, class 2 on the other hand requires other virtualization techniques that are affecting all other functions realized on the same ECU.

Combining items inside one ECU or even within one subnet requires the aggregation of security levels by choosing the highest occurring security level for each element in the vectors. Nevertheless, it is up to the system designer to choose the level of aggregation that provides the best trade-off between detail and performance/hardware requirements.

V. Discussion

This is a first attempt towards a mapping between security levels and required security mechanisms for the automotive domain. We strongly believe that having a standardized and mandatory way to select required security mechanisms based on the security level of the component is the next step in vehicular security. There exist many models for how to assess threats which result in a security classification of components, and we continue at this point and identify suitable security mechanisms for each security level.

Strict rules are necessary since many parties are involved in the design and development process. Vehicle manufacturers develop some parts of the vehicular system in-house, but many components are provided by suppliers. We believe that such a strict rule-set is necessary in order to not leave the responsibility for developing secure systems to the individual developers or to third-party developers. With this framework in place, all involved parties know the minimum security measures that need to be implemented and all designers are already provided with basic protection against a large number of threats. Additionally, designers may also add extra application specific requirements that are not covered by the

required mechanisms. The decision of not implementing a certain mechanism may arise, due to other restrictions, such as cost and energy consumption. In such a case the choice of using alternative methods needs to be properly justified.

Moreover, this framework enables system designers to easily obtain an overview of the required mechanisms for each item in a bigger system context, making it possible to see dependencies between items, safety-critical or not, at an early stage.

We are aware that our proposed security mechanisms need to be defined in more detail, e. g., requirements on the Hardware Security Module, key derivation methods, encryption algorithms, and key lengths, and that the assigned security levels may need to be adapted, however, we are convinced that a framework similar to what we propose needs to be in place for automotive security. In addition, we show with the Cruise Control (CC) use case how this framework should be applied and we have validated the usefulness and correctness of the identified mechanisms and their corresponding security levels with a large vehicle manufacturer.

VI. Conclusion

Threat Analysis and Risk Assessment (TARA) is commonly used to derive security levels which indicate the security demands for components. Current models, such as HEAVENS [6], describe methods to obtain security requirements, however, they do not provide a direct mapping from security levels to security mechanisms and design rules which reduce the feasibility and impact of an attack. In this paper, we have identified appropriate security mechanisms applicable for the automotive domain and consequently associated these mechanisms with security levels that describe the demand for security. Having such a framework in place increases the efficiency and transparency when deriving security requirements for an automotive system, as current approaches place the task of deciding which security mechanisms to implement on the individual designers. We have additionally motivated the proposed mapping with an automotive use case which has been verified with a large vehicle manufacturer.

We have listed the identified mandatory security mechanisms required for specific security levels in Table II. Some components may require stricter rules in order to fulfill the demand on security but having a framework like we propose in place covers already basic security requirements that must be implemented.

The interaction between vehicle manufacturers and third-party developers of modules and ECUs benefits from such a framework as well, since all involved parties will have the same expectations of what has to be implemented for each component.

We believe that this framework is a first step towards a standardized mapping to security mechanisms. Such a framework is necessary, as it makes security design easier for system designers, developers, suppliers and all other involved parties. In addition, it prevents individual developers from making poor security design choices since the components have to provide the required security mechanisms.

In future work, this framework needs to be validated with more use cases and security mechanisms have to be specified in more detail. However, a task such as defining the specific algorithms to be used has to be performed by standardization organization in form of regularly updated recommendations.

Acknowledgment

This research was funded by the HoliSec project (2015-06894) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

References

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[2] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[3] "ISO 26262:2011 Road Vehicles – Functional Safety," International Organization for Standardization (ISO), Standard, 2011.

[4] Microsoft Corporation, "The stride threat model," 2005. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee823878.aspx

[5] "SAE J3061: SURFACE VEHICLE RECOMMENDED PRACTICE - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," SAE International, Standard, 2016.

[6] M. M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, "A risk assessment framework for automotive embedded systems," in *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security - CPSS 16*. Association for Computing Machinery (ACM), 2016.

[7] "ISO/IEC 27000:2016 Information technology – Security techniques," International Organization for Standardization (ISO), Standard, 2016.

[8] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weylr, "Security requirements for automotive on-board networks," in *2009 9th International Conference on Intelligent Transport Systems Telecommunications, (ITST)*. Institute of Electrical and Electronics Engineers (IEEE), oct 2009.

[9] "IEC 62443 – Industrial communication networks - Network and system security," International Electrotechnical Comission, Standard, 2013.

[10] "NIST FIPS PUB 199 – Standards for Security Categorization of Federal Information and Information Systems," National Institute of Standards and Technology, Standard, 2004. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.199

[11] "NIST Special Publication 800-53 – Security and Privacy Controls for Federal Information Systems and Organizations," National Institute of Standards and Technology, Standard, 2013. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-53r4

[12] S. Galgano, M. Talas, W. Whyte, J. Petit, D. Benevelli, R. Rausch, and S. Sim, "Connected Vehicles Pilot Deployment Phase 1 – Security Management Operating Concept – New York City," 2016.

[13] UNECE, "TFCS-09-14 Draft Recommendation on Cyber Security of the Task Force on CyberSecurity and Over-the-air issues of UNECE WP.29 IWG ITS/AD," 2017.

[14] *MISRA C: Guidelines for the Use of the C Language in Critical Systems 2012*. Motor Industry Research Association, 2013.

[15] "SEI CERT C Coding Standard Rules for Developing Safe, Reliable, and Secure Systems," Carnegie Mellon University, book, 2016.

[16] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[17] A. Yadav and C. Sandberg. (2018) Holisec reference architecture. [Online]. Available: http://autosec.se/wp-content/uploads/2018/04/HOLISEC_D4.1.3_v1.0.pdf