Chapter 9
Cyber Security Requirements Engineering

Christof Ebert

Abstract Virtually every connected system will be attacked sooner or later. This holds specifically for cloud-based services and systems. A 100% secure solution is not feasible. Therefore, advanced risk assessment and mitigation is the order of the day. Risk-oriented security engineering helps in both designing for robust systems as well as effective mitigation upon attacks or exploits of vulnerabilities. Security must be integrated early in the design phase to understand the threats and risks to expected functionality. The security analysis provides requirements and respective test vectors so that adequate measures can be derived for balancing security costs and efforts. This book chapter provides experience and guidance concerning how information security can be successfully achieved with a security requirements engineering perspective. Our experiences from embedded security in critical IT systems show that security is only successful with a systematic understanding and handling of security requirements and their interaction with functional requirements. Four requirements engineering-related levers for achieving security are addressed: security requirements elicitation, security analysis, security design, and security validation. We will show for each of these levers how security is analyzed and implemented. A case study from automotive systems will highlight concrete best practices. Only systematic and disciplined security requirements engineering will ensure that security needs are met end to end from concept to architecture to verification and test and—most relevant—operations, service, and maintenance.

Keywords Cloud-based systems · Cyber security · Embedded systems · Quality requirements · Validation · Systems engineering

C. Ebert (✉)
Vector Consulting Services, Stuttgart, Germany
e-mail: christof.ebert@vector.com

9.1 Introduction

IT evolution is driven by five forces: Collaboration, Comprehension, Connectivity, Cloud, and Convergence (Fig. 9.1):

- Collaboration, i.e., consumer Internet, social network interaction, single customer segmentation, configurators for products and services, digital money, computer-assisted collaboration tools, crowdsourcing;
- Comprehension, i.e., augmented reality, semantic search, big data handling, smart data, data analytics, data economy, online data validation, data quality;
- Connectivity, i.e., ubiquitous mobile computing, mobile services, cyber-physical systems, industry 4.0, machine-to-machine (m2m) communication, sensor networks, multisensor fusion;
- Cloud, i.e., applications and services in the cloud, location-based networks, new license models for software and application, sustainability, energy efficiency;
- Convergence, i.e., mobile enterprise, bioinformatics, Internet of things, pervasive sensing, autonomous systems.

The five forces all relate to cyber security and will not work adequately without end-to-end cyber security engineering. Coupled with the underlying complexity and scale these drivers demand new solutions for cyber security. Examples include new IT architectures that facilitate seamless connectivity, robust infrastructures for cyber-physical systems in safety-critical environments, or data analytics to predict choices and behaviors to improve overall customer experience. Such software-driven solutions can create nontraditional market entry points and

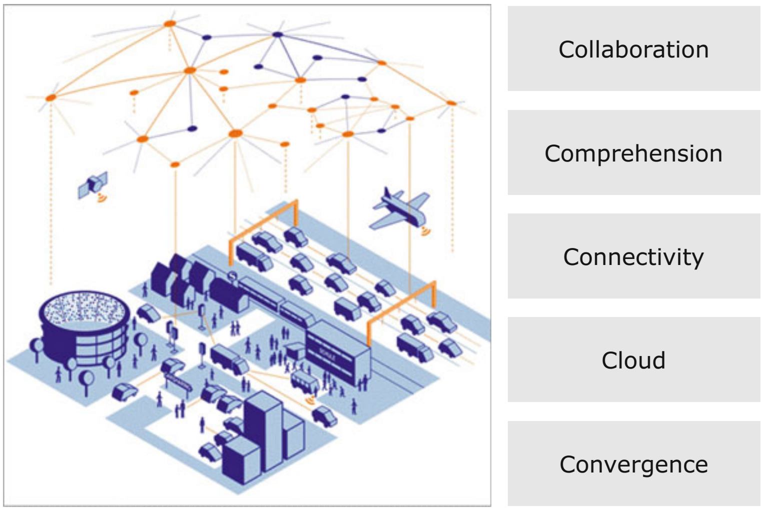


Fig. 9.1 Cyber security is impacted by five forces: collaboration, comprehension, connectivity, cloud, and convergence

consequently entirely new mechanisms to address a single customer with time-specific and location-specific services.

New technologies not only create numerous opportunities but also introduce complexity. Thereby, these solutions introduce new challenges, for instance, with respect to information security, robustness, and usability.

Security and robustness have tremendous impact on business decisions. The more we share and network, the more we are exposed to attacks of all kinds. The exploding need for secure software and protection schemes for our business processes, end to end, indicate this impact. Imagine automotive suppliers working on multisensor fusion connected to GPS and vehicle-to-vehicle communication to predict critical situations and foresee appropriate measures at situations where even the driver might not even be aware of what will happen. Another example is service companies who leverage their sales channels to flexibly provide related services such as door-to-door transportation, or firms that offer a single service card for identification, payment, and access to services of various providers both physical and in the cloud.

Complexity and scale demand focus on usability. We already face situations where users without adequate training are forced to operate systems which they do not understand sufficiently to meaningfully assess risks and stay in control across normal day-to-day scenarios. Insufficient usability today is a major source of critical failures caused by humans in health care, transportation, and production plants.

For embedded software–hardware systems complexity and technology will grow fast. The resulting competence gap will lead to even stronger fight for skills. From the survey and interviews, we can see that companies will continue to invest in growth through innovation by developing new products and solutions, because this determines their market position. They are aware of the volatile market situation and want their development teams across the world to be as lean and innovative as possible.

The IT industry deals already since years with strategies for data protection and to provide secured networks to prevent them against unauthorized access. Wide experiences are available here that, with special considerations, can be adapted and are useful for different industries. This allows, for instance, taking over the proven software architecture of Ethernet, so that a number of approved protocols are available as well for a secured data transmission. Essentially, they are based on cryptography, software algorithms based on more or less complex mathematics. The algorithms itself are not the secret and are available to the public, but keys provide the secret and they must be created, distributed, and maintained carefully. A popular key management system used by the IT industry is the PKI (Public Key Infrastructure). It contains a hierarchical certificate management with associated keys and builds the basis for an authenticated communication between partners.

We will look in this book chapter to key elements of security requirements engineering, namely requirements elicitation and security requirements analysis.

Our examples mostly come from automotive systems, because unlike any other industry, automotive connects three relevant drivers of modern IT systems, namely the following:

- Systems Engineering with a combination and integration of mechanics, hardware, and software;
- Embedded real-time systems with safety-critical requirements;
- IT systems with huge computing power and distributed cloud services.

On this basis technology transfer to other industries is easily feasible.

9.2 Cyber Security Requirements

Security is a quality attribute which heavily interacts with other such attributes, such as availability, safety, or robustness. It is the sum of all attributes of an information system or product which contributes toward ensuring that processing, storing, and communicating of information sufficiently protects confidentiality, integrity, and authenticity. Cyber security implies that it is not possible to do anything with the processed or managed information which is not explicitly intended by the specification of the embedded system [1, 2].

Based on the specific challenges of cyber security, system, and service suppliers have to realize an effective protection against manipulations of IT and embedded electronic and electric systems. Key points in the development of protected systems are the proper identification of security requirements, the systematic realization of security functions, and a security validation to demonstrate that security requirements have been met. The following items need to be considered to achieve security in the development process:

- Standardized process models for a systematic approach which is anchored in the complete development process. This starts in the requirements analysis through the design and development to the test and integration of components and the network.
- Quick software updates to close vulnerabilities in installed operational software, be it in the cloud or embedded in systems.
- Reliable governance that is state of the art and meets long-term security demands, such as key management and updates of crypto algorithms.
- Robust networks and system architecture that provides flexibility and scalability and are designed under consideration of security aspects.

Based on our experiences in many cyber security projects, we show which security engineering activities are necessary to create secure systems and how these activities can be performed efficiently. In the following discussion, we want to examine each of these topics, the current activities, and provide suggestions concerning how to mitigate the security risks.

Traditionally systems and electronics requirements are function driven. But, by defining functionalities alone, there is nothing said about the correlation of features which is where security risks typically show up (see our introductory example). We will start with explicit security requirements, as they have emerged in IT systems over time [1–5]:

- Confidentiality demands for information being unavailable for unauthorized entities. Note that data may be gathered by unauthorized entities without losing confidentiality, as long as the information contained in the data is not revealed.
- Integrity requires information remaining unchanged by unauthorized entities.
- Authenticity necessitates that the origin of information or the identity of a communication partner can be satisfactorily proven.
- Availability hardens that the system to be protected by making it highly reliable, including all necessary cyber security mechanisms.
- Governance ensures that agreed policies and protection mechanisms both hard and soft, specifically those being people oriented, are used and part of the culture, independent of time pressure or budget impacts.

Security requirements encountered in IT, cloud services, and embedded systems development typically target its dependability. Such systems are embedded into a technical process, thus dependability is imperative to prevent failures of the technical process itself or its environment. Dependability demands that system functionality, determined by its functional requirements, is delivered correctly—considering feature correlations and disturbances from the outside. Besides accurate realization of the functionality, information needs to be correctly processed during operations of the embedded system, i.e., without being distorted during transmission or storage. Additionally to the need for correct information processing, embedded systems interact with real-world objects, which means they are subject to real-time requirements of these real-world objects. Information must not only be processed correctly, but also within determined time limits.

Both cyber security requirements and embedded systems' reliability requirements have one thing in common: They aim to deflect unauthorized manipulation of information inside of computer systems—be it interferences with the system environment or intentional manipulations of unauthorized entities (i.e., attacks).

9.3 Risk-Oriented Security

Over the past decade trends like IoT, connected service workflows and driver assistance systems among others have led to software and connectivity playing an increasingly important part in developing critical systems and also for business models of OEMs and suppliers likewise.

Devastating impact of security issues is already known from industrial sectors like IT-infrastructure, aviation, information technology and telecommunications, industrial control systems, and energy and financial payments. Virtually every connected system will be attacked sooner or later. A 100% secure solution is not feasible. Therefore, advanced risk assessment and mitigation is necessary to protect assets. Consequently, the typical solution to security in these industries relies on suitable risk assessment that projects threats on assets of interests. Thereby cost of implementing specific security measures can be compared with the probability of a particular threat that they counter.

Security in a complex system cannot be achieved by applying countermeasures on single items. It requires an analysis of the complete functionality or system as a whole and to apply countermeasures as an integral part. First, you need to identify what are the assets I want to protect. Besides financial aspects also confidentiality and safety functions must be considered carefully. The next step would be a threat analysis: who has access to my assets, what are potential attackers, and where are my access points. A typical approach to this is the construction of a data flow diagram in which the assets are identified. It provides an overview of all connections and access points, where attacks and manipulations can be achieved. From the material above a risk assessment can be done to obtain the measurements and results in a classification of the risk. An example of such a risk assessment can be found in the picture below. Here, as an example, the classification was defined in three categories: Low, medium, and high (Fig. 9.2).

This process provides systematic means to deal with the subject and results in a balanced trade-off for cost and efforts. Depending on the determined security level, countermeasures can be defined on system level and further derived as security input requirements. The analysis phase provides now also requirements for hardware extensions, e.g., if hardware acceleration is needed for authentication or if a specific key management is required for higher security measures. The requirements are also an input to define test vectors on functional (e.g., for an ECU) and system level (e.g., for the vehicle). These tests, together with standard penetration tests, then will help to provide evidence for successful application of the security to the function and system.

Asset-based risk assessment is a suitable tool for companies to steer efforts for security engineering in a systematic and comprehensive way and thereby involve all relevant stakeholders in the organization. For example, a CEO may not find it very helpful to have a long exhaustive list with every attack vector or potential threat—they need to be provided with a ranked listing and useful decision-support tools which clearly shows alternatives and consequences. From the view of a system

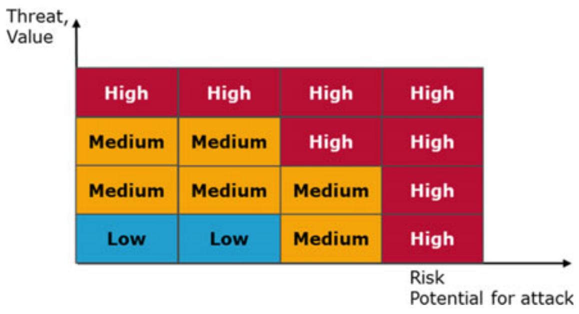


Fig. 9.2 Definition of security level derived from threat analysis and risk assessment

developer, a flat listing of potential threats might not help to improve the system. To really help, they need to be able to map security threats, countermeasures and requirements to system/architecture elements in their scope of the project.

The systematic management of security threats and associated security goals is essential to actually providing safe and competitive products, and to protect valuable assets and business models.

But what makes security engineering so complex?

Developers face the challenge of securing a system against attackers whose capabilities and intentions are at best partially known. Some attacks might today appear infeasible, but today's impossible attacks might become more likely in the near future. An example of this is attacking a vehicle simply by exploiting wireless interfaces, 20 years ago would have been extremely unlikely; however, today a cheap software-defined radio accomplishes these types of attacks with little effort. On the other hand, an attacker might invest more effort into launching an attack the more valuable a successful attack is to him. Some attacks represent more effort to the attacker than others given the specific potential of the attacker. It is this risk/reward payoff that is analyzed in security engineering.

Likewise during testing and verification, suitable methods to verify that the vehicle has the required security level and process goals like test strategy and coverage need to be chosen.

Furthermore, the assets to be protected from attacks are decided by stakeholders involved, e.g., drivers would indicate different assets of their vehicle to be protected compared with what a developer considers an asset. However, customers/drivers need to be satisfied with their vehicle in order to buy another one from the same company. Consequently, security engineering must seek trade-offs between cost of security measures and benefit to assets in order to make sustainable decisions.

Security concepts must balance the cost of not having enough security and thus being successful attacked with all damaging consequences and the cost spent to implement appropriate security mechanisms and keep them updated along the life cycle—covering service workflows as well (Fig. 9.3).

To summarize, the relationship between assets, attackers and threatsis complex and dynamic (e.g., attacks are more probable the less effort is required and the more

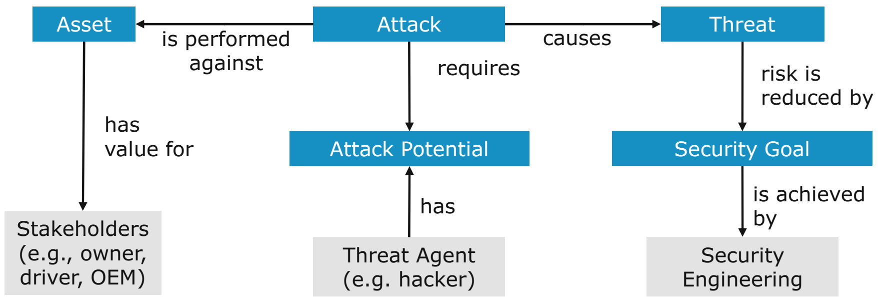


Fig. 9.3 Overview of cyber security analysis process

value successful attacks represent; attack vectors and effort change over time). Furthermore, common understanding of assets among all stakeholders of security engineering is mandatory in order to provide information for steering the security engineering.

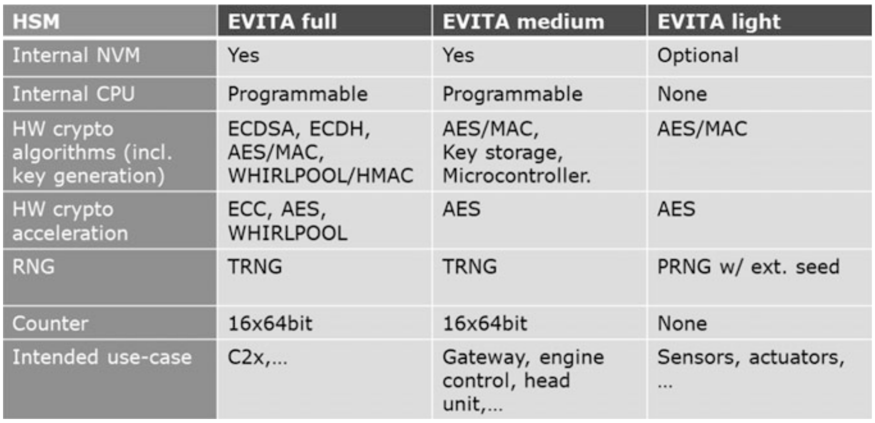
Choosing the right set of security engineering methods for analysis, concept, and testing is challenging but required in order to enable goal-oriented and manageable security engineering.

Risk-based Security Engineering combines state-of-the-art methods for cyber security risk assessment in a practical framework and supports all involved stakeholders to develop "secure-enough" products. The method and our approach for proposing a concrete technical security concept is based upon security best practices such as the following:

- ISO 15408 (Evaluation criteria for IT security) with its focus on IT systems, specifically the seven evaluation assurance levels (EAL) for security requirements and guidance on common criteria.
- ISO 27001 (Information security management systems) with its governance requirements for security engineering across the entire value chain.
- IEC 62443 (Industrial communication network security) with its strong view on distributed systems and necessary security technologies and governance.

The Vector Security Check and our security engineering method adopt the state of the art not only from standards mentioned above but also from significant research work. For example, several research projects like "E-safety vehicle intrusion protected applications" (EVITA) funded by European Union and HEAVENS, proposed solutions for security risk assessment (Fig. 9.4).

We will further on show by examples how to use the risk-oriented security concept covering the entire security life cycle with focus on the upper left activities, namely



HSM	EVITA full	EVITA medium	EVITA light
Internal NVM	Yes	Yes	Optional
Internal CPU	Programmable	Programmable	None
HW crypto algorithms (incl. key generation)	ECDSA, ECDH, AES/MAC, WHIRLPOOL/HMAC	AES/MAC, Key storage, Microcontroller.	AES/MAC
HW crypto acceleration	ECC, AES, WHIRLPOOL	AES	AES
RNG	TRNG	TRNG	PRNG w/ ext. seed
Counter	16x64bit	16x64bit	None
Intended use-case	C2x,...	Gateway, engine control, head unit,...	Sensors, actuators, ...

Fig. 9.4 EVITA classification for the hardware security module (HSM)

- Asset Definition and Threat and Risk analysis
- Security Goals
- Security Concept.

9.4 Industry Case Study

To better illustrate evolving cyber security needs we will look to modern automotive systems. Figure 9.1 shows the interaction of functions in their distributed networks being an essential part for our today's modern infrastructures with their needs for safety and comfort. Besides the further development of innovative sensors like radar and camera systems and the analysis of the signals in highly complex systems, the connected cars will be a driving factor for tomorrow's innovation. Internet connections will not only provide the need for information to the passenger.

Cloud-based functions like eCall or communication between cars or car to infrastructure (vehicle2x) shows high potential to revolutionizing the individual traffic. This includes the improvement of the traffic flow controlled by intelligent traffic lights, warnings from roadside stations, or brake indication of adjacent cars. This builds the basis for enhanced driver assistant systems and automated driving. But the connection to the outer world also bears also the risk for attacks to the car.

Figure 9.5 shows interworking for vehicle2x and external communication that are already available today or will become available in the near future. Each connection to the car has a potential risk for an attack, regardless whether it is wireless

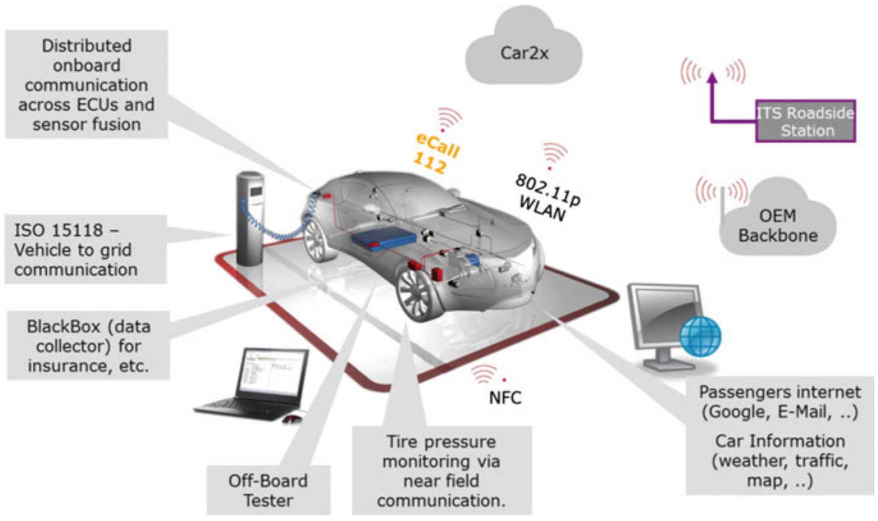

Distributed onboard communication across ECUs and sensor fusion
Car2x
ITS Roadside Station
eCall 112
802.11p WLAN
OEM Backbone
ISO 15118 – Vehicle to grid communication
BlackBox (data collector) for insurance, etc.
NFC
Passengers internet (Google, E-Mail, ..)
Car Information (weather, traffic, map, ..)
Off-Board Tester
Tire pressure monitoring via near field communication.

Fig. 9.5 Car with remote connections

or wired. Just the threat is different. The access through a connector is only possible for a limited amount of cars, whereas a far field connection can be accessed from anywhere in the world. But also near field connections play an important role, such as tire pressure monitoring system, Bluetooth, and wireless LAN. Security and reliability of these connections will be essential for the acceptance and success of these systems. With the introduction of this technology precautions must be taken to increase the reliability and to reduce the vulnerability to the system.

We will show security engineering with the example of a connected car. The car is connected to an external cloud from which it receives secured updates and diagnosis support. The car itself has numerous controllers (so-called ECU, electronic control units) which are connected by secured bus systems (see also Fig. 9.1).

As an example, we utilize the simplified functionality of an Automotive embedded control unit (ECU) that controls the automatic opening and closing of the roof of a convertible. Security impacts are manifold with this example, from getting access to the car and its contents to inserting a safety hazard to the driver if the roof opens during driving. The top level functional requirements are presumed to be (the abbreviation FR denominates functional requirements and SR security requirements):

- The roof is to be opened, if the open roof button is pressed (FR 1).
- The roof is to be closed, if the close roof button is pressed (FR 2).
- When the roof is completely opened or closed, feedback is given to the driver (FR 3).
- Additionally, two-top level safety requirements are supposed:
- The roof is not allowed to move, if the speed of the car is greater than 10 km/h (SR 1).
- If an obstacle is detected in the direction of movement of the roof, the roof has to stop the movement within 0.1 s (SR 2).

To conduct the security analysis, we assume the following system situation. A controller receives the following information to execute its functionality: “open roof button pressed,” “close roof button pressed,” “vehicle velocity,” and “obstacle detected.” The following information is sent by the ECU: “roof completely opened” and “roof completely closed.”

All information is received or sent via the embedded network, implemented by bus systems such as CAN and Flexray, which are controlled by the AUTOSAR base software. The ECU actuates the roof motor by controlling the electric current to the motor. Obstacles are detected by a smart sensor, which is also connected to the CAN bus. The system functionality is realized as software on one microcontroller inside the ECU. The microcontroller features internal nonerasable memory. Figure 9.6 depicts the system.

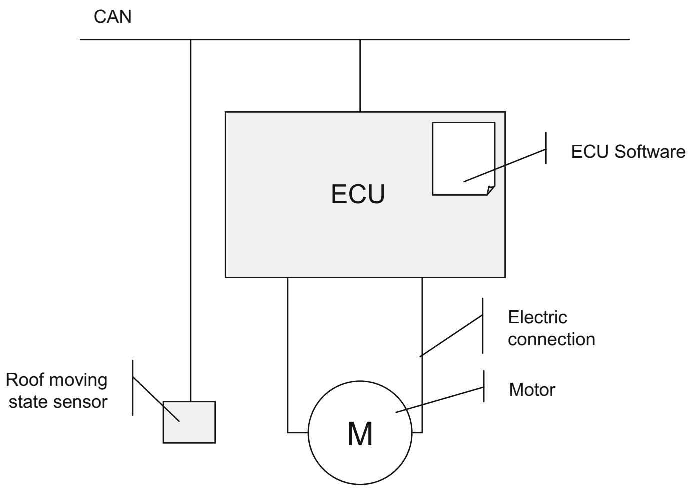


Fig. 9.6 Assumed system layout

9.5 Security Requirements Elicitation

A first step is setting the security objectives. When considering above requirements, it becomes clear that the detail level of such information is not sufficient for security analysis. Possible security threats emerge from unauthorized information gathering and manipulation. To judge these possible attack vectors, more detailed knowledge of the embedded system in question is required. To discuss security threats, the communication transactions of the system must be known as well as the effect of these transactions.

Knowledge of the communication technology to be employed is equally important, because different communication standards imply different vulnerabilities, where an attacker can mount an attack. The same holds true for determination of security threats against device software. The distribution of functions on different devices is to be known as well as underlying hardware details. The device hardware constitutes, which (hardware) interfaces can be used by attackers and if stored information can be deleted or modified.

9.6 Security Analysis

To understand vulnerabilities and determine security risks we apply misuse cases. Similar to use cases, misuse cases show a specific way to use a system. Misuse cases describe sequences of events that, taken together, lead to a system doing something that is not intended or even unwanted. Misuse cases that imply an unacceptable risk are taken to deduce concrete security requirements which are subsequently translated into functional requirements. Here is additional concrete guidance: Each identified security requirement must be linked to at least one functional requirement that is linked to design artifacts and test cases and monitored until closure—from design to validation and service.

9.6.1 Threat Analysis

In the first step, we will discuss the possibilities of an attack on the convertible's roof. First, the information that are received by the roof ECU and used to act accordingly are to be considered. Second, the ECU software program, which implements the ECU functionality, needs to be regarded. Both information entities, transmitted via communication systems or stored as a program, can, in principle, be tampered with by an attacker.

Different transmitted information is used by the roof ECU:

- Information that the roof is to be opened or closed (from FR 1, FR 2)
- Information on vehicle velocity (from SR 1)
- Information if an obstacle is detected (from SR 2).

As depicted in Sect. 9.3, there are different ways to attack communication. The network, which is used to transmit information concerning the roof ECU, shows vulnerabilities against all these ways of attack. Thus, the following functionalities need to be considered to protect such distributed communication:

- Protection of confidentiality to prevent acquisition of information by attackers
- Protection of content integrity to detect manipulation of messages by attackers
- Protection of authenticity to detect broadcasting of messages by attackers
- Protection of temporal integrity to detect delay or replay of messages by attackers.

The program controlling the ECU is verified by means of checksums, making it difficult for an attacker to change it. Attack paths thus need to consider software updates starting from the code creation and its validation up to its delivery in a repair shop anywhere in the world. Determining attacker motivation is difficult in the given case. Generally, it seems unlikely in the example that someone would manipulate functionality as the one depicted above. However, we will look into possible impacts of manipulations to determine which protection functionalities need to be realized and which can be disregarded.

9.6.2 Risk Assessment

Possible hacker motivations may include curiosity or sabotage. Thus, we quantify attacker motivation with "3," meaning a medium motivation to attack.

Because of the high degree of publicity of bus specifications, its vulnerabilities to attacks, and the availability of hardware/software tools for manipulation, attacker capabilities need to be judged at least to be "4," meaning the attacker possesses advanced capabilities to manipulate the bus.

Effects of attacks depend on the information to be manipulated. Misuse cases related to the functional requirements presented result in malfunctions that may be inconvenient but are essentially harmless. A forged request to open or close the roof would result in the opening or closing of the roof, without the driver requesting this operation. Here we have a security requirement with clear safety impacts. If the messages containing information about vehicle velocity or obstacle detection are manipulated, the roof could be opened at high speed or the closing roof would not be interrupted despite an obstacle in the path of the roof. Both incidents can result in the mentioned effects, so the cost effect is assigned to be "5," resulting in a risk priority number of 60.

Assuming an acceptable residual risk of 50, one would define the following security requirements:

- Vehicle velocity data communication must be protected
- Obstacle detection data communication must be protected.

We see that dependability requirements are a good starting point to identify relevant security requirements and to guide elicitation of further functional requirements that will mitigate security risks. The same technique as outlined here can be applied for other scenarios—always starting with attacker motivation or functional risks due to the system architecture. Our guidance: Do not limit exposure to known incidents and defects as some textbooks suggest. Security analysis is not a checklist approach. It has to consider attack motivations of persons thinking different than the usual engineer. However with an engineering mind, we can easier identify vulnerabilities in our architectures.

9.7 Security Design

Although one might argue that design is not much related to security requirements engineering, we will elaborate some of the techniques to show how traceability from security requirements to their implementation is achieved. Without such traceability not only the validation is impossible but also there would be no way to prove—after an incident—that the necessary cautions had been taken.

9.7.1 Security Functionality with Minimal Resource Impact

Different mechanisms exist to realize protection of communication: encryption for protection of confidentiality, message authentication codes for protection of content integrity, digital signatures for protection of authentication, and time stamps as well as sequence numbers for temporal integrity protection. Since these mechanisms need to be deployed to embedded systems at field level, the realization of the mechanisms must strive for minimal resource (especially memory) consumption. Therefore, the notion is to avoid the use of monolithic protective mechanisms, such as digital signatures, but to identify more fine-grained mechanisms instead, which provide protection functionalities by combination of one or more of such smaller mechanisms. Ideally, these protective mechanisms can be used to provide different protection functionalities, while being implemented only once.

To provide confidentiality, encryption is the mechanism of choice. For content integrity, cryptographic hash functions exist, but an attacker, who is able to change the content of a message, will also be able to compute the hash value and change it, pretending the integrity being intact. Therefore, keyed hash functions exist, which secure message integrity against purposeful manipulation by incorporating encryption into the hash value. If encryption already has been selected to realize confidentiality, it can be reused in conjunction with hashing to provide content integrity consequently saving resources.

Likewise, authenticity can be provided by digital signatures, which can also be constructed using an ID and keyed hash functions. It is then realized with a non-ambiguous identifier of a device and the reuse of hash and encryption functions. Finally, temporal integrity can be verified with time stamps and sequence numbers, but for these mechanisms to work, it is required that an attacker is not able to manipulate time stamps/sequence numbers or pretend to be the origin of the message. This requires the aforementioned mechanisms. The delineated dependencies yield a layered structure where major implementations are linked with security requirements thus facilitating semi-automatic consistency checks (Fig. 9.7).

To manipulate functionality, attackers with physical access to devices can flash the device memory with new programs, which fulfill the attackers' requirements. Examples for this kind of attack are manipulations of mileage indicators or unlocking of programmed limitations (e.g., maximum speed) of motorized vehicles. Modern microcontrollers can usually be flashed using defined interfaces, such as

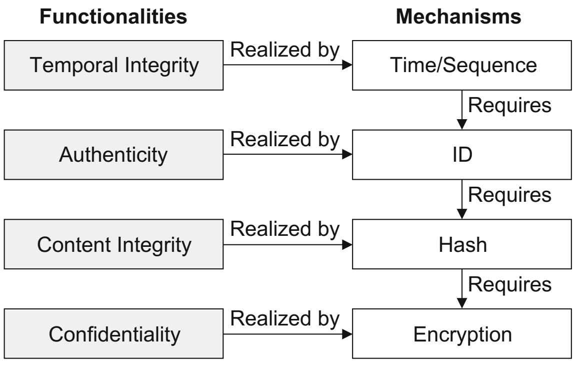


Fig. 9.7 Protection functionalities and mechanisms for communication

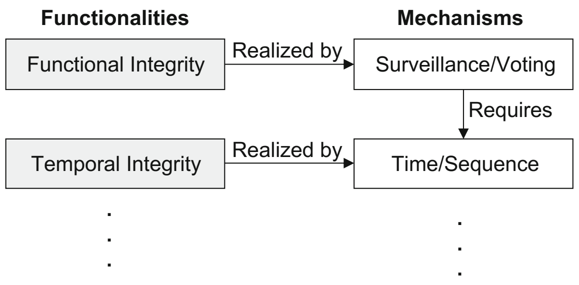


Fig. 9.8 Protection of functional integrity and communications security

JTAG or SPI. In consequence, surveillance functionality is required to monitor the integrity of device functionality, i.e., the program code. A complete deletion of the device memory would also eradicate the surveillance functionality. Therefore, surveillance functionality must be distributed to different devices, which then monitor the functional integrity of other devices.

In case of detection of irregularities, e.g., manipulated code or unreachable devices, the monitoring devices can vote for counter measures (assumed, there is more than one device monitoring the manipulated one) and react in an appropriate way, for instance, by ignoring messages of the manipulated device or by transferring the technical process into a safe state.

For distributed surveillance and voting, communication is required. This communication must be protected against manipulation. Otherwise, an attacker might exploit this functionality to simulate manipulated devices. Thus, the layered structure can be extended with a fifth layer (Fig. 9.8).

9.7.2 Composition of the Layers

The layers, i.e., the mechanisms to realize the desired functionalities, each provide a service. Every service specifies what activities are to be realized on a certain layer, but not how the activities are realized. To satisfy real-time requirements, the realization, i.e., the selected algorithms must be deterministic. This is the case for most cryptographic algorithms, making it possible to calculate an execution time for the algorithms. To meet the demands of timing constraints, efficient algorithms ought to be selected.

This composition offers a high degree of adaptability and flexibility. It is possible to adapt to three different conditions, which need to be determined during the risk analysis phase of security engineering:

- Parts of the system are secured by physical protection. Thus, only unprotected parts need to be protected by software-based mechanisms. If, e.g., a field device is physically protected, it is not necessary to implement functional integrity checks. Thus, the lower four layers need to be realized with software-based mechanisms, the fifth layer is then realized as a physical mechanism.

- Certain system elements are not vulnerable to specific attacks, because they already include security mechanisms or there are intrinsic features that prevent or detect these attacks. In that case, the respective layer can be a "dummy" layer, which does not contain a software-based protective mechanism.
- Even if a system is susceptible to attacks, it might be the case that there are only minimal resources available in a controller. It is then possible to select mechanisms that consume few resources (which might result in reduced protection strength). Another possibility is to realize only a "base" protection, using only lower layers.

While these adaptations can be made during development time, it is also possible that ambient conditions change during run time. During the long life span of field level system elements, it is probable that specific protective mechanisms are compromised, which has been the case with several cryptographic algorithms during the last years. Thus, it is necessary to be able to exchange protective mechanisms, even when cars are already on the road. The ability to flexibly exchange protective mechanisms during run time depends on the implementation of the mechanisms, which is depicted in the next section.

9.7.3 Implementing Security Functionality

When implementing protective mechanisms, the limited resources of embedded systems need to be considered. Additionally, implementations of these mechanisms should be tested and well proven. Otherwise, vulnerabilities due to faulty code could be inserted into the system. On this account, reuse of existing software components is a promising approach. Therefore, a software component technology has been selected, which allows for implementation in structured programming languages. Such structured components for embedded systems are implemented in "structured C."

Figure 9.9 shows the assembly of one layer. The layer component ("SecurityLayerX") can access different interfaces ("ISecurityMechanismA," "-B," "-C") of protective mechanisms ("SecurityMechanismA," "-B," "-C"). So, multiple mechanisms can be used on every layer, e.g., to provide different kinds of encryption. The interface "ISecurityLayerX" is used to make the service of layer X available to the upper layer X + 1 in a uniform way. Likewise, the layer component uses the interface "ISecurityLayerX − 1" to access the service of the lower layer X − 1 (Fig. 9.9).

To realize protection, the required protection functionalities need to be selected. All layers, which provide required services, are to be set in. Furthermore, the concrete mechanisms are to be chosen, which fulfill the given requirements (e.g., integration into an existing system that requires asymmetric encryption). The

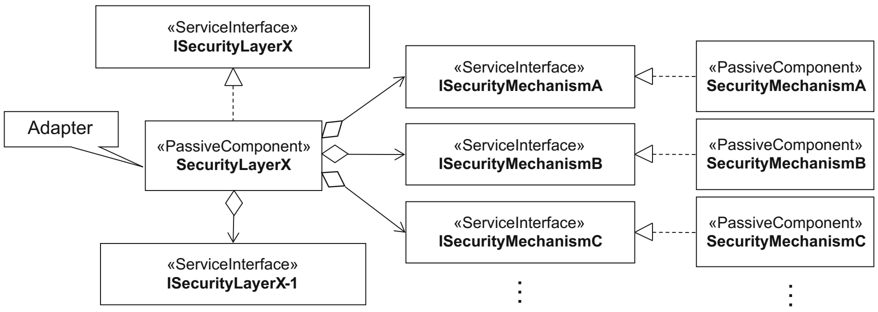

«ServiceInterface» ISecurityLayerX
Adapter
«PassiveComponent» SecurityLayerX
«ServiceInterface» ISecurityLayerX-1
«ServiceInterface» ISecurityMechanismA
«ServiceInterface» ISecurityMechanismB
«ServiceInterface» ISecurityMechanismC
«PassiveComponent» SecurityMechanismA
«PassiveComponent» SecurityMechanismB
«PassiveComponent» SecurityMechanismC

Fig. 9.9 Assembly of a layer

chosen layer components ("SecurityLayerX," "-X-1," …) need to be connected to each other in order to be able to access the lower layer services and to the selected mechanism components, which implement the services of the layer. With the ability to dynamically exchange components during run time (given the underlying hardware platform supports modification of software at run time, e.g., by flashing), there is a high degree of flexibility of the concept. With this flexibility, it is possible to update implementation failures as well as compromised algorithms. Modern security modeling tools allows tracing security requirements (and their functional counterparts) down to the code-level as requested when security certification should be done before release.

9.8 Security Validation

Security validation is conducted on two different tiers. To ensure the quality of the software components, every component is subject to a rigid review process looking for typical design errors and manually checking adherence to security requirements. Additionally, comprehensive unit and system tests are made. Like all other software components, the test cases themselves may contain errors as well and should be checked before use. Such quality ensuring procedures are imperative for security functionality, because these are often targeted by attackers to manipulate the protected system.

Automatic regression testing of security requirements is absolutely mandatory due to the many changes to the code during the product life cycle. To automate testing of security requirements, an automatic penetration test tool for embedded systems has been created and used in real-world embedded systems. This penetration test tool makes it possible to define attacks against embedded systems based on the identified security requirements. The prototype allows for automatic execution of these attacks and detection of attack results, i.e., if an attack was successful or has been neutralized by the security functionality.

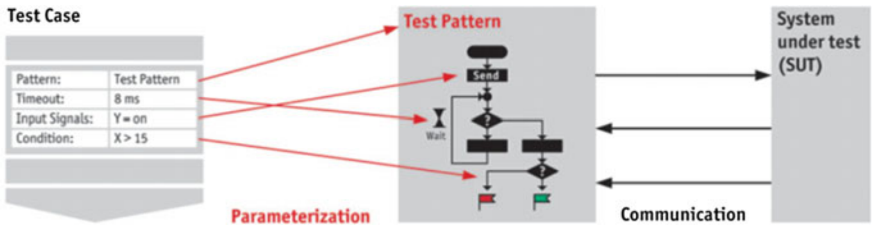


Fig. 9.10 Automated security validation based on specific security test patterns

Abstraction is a commonly used and important method for handling complexity in software development and system design. Abstraction on the signal level is a common way to test ECU functionality. In a common distributed network system, for example, an interaction layer in the ECU provides the signal abstraction. Abstraction layers in ECU and test environment must utilize the same abstractions to allow same reference signals, authentication checks, and even sandboxing of unknown signal patterns. Simultaneously, signal abstraction also represents—at least on the protocol level—the remaining bus simulation. For example, it ensures that periodic signals are actually transmitted periodically. This allows using security test pattern also for real-signal load and overload or DOS-attacks. When a change is made to the system's communication matrix, such test patterns and associated test cases are reused thus ensuring consistency during the life of a component.

Figure 9.10 shows security validation on the basis of security test patterns. Tools such as PREEvision are used to model the system, its embedded components, the network (both sensors and actuators) to the level, where the control algorithm is detailed in a controller model.

9.9 Relevance and Outlook

Security is thus of growing relevance to all industry areas. Both advanced IT systems as well as embedded systems increasingly utilize cloud-based networked software components based upon standardized open architectures. Due to their long lifetime within changing environments, different versions and configurations are combined in different variants over time with software or hardware upgrades.

Currently used concepts, such as proprietary subsystems, the protection of components, firewalls between components and the validation of specific features are insufficient to ensure security on a systems level [5]. Intelligent attack scenarios evolve from different directions, such as attacks on unprotected networks, introduction of dangerous code segments through open interfaces, changes to configurations, and prove that security has to become a topic throughout the entire organization and with high management attention.

Systematically ensuring security from requirements to service of systems

- protects against manipulations,
- increases the safety and reliability of users, and
- facilitates even more software-driven services, applications and business models.

Security demands an end-to-end requirements engineering perspective. The article with its many practical examples underlines that security of IT and embedded systems can be achieved with clear and systematic focus and limited extra effort on the basis of disciplined requirements engineering. Security engineering in embedded systems has to start with a clear focus on security requirements and related critical quality requirements, such as safety, footprint, or performance and how they map to functional requirements.

Software suppliers and integrators first define the key functional requirements. These requirements are then analyzed on their security risks and impacts. Security requirements are expanded into further functional requirements or additional security guidelines and validation steps. Requirements engineering security concepts are subsequently and consistently (i.e., traceable) implemented throughout the development process. Finally, security is validated on the basis of previously defined security requirements and test cases.

We practically showed how security requirements engineering is mastered along the entire system life cycle. Many security attacks are the result of poorly managed software updates and uncontrolled complexity growth. Architectures, systems, and protocols have to be developed with security in mind (i.e., design for security). Competences have to be developed around security engineering, and employees have to be trained how to design, verify, and sustain security throughout the product's life cycle. Most important it is that before-mentioned methods and processes are implemented consistently, systematically, and rigorously with traceable effects. Only with continuous measurements on their effectiveness the value of security measures improves.

Traditional embedded software engineering ignored security for various reasons, such as having isolated components, dealing with heavily constrained resources, and being unable to handle the computational overheads. Today however, embedded security is in the foreground due to safety, legislative, and intellectual property concerns [5].

With our described product life cycle-oriented security requirements engineering, the good news is that different from Internet security securing embedded systems is likely to succeed in the next 5 years. By doing so, embedded system suppliers and integrators are increasingly in a position that allows marketing and selling security as part of an overall quality concept. It will help to master liability risks and to ultimately increase revenues.

References

1 Cyber Security and Functional Safety white papers and practice guides: www.vector.com/security, www.vector.com/safety
2. Ebert, C.: Systematic Requirements Engineering. Dpunkt, Heidelberg, Germany, 5. edition, 2014.
3. Firesmith, D. G.: Engineering Security Requirements. Journal of Object Technology, Vol. 2, pp. 53–68, 2003.
4. Giorgini, P., F. Massacci, and N. Zannone: Security and Trust Requirements Engineering. In Foundations of Security Analysis and Design III—Tutorial Lectures, LNCS 3655, pages 237–272. Springer, 2005.
5. Haley, C.B., J.D. Moffett, R. Laney, B. Nuseibeh: A framework for security requirements engineering. Proc. SESS 2006, 2006.
6. ISO/IEC 15446:2004. Information technology—security techniques—Guide for the production of protection profiles and security targets. 2004.
7. ISO/IEC 15408:2005. Information technology—Security techniques—Evaluation criteria for IT security (Common Criteria v3.0), 2005.
8. ISO 27001:2006, Information Security Management—Specification With Guidance for Use. International Or-ganization for Standardization, 2006.
9. Mead, N.R.: How to compare the security quality requirements engineering (SQUARE) method with other methods. Software Eng. Inst., CMU/SEI-2007-TN-021, Aug. 2007.
10. Poulsen, K.: Slammer worm crashed Ohio nuke plant network. SecurityFocus, http://www.securityfocus.com /news/ 6767, 19.08.2003.
11. Ramachandran, M: Software Security Engineering: Design and Applications. Nova Science Publishers, New York, USA. ISBN: 978-1-61470-128-6, https://www.novapublishers.com/catalog/product_info.php?products_id=26331, 2012.
12. Ramachandran, M : Software Security Requirements Engineering and Management as an Emerging Cloud Service, International Journal of Information Management, Vol. 36, No. 4, pp 580–590, 2016. doi:10.1016/j.ijinfomgt.2016.03.008.
13. S. Myagmar, A. J. Lee W. Yurcik: Threat Modeling as a Basis for Security Requirements, National Center for Supercomputing Applications (NCSA), University of Illinois
14. Sindre, G. and A. L. Opdahl: Eliciting security requirements with misuse cases. Requirements Engineering, No. 10, pp. 34–44, 2005.
15. Wired: Hackers remotely killed a Jeep—with me in it. www.wired.com/2015/07/hackers-remotely-kill-jeep-highway, July 2015
16. Whitman, M., Mattord, H., Principles of Information Security, Course Technology, Boston, 2007.
17. Yoshioka, N., S.Honiden, A.Finkelstein: Security Patterns: A Method for Constructing Secure and Efficient Inter-Company Coordination Systems. IEEE Int. Conf. on Enterprise Distributed Object Computing, 2004.