Chapter 7
Identifying Security Requirements and Privacy Concerns in Digital Health Applications

Gerd Stefan Brost and Mario Hoffmann

Abstract Security and privacy by design are important paradigms for establishing high protection levels in the eHealth domain. This means that security requirements and privacy concerns are considered and analyzed from the very beginning of any system design. For a reliable and robust system architecture and specification we recommend a four-step approach: (1) Decompose the system and identify the assets on the basis of the multilateral security concept, i.e., taking all participants of an eHealth scenario as potential attackers into account; (2) evaluate threats based on STRIDE for a holistic and systematic modelling of threats; (3) define use case-specific security requirements and privacy concerns as well as their relevance; and (4) mitigate threats by deciding what countermeasures should be implemented. After the introduction of each step this chapter illustrates the practical use in a step-by-step walkthrough with a real-world eHealth scenario and discusses advantages of security and privacy by design as well as its limitations.

7.1 Introduction

The healthcare sector has entered the digital age. Users of mobile healthcare devices and services no longer only monitor and analyze stress level, heartbeat, and blood glucose but also maintain detailed health diaries in the Cloud and share experiences with social communities—in March 2013 a study counted 97,000 mHealth applications [1]. Biosensors communicate with your watch, your watch with your smartphone, and your smartphone with healthcare services or communities [2]. Health cards have been introduced for storing individual diagnoses and medication and supporting not only administrative processes between patients, doctors, hospitals, pharmacies, and insurances but also emergency cases if allergies have been indicated on the card. Telemedicine and the exchange of large genome data sets take advantage of broadband connections and Cloud infrastructures. Healthcare has become an

G.S. Brost • M. Hoffmann (✉)
Fraunhofer AISEC, Munich, Germany
e-mail: gerd.brost@aisec.fraunhofer.de; mario.hoffmann@aisec.fraunhofer.de

S.A. Fricker et al. (eds.), *Requirements Engineering for Digital Health*,
DOI 10.1007/978-3-319-09798-5_7

important application domain of the Internet of Things and Services with a growing demand for research and development for example to support patient's self-management or provide Cloud-enabled platforms for individualized personal healthcare services.[1]

Information systems that are operated in a digital health context, however, have an intrinsically strong need for information security, since patient data is considered to be very sensitive in many contexts. This affects in total seven application categories according to [3] (1) education and awareness, (2) helpline, (3) diagnostic and treatment support, (4) communication and training for healthcare workers, (5) disease and epidemic outbreak tracking, (6) remote monitoring, and (7) remote data collection. For a detailed security evaluation in one of such application categories the parties involved as well as the assets to be protected have to be identified. Then typical threats, such as misuse, spy out, deny, misinform, divert, and tamper, can be applied and provide the common starting point to a detailed security evaluation.

User trust in an eHealth system is one of the central aspects [4]. Without sufficient trust, only few users are willing to enter their personal data into a system or use it for transmission and processing of their patient data. The effects of user's perception of system security and privacy concerns have been studied in [5] and have proven to be measurable. Security and privacy considerations must go hand in hand with the overall design. Only adequate security and privacy-enhancing technologies integrated in the design phase and implemented in systems can help establishing a certain level of trust.

Trust can be ephemeral and one security incident can cause the whole user base to mistrust a system. Once lost, it can be hard or impossible to gain it back.

Regarding this, security engineering has direct impact onto system design, since good security engineering is mandatory to create a trustworthy system. The security engineering process accompanies the system design and starts with defining security requirements. This can even affect the overall system functionality and user experience if strict security requirements have to be enforced. A late change in this kind of requirements can have an impact as hard as a late change in fundamental functional requirements.

Security experts, on the one hand, complain the fact that the analysis of implications to security and privacy still only follows—if any—the functional realization of eHealth solutions. For most parties in the healthcare ecosystem security requirements, protection goals, and threats remain abstract and risks of security incidents are not taken seriously into account. The investment in a security and privacy by design approach [6], on the other hand, can save money if properly implemented. Typically, in case of an incident reputation loss, recovering a compromised system, and integrating security mechanisms afterwards lead to much higher invests.

[1] In order to get an overview about EU-funded projects in FP6 and FP7 visit the EU PHS Foresight project (http://www.phsforesight.eu/). A good starting point for further search is the EC's website "eHealth and Ageing" (https://ec.europa.eu/digital-agenda/en/living-online/ehealth-and-ageing). Future European funding opportunities can be found at http://ec.europa.eu/research/participants/portal/desktop/en/home.html; example: H2020-PHC-2014-2015.

This chapter aims at providing a guideline for designers and practical security engineers of eHealth ecosystems as Sect. 7.2 explicates in detail. This includes how to conduct a threat analysis, define protection goals, and analyze your specific risks. Section 7.3 elaborates a concrete example of security engineering for a real-world digital health application. The discussion in Sect. 7.4 illustrates the limitations, costs, and benefits of a security engineering process, i.e., what can be achieved and what not. Finally, we conclude the key findings of this book chapter in Sect. 7.5 in order to highlight the arguments for security and privacy by design.

7.2 A Guideline for Practical Security Engineering

In order to structure the security engineering process we follow the holistic concept "multilateral security." Here, multilateral security means taking into consideration the security requirements and privacy concerns of all parties involved in the eHealth scenario that needs to be analyzed. It also means considering all involved parties as potential attackers. Not all parties are in favor of this approach but it includes the case that security incidents can be caused unintended as well as the case of malicious insiders. This is especially important for open communication systems, such as complex health ecosystems, as one cannot expect the various parties to trust each other [7].

In order to follow the multilateral security concept robust security design requires that the protection goals are made explicit [8]. They serve to protect assets and shape the security engineering process. So, in security engineering for a specific system it is a good starting point to get a thorough understanding of what protection goals are relevant for system design and which aspects need to be covered. They could be interpreted as a set of requirements for the security engineering process itself.

7.2.1 Security and Privacy Protection Goals

While designing an information system, it is useful to bear certain security goals in mind. As general building blocks of information security, three concepts are popular and serve as security goals when designing or evaluating information systems [9]:

Confidentiality: Confidentiality is the assurance that access controls are enforced and information is not disclosed to entities that they are not meant for. It is one of the core goals when dealing with sensitive information. Achieving confidentiality requires defining what information is to be considered as confidential, then to secure all exposed communication channels, and to store information in a secure way as well as other means to avoid leakage of this data. Without it, sensitive data would spread around.

Integrity: The concern when dealing with integrity is to make sure that data is protected from unauthorized modification or deletion. This can be expanded to an undo-functionality to revert illicit changes. It is important to achieve integrity to avoid these changes. Loosing integrity means making information in a system not to be trusted.

Availability: This goal aims to achieve continuous accessibility of relevant data and operation of the system. A system without sufficient availability will be neglected by users. Disrupting availability is a popular method to break system functionality (e.g., with denial-of-service attacks).

This core model is simple, robust, and served for many years. However, limitations became apparent. Ongoing discussion is concerned if and how to extend this model. Other properties or goals that are often used are [10]:

Authenticity: The property that information is authentic and coming from a person that is guaranteed to be the one it claims to be. Without authenticity, it is possible to have secret data transmissions, but it is not sure who that information comes from.

Non-repudiation: This is a property motivated by legal considerations. It implies one intention to fulfill their obligations to a contract. Not having non-repudiation makes it hard to fulfill certain legal standards. Users, e.g., could repudiate statements of will.

Another set of properties comes from recent discussion of privacy and is called privacy protection goals. Considering we are dealing with eHealth systems, privacy appears to be one of the most important goals. Important privacy goals are [11]:

Unlinkability: Unlinkability ensures that privacy-relevant data cannot be linked across privacy domains or be used for a different purpose than originally intended. This can be achieved by, e.g., early erasure, anonymization, or pseudonymization. Anonymization means removing identifying properties from data (e.g., removing the name of a person). Pseudonymization means the replacement of these properties with something less identifying, but reversible (e.g., replacing the name with a number).

Transparency: Transparency is one of the cornerstones of every modern privacy-oriented system and has found its way in some legislation. To achieve this, an adequate level of clarity of processes is necessary and brought to the user. This has also an important impact in user trust, since trust will hardly be achievable with a non-transparent system.

Intervenability: To achieve intervenability, data subjects and operators must be able to interfere with planned or ongoing privacy-related data processing.

The privacy goals are an important aspect of generating trust into a system, both on the technical and legal layer.

Protection goals in general are a direct reaction to threats. In order to understand and model these threats we recommend following the STRIDE[2] model. It has been

[2] http://msdn.microsoft.com/en-us/magazine/cc163519.aspx

defined by Microsoft in 2005 and has been established in many application domains already. The key aspects of STRIDE are:

S—Spoofing: "An example of identity spoofing is illegally accessing and then using another user's authentication information, such as username and password."

T—Tampering: "Data tampering involves the malicious modification of data. Examples include unauthorized changes made to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet."

R—Repudiation: "Repudiation threats are associated with users who deny performing an action without other parties having any way to prove otherwise—for example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations."

I—Information disclosure: "Information disclosure threats involve the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers."

D—Denial of service: "Denial of service (DoS) attacks deny service to valid users—for example, by making a Web server temporarily unavailable or unusable. You must protect against certain types of DoS threats simply to improve system availability and reliability."

E—Elevation of privilege: "In this type of threat, an unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats includes those situations in which an attacker has effectively penetrated all system defenses and become part of the trusted system itself, a dangerous situation indeed."

In order to illustrate STRIDE in the following subsections we decompose a simplified health care system into relevant components, analyze each component for susceptibility to the threats, and try to mitigate the threats. According to the STRIDE model, then, you need to repeat the process until you are comfortable with any remaining threats. Alternatively, a cost and risk analysis can help to quantify and qualify the remaining threats in order to take a decision whether all threats have to be mitigated. Note: There is no 100 % security.

7.2.2 Security Engineering Process

In order to deal with possible threats concerning a system and its assets, the following steps are taken:

Figure 7.1 gives an overview of the security engineering process. In step 1, the application that is analyzed is decomposed. Based on this decomposition, relevant assets for the threat evaluation are determined. In step 2, threats are determined that can affect the assets. Misuse cases can help to see the "other side" by changing the perspective to that of an attacker. Misuse cases can be used to gain a better

Fig. 7.1 Security engineering process

understanding of attack scenarios. In step 3, starting with the selection of security goals, appropriate security requirements are identified and described. The security requirements are the baseline for threat mitigation decisions in step 4. We will take a brief look on these steps, before demonstrating them on a real-world example in the next section.

7.2.2.1 Decompose System and Determine Assets

Since we deal with security engineering that accompanies the development process, we have access to software design documents like architecture diagrams (e.g., UML Component Diagrams and Collaboration Diagrams) or use case descriptions. These are then used to gain an understanding of the outer and inner workings of the system and help to identify assets and data flow patterns. Assets and data flow patterns are important for identifying protection mechanisms later on.

Assets are objects with direct and indirect value. A direct value, e.g., would be the monetary value of a server machine. An indirect value, e.g., would be the money that could be gained by selling patient data.

These assets can be furthermore divided into tangible assets (like Smart Cards, Desktop Computers or Servers) and intangible assets (like patient data or a PIN code). Tangible assets may be physically stolen or destroyed. Intangible assets may also be stolen by copying them or deleting them from disk.

Many risk assessment models require the quantification of an asset's value. This is often hard to measure. While the loss of a server machine is easy to quantify, the financial impact of losing reputation and user trust due to leaked personal information is quite difficult to put into numbers. When dealing with intangible assets, e.g., data, it is often more practical to identify channels and data sinks where sensitive data are transmitted and stored and secure those with adequate means.

7.2.2.2 Determine Threats

To determine treats do the identified assets; it helps to define misuse cases that accompany the system's relevant use cases.

Abuse/misuse cases [12,13] help to change the perspective by accompanying every use case of the system with appropriate misuse cases. So when a user is signing in at the system to access his data, an attacker might try to feign authentication

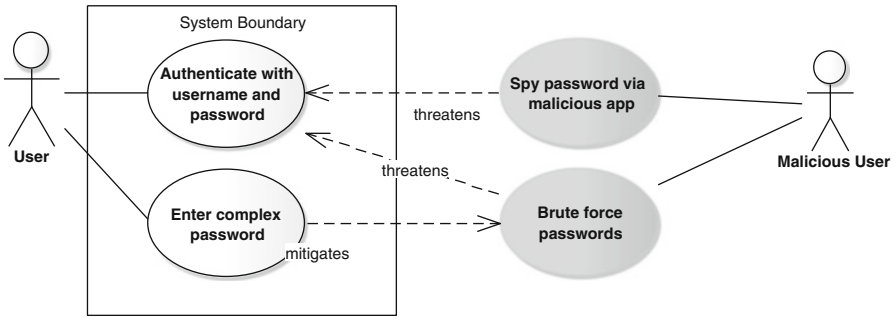

Fig. 7.2 Authentication misuse case diagram (*white*: use case, *grey*: misuse case)

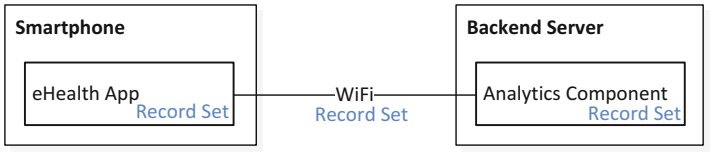

Fig. 7.3 Simplified system diagram

and access the data without proper rights to do so. It is helpful to reduce the usage to central misuse cases that do not descend into unnecessary design and architectural constraints. Figure 7.2 depicts a very simple misuse case diagram, where a malicious user attempts to brute force the user's password. This could be mitigated by choosing a complex password.

After getting a better understanding of the system (and the use cases and misuse cases), a threat analysis is performed to cover as many attack angles as possible so that a complete set of security requirements can be derived. To define threats for a system, it is necessary to choose an attacker model.

In formal protocol verification, the first step would be to choose an attacker model. The most prominent would be the Dolev-Yao attacker model [14]. The properties of these models are often a problem and have always been a point of discussion [15]. Choosing an attacker model is still a necessary prerequisite for threat analysis. This includes which channel the attacker can read and what tools he has. In practical work, choosing a powerful but realistic attacker produces the most benefit. To narrow the scope of the analysis, it often makes sense to exclude components that are not part of the current project. For example, the hospital backend already deals with highly sensitive information and might be excluded. This changes, however, when the usage of such a system is altered. An example would be if a closed data storage component is attached to the Internet and used for external communication.

As stated before, modeling data sinks and communication channels grounds on system decomposition and is the baseline for threat analysis.

Figure 7.3 shows a highly simplified overview of an eHealth App with a connection to the backend. It is a simplified version of a deployment diagram enriched with

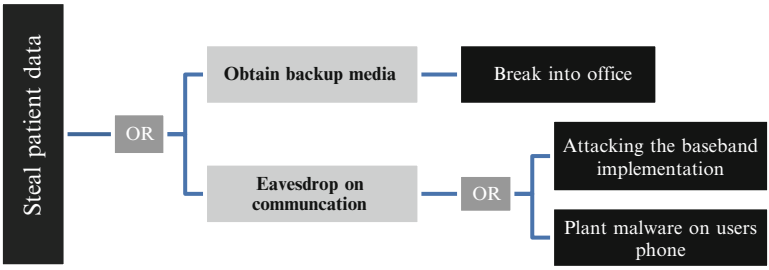

Fig. 7.4 Sample attack tree (*black*: root or leaf leading to sub-tree, *light grey*: node)

data that is stored and transmitted. Analyzing such architecture instantly makes clear that patient data is stored in the backend and on the user's smartphone as well as transmitted between those two. From this it is possible to derive attacks on the components and communication channels that aim to block, read out, or even modify communication.

To gain a better understanding of possible attacks on these assets and to avoid missing relevant attacks, the creation of attack trees is a common practice. Let us take a look at the example that an attacker wants to steal patient data. The root of each tree is the attack goal (e.g., steal patient data), where the nodes lead down to specific attacks (e.g., breaking a weak WLAN encryption to sniff user data).

In Fig. 7.4 we show an exemplary attack tree with possible sub-threats related to the outcome of stolen patient data. These attack trees can be broken into sub-trees. In this example, all three sub-threats need further detailing, resulting in a sub-tree for each. More detailed information about attack trees can be found in [16].

It is common practice to use formulae to calculate values for risks. This is a quantitative approach with a certain justifiable charm. Formulae to evaluate threats are mostly designed in a way similar as it is depicted in the following formulae:

$$\text{Risk} = \text{Likelihood} \times \text{Damage}$$

$$\text{Likelihood} = \frac{\text{Sophistication Level} + \text{Difficulty of Implementation}}{2}$$

$$\text{Damage} = \frac{\text{Financial Severity} + \text{Casual Severity} + \text{Privacy Loss}}{3}$$

Depending on the scenario, these formulae can get very complex and tend to produce rankings that are hard to verify or to understand.

Advantages and disadvantages for quantitative and qualitative approaches are shown in [17]. We will present a hands-on, qualitative approach (Table 7.1).

Table 7.1 Advantages and disadvantages of risk analysis methods

Quantitative methods	Qualitative methods
Advantages	
Applicability to all assets	Simple risk calculation
Mathematical foundation	Usefulness when asset value is irrelevant or unknowable
Using a management specific language (support cost benefit decision)	Less time consuming
Accuracy tends to increase over time as the organization builds historic record of data while gaining experience	Easier to involve people who are not experts on security or computers
Disadvantages	
Inappropriateness of monetary asset value	Coarse granularity
Inappropriateness of general statistics	Inability of cost benefit decision
Time consuming, requires much preliminary work	Subjective results, depend on quality of risk management team

Table 7.2 Threat evaluation matrix example

Attack 1: Steal user data by bruteforcing or spying on weak password			
Vulnerabilities exploited	Weak password chosen by user	Safety relevant?	No
		Component/system	Smartphone
		Attack type	Information disclosure
		Financial severity	Low
		Loss of privacy	Yes
Risk	6.00	Sophistication level	Low
Likelihood	3 (High)	Difficulty of implementation	Low
Resources required	Access to smartphone		
Attack scenario	Attacker either steals smartphone or accesses it while unattended. Weak passwords are tested or a simple pin that has been eavesdropped before is tried.		
Outcome	The attacker gains knowledge of user data.		

In a practical scenario, the security evaluation team must judge if it seems more sensible to evaluate each of the threats on its own and judge if this threat is going to be mitigated and how, or a more formal method is required using adapted formulae and threat matrixes. This choice, however, must always be made regarding the project and organizational situation.

To document the values and the qualitative aspects of an attack, such as technical details or setup requirements, threat evaluation matrixes can be used. Table 7.2 shows the evaluation of a threat resulting from a specific attack.

From this highly simplified example, we can learn that it is easy to gain access to the data a user has on his smartphone and a simple PIN protection might not be sufficient. Later on, a conclusion could be that we will need to implement mechanisms to keep user data safe.

7.2.2.3 Identify Security Requirements

The security goals we discussed are taken as a baseline for specific security requirements. It must be evaluated which of these security goals are suitable for the current system. Then requirements can be derived, by analyzing the goals and requirement categories. These requirements might be several or all of the following [18]:

- Identification requirements
- Authentication requirements
- Authorization requirements
- Immunity requirements
- Integrity requirements
- Intrusion detection requirements
- Non-repudiation requirements
- Privacy requirements
- Security auditing requirements
- Physical protection requirements
- System maintenance security requirements

We will briefly describe each requirement category. Identification means identification of entities (e.g., users or devices), whereas authentication is the process to confirm that identity. Authorization defines how a system specifies and grants access rights to resources. Immunity specifies the extent to which a system or component should protect itself from infections, e.g., from viruses. Intrusion detection covers means for a system to detect access or modifications by unauthorized entities (e.g., programs). Integrity, non-repudiation, and privacy directly relate to the security goals specified before. Security auditing means auditing status and use of security mechanisms. Physical protection defines protection against physical access, where system maintenance (in the security context) is concerned about avoiding maintenance operations colliding with security mechanisms.

To illustrate the identification of security requirements with an example: Confidentiality is the baseline for a privacy-related requirement that all patient data transmitted to the backend must be encrypted with a specific cypher (an algorithm to encrypt data).

7.2.2.4 Threat Mitigation

When the system is being designed in a way to mitigate a threat, it needs to be decided what countermeasures should be installed. Here, financial impact (cost), impact on usability, impact on performance, and threat severity need to be weighed against each other to make a choice. Security, usability, and performance are factors that influence each other. Many highly secure authentication mechanisms reduce usability (e.g., always carrying a smart cart and presenting it to the smartphone with a user password) or performance (e.g., a highly secure, but bandwidth consuming transfer protocol).

Documents with recommendations on what cryptographic methods, algorithms, and key lengths to use like [19], [20], or [21] are valuable tools for this task. These guidelines help decide about technical or organizational security means without having to be too familiar with all elements and the most recent attacks.

If the effort to mitigate a threat exceeds the possible budget or the usability decrease in doing so would be reduced as much to make the system unusable, it might be necessary to recommend alternative system designs that avoid that threat.

7.2.2.5 Summary

We discussed necessary steps for the security engineering process. The steps for verifying the security design like penetration testing are not part of this description. In the next section we will apply it on a realistic use case and discuss short examples of how these steps can be performed.

7.3 Example of Security Engineering for a Real-World Digital Health Application

To illustrate the security engineering process, we picked a real-world example for an eHealth system. We are discussing the diabetes share system, which has been designed as part of the FI-STAR project.

> The Diabetes Share System (DSS) is intended for patients, next-of-kin (e.g. relatives), physicians, and nurses who train, monitor, and consult an empowered Diabetes patient. DSS is a FI-STAR cloud solution that enables mobile recording of health and biometrical parameters, remote counselling, and comparison with other patients' anonymous observations. Unlike in-clinic treatment based upon manually recorded or lacking health parameters, DSS increases evidence to support treatments, increases the patient's knowledge base, assists in maintaining a healthy lifestyle, reduces the number of in-person appointments, and improves the patient's diabetes condition, wellbeing, and health. [22]

We will illustrate the security engineering process step by step with details taken from this example, following the steps briefly explained in the section before.

7.3.1 Decompose System and Determine Assets

In this step, we take design documents and architect input for the software design phase and try to decompose the system in a way that we can get an understanding of the assets and data flow in the system.

Figure 7.5 shows a component diagram from the DSS example. Patient data is transmitted over component boundaries and processed in remote locations. Diagrams

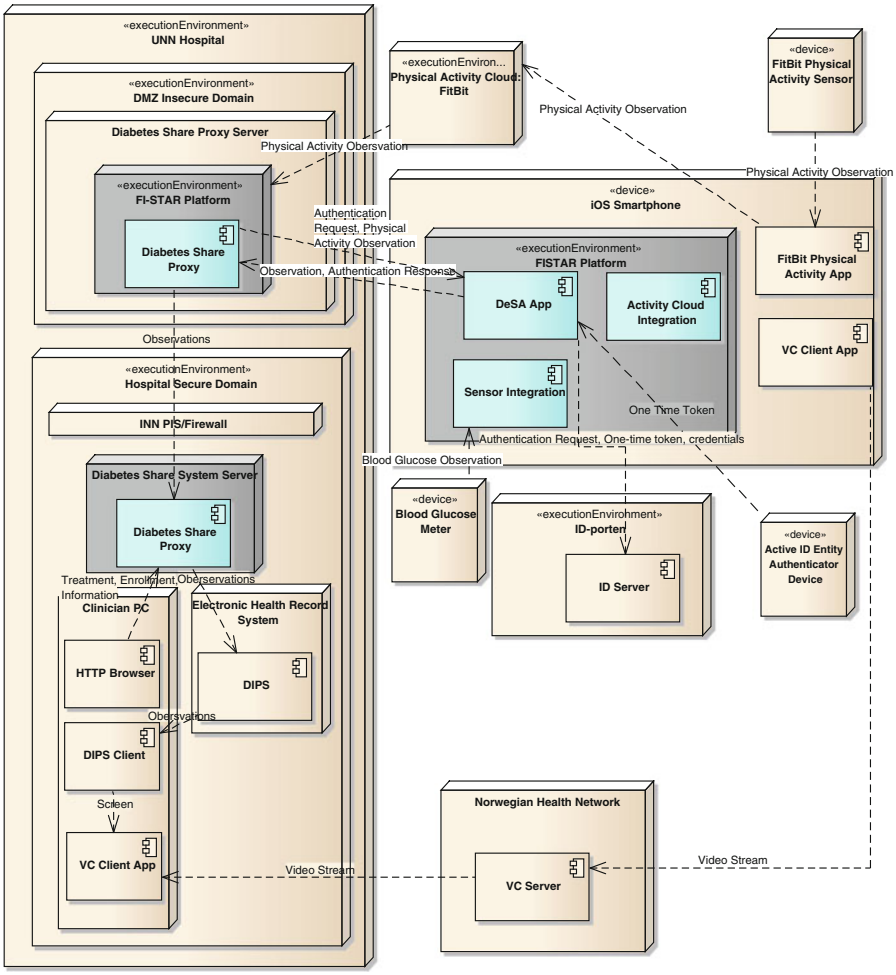

Fig. 7.5 Real-world example component diagram

like this one, coming from the design team, are very useful for determining potential assets and communication channels.

Several communication channels can be identified in the figure and the most relevant ones are listed in Table 7.3.

This list of communication channels and transmitted information is an important factor to gain knowledge about assets and potential threats. However, it is necessary to not only rely on certain views on the system design. This component overview, for example, does not reveal all information about the technical realization of the system since it is just a logical view.

Table 7.3 Communication channels

	Channel between	Assets
C01	Smartphone, FitBit Cloud	Physical activity observations
C02	FitBit Cloud, Diabetes Share Proxy Server	Physical activity observations
C03	Diabetes Share Proxy Server, Smartphone	Physical activity observations
C04	Smartphone, Diabetes Share Proxy Server	Observations, authentication data
C05	Smartphone, VC Server	Video stream data
C06	VC Server, Clinician PC	Video stream data
C07	Smartphone, ID-Porten-Server	Credentials
C08	Blood Glucose Meter, Smartphone	Blood glucose observations
C09	Physical Activity Sensor, Smartphone	Physical activity observations
C10	Diabetes Share Proxy Server, Diabetes Share System Server	Observations
C11	Clinician PC, Diabetes Share System Server	Treatments, enrollment information
C12	DSS Server, Electronic Health Record System	Observations
C13	Electronic Health Record System, Clinician PC	Observations

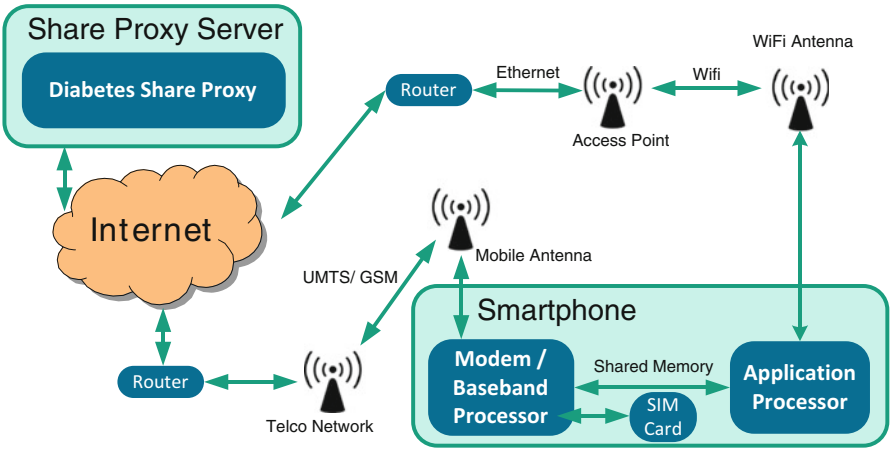

Fig. 7.6 Detailed communication scenario

Figure 7.6 is a refinement of Fig. 7.5 according to the dimension of communication network topology.

If we take a look on the communication over a smartphone, we can see that communication can be achieved over different channels and different networks. When we rely on the encryption of point-to-point-connections, it is hard to achieve overall security.

Figure 7.6 shows a more technical view on the smartphone's communication channels. Perspective is changed from a logical component or deployment diagram to a real technical decomposition. Depending on how the smartphone transmits its data (WiFi or mobile network data connections), several networks are involved.

Table 7.4 Assets

	Asset	Locations	Sensitive?
A01	Observations	Smartphone, DSS Proxy Server, DSS Server, Electronic Health Record System, Clinician PC	Yes
A02	Authentication data	Smartphone, Diabetes Share Proxy Server	Yes
A03	Video stream data	Smartphone, VC Server, Clinician PC	Yes
A04	Credentials	ID-Porten Server, Smartphone	Yes
A05	Blood glucose observations	Blood Glucose Meter, Smartphone	Yes
A06	Physical activity observations	Physical Activity Sensor, Smartphone	Yes
A07	Treatment plan	Clinician PC, DSS Server	Yes

Data leaves the phone, passes into wireless or mobile networks, and is routed into the Internet, and delivered to the hospital infrastructure. The smartphone itself is not a single component but a system of components, which can be manipulated and internal communication channels that could be eavesdropped on. It is hardly possible to take all of these details into account, but it is important to be aware of the complexity and take reasonable decisions what areas to cover. In this case it would be reasonable to care for the security of the application and its data flows, but not drilling down into securing the smartphone itself. If a project relies on components and systems that are considered to be secure, these decisions must be documented to make the rationale traceable for others.

The same is true for certain communication flows embedded in protocols. A typical example is authentication protocols. When a standard authentication protocol is used and the used protocol is considered to be secure, we can exclude transmitted data (e.g., credentials) from our asset and channel list. This decision, however, also needs to be documented.

Determining assets and analyzing data flows over the communication channels in Table 7.3 can be helpful, but is not sufficient. Data is often at rest or not transmitted at all, but still represents an asset that needs to be protected. So user data that resides on the smartphone and is not transmitted must also be protected by adequate means (e.g., by a password policy).

To identify and evaluate threats to assets, the value of those assets must be determined. A quantitative approach is often difficult, since exact amounts of damage done by data leakage and the damage resulting from bad system reputation and lost user trust are hard to measure or define. A qualitative approach can help to ease the decision what assets to protect and what level of security is needed. For this simplified example, we will just define what assets are to be considered sensitive. We will only take into account intangible assets and indirect values here, since tangible assets are either in the responsibility of the patient (smartphone) or form a component in an existing server infrastructure which is not part of this example. In this example, we consider all assets in "Table 7.4: Assets" as sensitive.

7.3.2 *Determine and Evaluate Threats*

Since we know what assets are present in our system and need to be protected, we can get an understanding of what approaches could be interesting for a malicious user (attacker) to use the system inappropriately.

In Fig. 7.7, use cases from the system design documents are taken and matched with misuse cases to help threat determination.

Building on misuse cases and the evaluated communication channels and assets, we can perform a classical threat evaluation, starting with attack trees. These trees provide the means for an effective and systematic approach to cover as many relevant attacks as possible. Figure 7.8 depicts a high-level attack tree, where the leaves define sub-trees. An example for that can be seen in Fig. 7.9. The attacks can be broken down until the desired granularity level is reached.

This breakdown of attacks can be performed to a level as detailed as a certain known attack, e.g., on a cryptographic algorithm. In most projects, this might not be necessary. It is sufficient in most cases that certain transmission type cannot be fully trusted. As in this example it becomes clear that there are attacks on wireless data transmission which would be a valuable input for security requirements engineering and later system design, e.g., advocating the need for end-to-end encryption of data later on.

When all relevant attacks are covered, they can be evaluated with a threat matrix. When using such a matrix, it is important to adapt it to the specific project's needs.

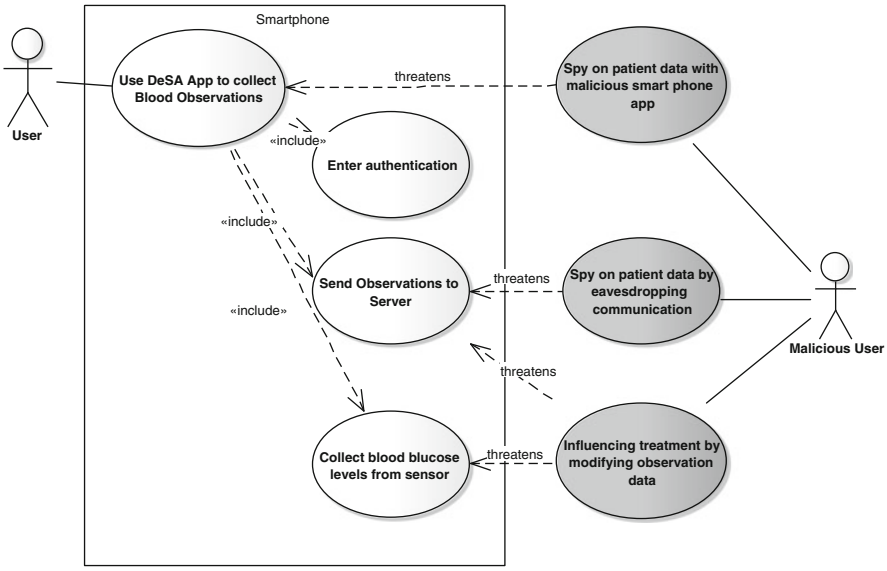

Fig. 7.7 Sample misuse case diagram without mitigation

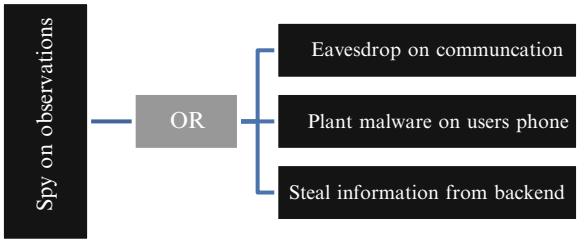

Fig. 7.8 Sample high-level attack tree

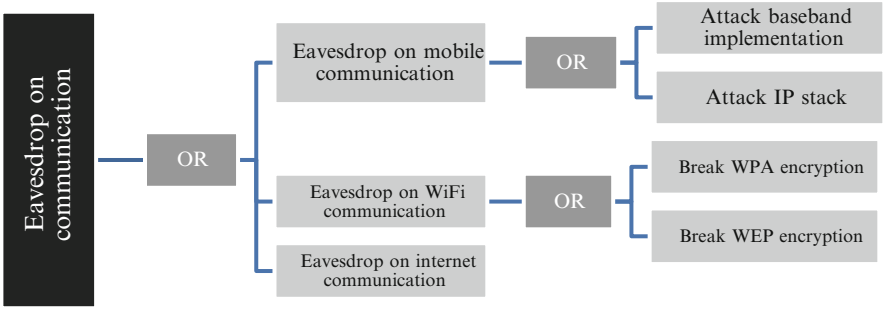

Fig. 7.9 Attack sub-tree

As discussed in the section before, if the formulae to calculate the risk are getting too complex, the value of the assessment becomes questionable.

Table 7.5 shows the threat evaluation for an attack on the baseband implementation of a mobile handset to eavesdrop on transmitted data.

7.3.3 Identify Security Requirements

Since we now have a good understanding of possible attacks and of the assets that are worthy of protection, we can identify relevant security requirements for our system. It is helpful to keep in mind that security requirements do not include architectural decisions for the software system. Although it is useful to state a certain level of protection for system access, the concrete type of access mechanisms is supposed to be left open for further investigations. This should be delegated to the architect and the decisions should be negotiated together. Such a negotiation process is described in [23].

As discussed before in the security engineering process description, there are a number of categories for security requirements like identification, authentication, and privacy requirements. Let us specify a few requirements that are an outcome of the threat evaluation. In this threat evaluation we realized that data could be read out by an attacker. This leads to the following privacy requirements for the DeSA application:

Table 7.5 Attack example: Eavesdropping on communication

<table>
<tr><td colspan="4">Attack 1: Steal patient data by eavesdropping on wireless communication by attacking the baseband implementation</td></tr>
<tr><td rowspan="5">Vulnerabilities exploited</td><td rowspan="5">Most encryption algorithms used in GSM and GPRS have weaknesses. Only the recent A5/3 algorithm can be considered as relatively secure. Regardless of the encryption algorithm GSM does not have mutual authentication and an attacker can launch a rogue base station attack.</td><td>Safety relevant?</td><td>No</td></tr>
<tr><td>Component/system</td><td>Smartphone</td></tr>
<tr><td>Attack type</td><td>Information Disclosure</td></tr>
<tr><td>Financial severity</td><td>Medium</td></tr>
<tr><td>Loss of privacy</td><td>Yes</td></tr>
<tr><td>Risk</td><td>2.67</td><td>Sophistication level</td><td>Medium</td></tr>
<tr><td>Likelihood</td><td>2 (Medium)</td><td>Difficulty of Implementation</td><td>Medium</td></tr>
<tr><td>Resources required</td><td colspan="3">Standard Laptop, GSM/GPRS Base Station (Hardware and Software), frequency jammer</td></tr>
<tr><td>Attack scenario</td><td colspan="3">It is possible to eavesdrop on communication via GSM/GPRS or UMTS/LTE. Since GSM/GPRS does not have mutual authentication a rogue base station can be employed in order to attack the ATM. Either the traffic is captured directly or encryption can disabled for this GSM/GPRS communication channel. Alternatively the attacker can try to break the encryption of a GSM or GPRS channel set up by his victim and another operator.
In the UMTS scenario, jamming is required to force a fallback to GSM and use the approach described above. Additionally, there are the following attack vectors:
• The RRC protocols of UMTS and LTE is spoken before authentication
• The attacker could try to break into a femtocell supplied by a provider</td></tr>
<tr><td>Outcome</td><td colspan="3">The attacker gains knowledge of any additional unprotected communication.</td></tr>
</table>

Since we are facing communication flows over several technological domains and media, we need end-to-end security, as stated in R2. Security requirements are seldom independent from each other. When securing connections to transmit data in private, other requirements, as for identification, can be derived as in R3. Identification requirements are often not sufficient for themselves, since authentication bases on identification, as derived in R4. Also authentication requirements are most often not sufficient for themselves but need to be accompanied by authorization requirements as in R5 or R6.

When the security requirements are derived for every component, this is the baseline for the security design of the overall system. Keep in mind that until now no decisions have been made how the technical realization will look like. These decisions can now be made by the architect in consultation with the requirements engineer to shape the solution.

7.3.4 Threat Mitigation

After the security requirements have been defined, a security concept for the overall system can be created. This concept will map requirements to specific means like authentication tokens, encryption algorithms, and access control methods. We give a short outlook on how threat mitigation could be realized by countering relevant threats with technical means.

Since the evaluation of threats led to the definition of security requirements, we need to realize the system in a way that the requirements from Table 7.6 are fulfilled.

R1 and R2 can be fulfilled by not relying on inherent security mechanisms of the underlying technology (e.g., UMTS or WiFi encryption), but to provide our own end-to-end security. This could be achieved by choosing TLS (recommended is version 1.2) and proper authentication and key exchange methods. This, however, is more complicated than it seems. The selection and configuration of cipher suites is a complex topic that should only be taken care of by experienced experts in that field. Slight misconfigurations could result in a complete loss of confidentiality. For example, AES is a very-well-known symmetric encryption algorithm.[3] By choosing the wrong mode of operation or a weak generation of random numbers for the key, AES can be made totally insecure. So, experience in the field of cryptography and its application is needed to construct a secure system.

R3 requires authentication of communication partners. This could be realized by issuing soft tokens that are compliant with PKCS#11[4] (Public Key Cryptographic Standard). This, however, requires the presence of a Public Key Infrastructure (PKI). In the scenario we discussed in Fig. 7.5 id-porten is used. This is a Norwegian

Table 7.6 Derived security requirements

Requirement ID	Short description
R1	The DeSA application shall not allow unauthorized individuals or programs access to transmitted data that flows between components.
R2	Communication that flows over several technological barriers or components must be secured end to end.
R3	The DeSA application must identify other valid communication partners before transmitting data.
R4	The DeSA application must verify the identity of each communication partner before transmitting data.
R5	The DeSA application shall allow the successfully authenticated Diabetes Share Proxy access to Observations the user authorized for transmission.
R6	The DeSA application shall not allow any other entity to access Observations.

[3] http://www.ijcset.net/docs/Volumes/volume1issue3/ijcset2011010306.pdf

[4] http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-11-cryptographic-token-interface-standard.htm

identification portal[5] with different authentication levels. R4 mandates that these identification tokens are not only available but also actually verified in the software. R5 and R6 require the selection of an access control model (e.g., mandatory access control) and a definition of access rights. This could be realized by choosing RBAC (role-based access control).

These decisions are complex and entail many details that need to be defined. Fortunately these tasks are similar in many different projects and a lot of guidelines and standards are available as, e.g., [20] or [21].

Some technological solutions are highly secure, but would ruin user acceptance. The use of smart cards is well understood and could be realized with modern smartphones that could connect to the smart card wirelessly over NFC (near-field communication). The usability drops dramatically, though. This is where user acceptance and usability engineering come into play to determine a realistic solution. One example would be the design of a mechanism that requires the presentation of the smart card only for specific operations. If accessed normally, the user is only required to enter a PIN.

For other components, it is not possible to actively influence the security levels. For user smartphones, for example, certain risks will always remain (e.g., malware) and maybe made worse through rooting the phone through the user. In this case, all we can do is to inform the user (relating to transparency) as best as possible. The same is true for the identification service. We can only decide if we want to trust id-porten or not. We can base our decision on security evaluation reports and certifications to make it justifiable, however.

7.4 Discussion

We gave an overview of a security engineering process to define and elicit security requirements.

Another important component to achieve security of a system is a security evaluation of the system and the verification of security requirements. Where other nonfunctional requirements as performance can be verified by load tests, it is possible to implement security tests to verify security requirements. This should be accompanied by specific penetration testing efforts. Penetration testing reverses the role of an engineer to that of an attacker. This is comparable to the design of misuse cases, where the user role is converted into a misuser or attacker role. Security must also be a part of managing change requests, since modifications of the source code without a specific focus on security can break security mechanisms easily.

We presented a comprehensive process for security engineering in a software system with distributed components, which is hands-on and inspired by practical experiences.

[5] http://eid.difi.no/nb/id-porten

However in reality, the security engineering process is not as linear as depicted. It is more an iterative approach where some requirements are detailed early in the project and some other relate to technical or organizational framework conditions that come up later in the project. So every process step has feedback loops with the other steps and the requirements need some time to stabilize. Some aspects of the process steps need to be customized for each project.

One huge driver of project effort can come from threat evaluation. The methodology to evaluate threats and represent these with hard numbers or at least qualitative values can grow very complex, making it hard to maintain and to verify. Complex formulae tend to have an esoterical touch for other project participants and make traceability hard, as well as the justification for technological measures, which directly increase cost. Reducing the complexity to a purely qualitative evaluation can make it also harder to justify decisions since for external readers, the results may appear less founded. Documentation is a key aspect to not loose important information on the way.

This evaluation, as well as the threat assessment matrices, needs to be customized to the project needs. There might be different needs for safety or financial losses and this should be reflected in the evaluation. Also the level of complexity for the threat evaluation formulae needs to be adjusted, where some regulatory or legislative requirements can demand a certain level of detail.

Another aspect that needs to be defined is the definition of protection levels. It would be possible to define certain levels and attach them to technological and organizational means. In simpler projects, it might be sufficient to define one level for the whole system.

An issue that is still not solved in a satisfactory manner is how to find the right trade-offs between security, usability, and performance. Each goal can be achieved on its own. Balanced combinations are subject for further research.

Regarding threat evaluations, there are multiple models and numerous variations of the exact process (e.g., remember the discussion of quantitative and qualitative approaches). It would be very helpful to evolve towards a framework where such decisions are covered and are made easier for security engineers.

7.5 Conclusion

This chapter has introduced security and privacy by design as a driving paradigm to realize reliable and trustworthy eHealth systems. In order to break down this paradigm to concrete security methods and technologies for use cases in eHealth a comprehensive requirements engineering process has been illustrated and applied. The conceptual basis is called “multilateral security” where all participants of a specific use case, e.g., patients, physicians, and third-party service providers, are considered as potential attackers. This is important for taking any kind of possible attack vector into account—no matter an attack is intended or by accident. Single attack vectors can then be reflected according to use case-depending protection goals. Typical

security goals are confidentiality, availability, and integrity; we recommend, however, to consider additional privacy goals such as unlinkability, transparency, and intervenability in order to specify a well-balanced system design.

The threat analysis that we have proposed supports designers and developers of eHealth systems to meet all of their protection goals. Following the STRIDE model we illustrated four important phases: (1) decompose system and determine assets, (2) determine and evaluate threats, (3) identify security requirements, and (4) mitigate threats. How these phases should be applied from our point of view is described step by step in a concrete example of a diabetes share system, which has been designed as part of the FI-STAR project.

From the discussion we, finally, derived that the engineering of security requirements is not a job for an isolated team, but requires constant communication between system architects and security engineers. The security engineering process accompanies the whole system development. Analyzing the security requirements and privacy concerns in the design phase, thus, is just one but important step in the complete security development life cycle.

References

1. Research2Guidance (2013). Accessed on 1 May, 2014, from http://research2guidance.com/the-market-for-mhealth-app-services-will-reach-26-billion-by-2017/
2. Lymberis A, De Rossi DE (2004) Wearable eHealth systems for personalised health management: state of the art and future challenges. Ios Press, Amsterdam
3. Vital Wave Consulting (2009) mHealth for development: the opportunity of mobile technology for healthcare in the developing world. United Nations Foundation, Washington, DC
4. Ruotsalainen P, Blobel B, Seppälä A (2012) A conceptual framework and principles for trusted pervasive health. J Med Internet Res 14:e52
5. Hsu C-L, Lee M-RS-H (2013) The role of privacy protection in healthcare informatipn systems adoption. J Med Syst 37(5):1–12
6. Cavoukian A, Chanliau M (2013) Privacy and security by design: a convergence of paradigms
7. Müller G, Rannenberg K (2004) Multilateral security in communications. Addison-Wesley, Reading, MA
8. Anderson RJ (2008) Security engineering: a guide to building dependable distributed systems. Wiley, New York, NY
9. McCumber J (1991) Information systems security: a comprehensive model. Proceedings of the 14th national computer security conference, Washington, DC, 1–4 October 1991
10. Zhou J, Gollmann D (1996) Observations on non-repudiation. In: Advances in cryptology. Springer, New York, NY
11. Hansen M, Probst T (2012). www.datenschutzzentrum.de. Accessed on May 2014, from https://www.datenschutzzentrum.de/guetesiegel/Privacy_Protection_Goals_in_privacy_and_data_protection_evaluations_V05_20120713.pdf
12. McDermott J, Fox C (1999) Using abuse case models for security requirements analysis. 15th Annual computer security applications conference. Phoenix, Arizona
13. Sindre G, Opdahl A (2005) Eliciting security requirements with misuse cases. Requir Eng 10:34–44
14. Dolev D, Yao A (1983) On the security of public key protocols. In: Information theory. IEEE, Washington, DC, pp 198–208

15. Roscoe, B. (2003). The attacker in ubiquitous computing environments: formalising the threat model. Proc. of the 1st Intl workshop on formal aspects in security and trust, Italy
16. Schneier B (1999) Attack trees. Dobb J 24:21–29
17. Tatat Ü, Karabacak B (2012) An hierarchical asset valuation method for information security risk analysis. International conference on information society, 25–28 June 2012, London, pp 286–291.
18. Firesmith DG (2003) Engineering security requirements. J Obj Tech 2:53–68
19. BSI (2014) Kryptographische Verfahren: Empfehlungen und Schlüssellängen.
20. ENISA (2013) Algorithms, key size and parameters report
21. Barker E, Roginsky A (2011) Transitions: recommendation for the use of cryptographic algorithms and key lenghts. NIST, Gaithersburg, MD
22. Fricker S, Thümmler C (2014) Technical requirements and architecture report including open call requirements. Technical Report. Accessed on 1 May, 2014, from https://bscw.fi-star.eu/pub/bscw.cgi/d42532/D1.1_R1.pdf
23. Fricker S, Gorschek T, Byman C, Schmidle A (2010) Handshaking with implementation proposals: Negotiating requirements understanding. *IEEE Software* 27(2):72–80