Bundesamt
für Sicherheit in der
Informationstechnik

# Security requirements for eHealth applications

## Technical Guideline BSI TR-03161

Trial Use[1]

1 Explanation in section 1.4

# Revision history

| Version | Datum | Name | Description |
|---|---|---|---|
| 1.0 | 15 April 2020 | Kleinmanns (DI 24) | First version |
| | | | |
| | | | |

Federal Office for Information Security
P.O. Box 20 03 63
53133 Bonn, Germany
Email: referat-di24@bsi.bund.de
Internet: https://www.bsi.bund.de

# Table of Contents

# 1 Introduction

## 1.1 Subject of the Technical Guideline

According to § 33a of the German Social Security Code, Volume Five (SGB V, *Sozialgesetzbuch (SGB) Fünftes Buch (V)*), persons with statutory health insurance are under certain conditions entitled to be provided with so-called eHealth applications. These applications are designed to support the "detection, monitoring, treatment or alleviation of diseases or the detection, treatment, alleviation or compensation of injuries or disabilities" [SGB V section 33a]. This Technical Guideline is addressed to manufacturers of eHealth applications for mobile devices. Furthermore, it can be used as a guideline for mobile applications that process and store sensitive data.

## 1.2 Objective of the Technical Guideline

Digitalisation of all areas of life, whether at work, in home environments, in individual or public transport, is progressing steadily. In 2018, the number of Internet users already exceeded the four billion mark. Two thirds of the world's current population of 7.6 billion use a smartphone. More than three billion people use social networks, and nine tenths of them do so on their smartphone (see [GDR18]). This development in the health care sector is continuing with the trend towards 'self-tracking', but also with the increasing demand for efficient use of medical data once it has been collected. Especially in the health care sector, it is convenient to be able to access own medical records independent of time and place. In this case, mobile applications store sensitive and personal data, from heart rate and rhythm of sleep records to medication plans as well as medical prescriptions and certificates. They connect the user to the corresponding services and act as communication hubs. A compromised smartphone can hence unintentionally expose the user's entire digital life. Compliance with adequate security standards, especially in mobile applications, can make this considerably more difficult and possibly even prevent it. Already during the development phase, manufacturers should meticulously plan how a mobile application will process, store and protect personal and other sensitive data.

IT security essentially pursues three protection goals: Confidentiality, integrity and availability.

Compliance with these requirements is particularly important for mobile health applications. In contrast to the financial sector, where fraudulently transferred money can be refunded to customers by the banks, the confidentiality of health data that is unwillingly disclosed will be lost once and for all. Although the patient could receive compensation for this, the disclosure cannot be undone. Furthermore, unintentional disclosure of health data, both in the social and professional environment, can lead to undesirable consequences with significant impacts.

Should an attacker be able to manipulate the health data of a third party and thereby violate its integrity, this could have a significant impact on treatment decisions and ultimately the individual's health.

This Technical Guideline is designed to assist developers of eHealth applications in developing secure mobile applications.

## 1.3 Overview of the Technical Guideline

### 1.3.1 Methodology

The term *application* as used herein refers to a mobile eHealth application within the meaning of § 33a SGV V (see [SGBV33a]). Operation can be performed autonomously with a *mobile application* on the mobile device or in combination with a secure *backend*. The term *backend* as used herein also includes cloud-computing platforms. Due to rapid technical progression and the diversity of mobile devices and their platforms, this Technical Guideline does not claim to be complete. It can be considered to constitute a minimum requirement for the secure operation of an application.

This Technical Guideline includes a Security Problem Definition (SPD), which identifies potential threat scenarios. Objectives for mobile applications and their platforms and/or deployment environments are derived from the SPD in order to ward off threats.

The threat scenarios and objectives in this Technical Guideline are based on the experience that BSI has gained in previous investigations. In addition, they are based on international standards, such as the 'Smartphone Secure Development Guidelines' [SSDG] and the 'Mobile AppSec Verification Standard' [MASVS] with the associated 'Mobile Security Testing Guide' [MSTG].

One basic requirement for eHealth applications is the orientation towards best practice recommendations and other general requirements for secure, distributed applications. This includes intensive functional and integration testing and, in particular, positive/negative testing of the application's security features. The Technical Guideline also identifies additional, specific requirements.

### 1.3.2 Terminology

The terms used in this Technical Guideline have the following meanings:

| | |
|---|---|
| MUST | The manufacturer must implement a certain property as a mandatory requirement. |
| MUST NOT | The application/backend must not under any circumstances whatsoever have a certain property. |
| SHOULD | The application/backend must have a certain property, unless it is shown that failure to transpose does not pose a risk to secure operation or implementation is currently not possible due to technical limitations. |
| CAN | The application/backend may have a certain property whereby a conversion of this property must be indicated by the solution provider. |

## 1.4 Open points

The Technical Guideline has 'trial use' status. This means that safety targets have been defined, but experience with the application of the objectives is yet to be gained. During the 'trial use' phase, attention focuses on feedback from industry.

This could lead to changes within individual objectives. It is also conceivable that objectives may be added or deleted.

Future versions will be supplemented by chapters to enable testing and certification according to this Technical Guideline.

# 2 Overview of security requirements for mobile applications

## 2.1 Application concepts on mobile devices

The term 'mobile application' refers to a program that runs on a mobile platform. Such applications can be generally divided into three categories. The first category consists of native applications (chapter 2.1.1) which are directly tailored to the platform on which they are executed. Web applications (chapter 2.1.2) are their counterpart. Their implementation is completely platform-independent and they are executed in the web browser of the mobile device. Hybrid approaches fall into the third category (chapter 2.1.3). They reflect all possible combinations of native and web applications.

Since web applications go far beyond their general use on mobile devices, this Technical Guideline focuses on native applications and the native part of hybrid approaches. For additional information regarding the secure development and operation of web applications, BSI recommends consulting further literature. The 'OWASP Application Security Verification Standard' [ASVS] offers a good introduction to this topic.

### 2.1.1 Native applications

A native application is tailored to a platform and its operating system. It is based on programming tools (Software Development Kits – SDKs) provided by the platform (such as Android or iOS). They provide direct access to device components, such as the GPS function, the camera or the microphone. Due to their proximity to the operating system, these components can achieve very good performance, high reliability and intuitive operation. The applications are, for instance, installed from the platform's own app store and can often also be operated offline.

However, there are also disadvantages associated with the proximity to the operating system. Changes to the operating system, for example through updates, can result in the need of adaptation of the application in order to keep its functionality. Furthermore, it cannot be installed on other operating systems. If the same application is to be published on multiple operating systems, a separate code base[2] must exist for each. This is often a complex and very costly exercise.

### 2.1.2 Web applications

Web applications are websites that are designed to look and behave like a native application. Unlike native applications, they are not based on an SDK of the underlying platforms, but on classic programming tools used for web development, with HTML5 and JavaScript being commonly used ones. They therefore only allow very limited access to device components.

2 Other approaches are *cross-platform* implementation concepts that support the simultaneous development of an application for different platforms. However, this merely shifts dependency into this very complex middleware, which must cover all target platforms.

However, their biggest advantage is that they are independent of the operating system. Since the applications are executed in a web browser, they can be equally used on any platform without any adjustments required at code base level.

### 2.1.3 Hybrid approaches

Hybrid applications combine both the advantages and disadvantages of native and web applications. The SDK is used to create a frame application which has all the advantages and disadvantages of native applications. It can access device components and it is available from an app store, but cannot be installed on other platforms without changing the source code. The frame application additionally includes an embedded web browser (called WebView) that is used to integrate web applications into native applications. This allows web applications to also access the device components. Otherwise, they would be reserved for native applications. However, this means that performance and stability losses must be accepted. Furthermore, the use of different user interfaces can adversely affect user experience. The platform dependency of the application now only applies to the frame application, so that migration to other platforms becomes significantly easier.

## 2.2 Backend services

Most applications do not rely solely on the resources provided by the runtime environment for processing and storing data, but rather shift these tasks to a central background system (backend). These systems not only process and store data, but often also perform tasks for authenticating and authorising users or other centralised activities. This means that not all functionalities of the application must be implemented on mobile devices, but are instead often limited to a graphical user interface (frontend). It is not possible to make a general statement as to how much functionality should be implemented in the application itself and how much is shifted to a server. These characteristics can vary from application to application.

An active Internet connection is mandatory for the use of applications connected to a backend. Communication between frontend and backend is usually routed via a HTTPS secured connection. The use of backend systems is not limited to mobile applications, but instead reflects the current state of the art for almost all applications. Since this Technical Guideline focuses on mobile applications, the following threat analysis and objectives refer to the protection of the application and additionally only to those features of the backend that directly impact the security of the application. For more detailed information on the secure operation and development of backend systems, we recommend to consult further literature. The ‘Top 10 Web Application Security Risks’ of the OWASP Foundation [T10WASR] offer a good introduction. Cloud computing users are advised to consult the BSI’s ‘Cloud Computing Compliance Criteria Catalogue’ [KCC-C5].

## 2.3 Security Problem Definition

The Security Problem Definition describes assumptions, threats and organisational security policies which are relevant to eHealth applications where security is concerned.

### 2.3.1 Assumptions

| | |
|---|---|
| A.Device | The platform on which the application is used is operated and protected against vulnerabilities by the user, for example, by updating the operating system after the respective updates have been made available. Its security was not compromised by the deliberate execution of 'roots' or 'jailbreaks'[3]. |

### 2.3.2 Threats

| | |
|---|---|
| T.AssetLocal | An unauthorised person gains access to sensitive data of the application, such as unencrypted data stored in the file system or RAM. This also includes a situation where an attacker can access encrypted, sensitive data in plain text after analysing the encryption mechanism. |
| T.AssetRemote | On the backend, unauthorised access to user and application assets (such as entropy for operational TLS connection) is possible. This is carried out, for example, through misuse of the mobile application or through a vulnerability in the implementation of the backend interface in the direction of the frontend. |
| T.Auth | An attacker gains access to sensitive data of other users under a false user ID or by using a false role or group affiliation. |
| T.Eavesdropping | An attacker succeeds in eavesdropping in on or interfering with communication of the application via an insufficiently encrypted/non-authenticated connection. A transport connection to an unauthorised component is established, for example, due to a lack of verification of certificate properties. |
| T.Expense | The application causes unforeseen, additional costs for the user. |
| T.Integrity | An attacker can manipulate data in a memory or in transit without being detected. |
| T.Guessing | An attacker could guess a password by trial and error and thereby gain unauthorised access to another user's sensitive data. |
| T.MemoryStructures | An attacker performs reverse engineering on the application, thereby discovering unprotected data structures in the memory, so that keys and sensitive data can be accessed. |

3 Softening of the operating system's inherent security functionalities by obtaining higher-level access rights and enabling the installation of applications from unknown sources.

### 2.3.3 Organisational security policies

| | |
|---|---|
| OSP.Authorization | The manufacturer develops an authorisation concept that controls both read and write access to sensitive data. The access permissions must be selected in such a way that only those rights are granted that are necessary to fulfil the primary purpose. The authorisation concept must be implemented independently of authentication. |
| OSP.BackendConnLog | Information concerning all outgoing connections is collected on the backend for post-mortem analyses of security incidents, including meta-information about proxies used and server certificates verified. |
| OSP.CriticalUpdates | The manufacturer permanently checks and monitors the application and its backend as well as the frameworks and libraries[4] used for exploitable vulnerabilities. The manufacturer must provide an update at short notice if exploitable vulnerabilities are identified. The backend must inform the application about the update and, after a grace period, stop using the application with an outdated application. |
| OSP.LibsIn | Data received from third-party libraries should be validated before being used in the application (for instance, XML schema validation, check for invalid encoding, etc.). The aim is to protect the application from attacks by malicious input. |
| OSP.LibsOut | The application should not send sensitive data in plain text to third-party frameworks and/or libraries. The use of suitable frameworks and/or libraries for protecting a communication channel or a local memory container is permitted. |
| OSP.RNG | Random numbers must be obtained from a random number generator with high entropy. The application should initially introduce entropy from the user into the random number generator of the platform. Thereafter, the application obtains random numbers from the backend and feeds them into the local random number generator. |
| OSP.Purpose | Any data collection, processing, storage and transfer may only be carried out for a limited purpose. To this effect, the manufacturer publishes the primary purpose of the application and in addition which data is processed and how, where and how long it is stored. The permissible communication behaviour as well as the internal and external sensors used must be selected in line with the primary purpose.<br><br>Application note/example: Tracking data, such as Wi-Fi-SSID, GPS, etc., may only be used for a limited purpose and in accordance with the principle of data minimisation. Direct or indirect persistence of such data in the device (for instance, in image recordings) is not permitted unless this is directly required by the intended use. |

4 A third-party framework and/or a third-party library can be understood as a container of functionalities that has not been created under full control of the developer of the application and that is also not part of the functionality of the operating system platform used.

### 2.3.4 Residual risks

The operation of eHealth applications is subject to particularly demanding requirements, which cannot be sufficiently met by existing devices and cloud solutions. The Technical Guideline therefore identifies certain existing residual risks as follows:

Mobile devices are particularly easy to steal. The open architecture of many platforms facilitates the use of malware. Installed apps can exploit existing vulnerabilities. A particular challenge is the protection of information during processing in the main memory.

The operation of the backend at public cloud providers entails special risks for sensitive user data. While a high level of entropy, secure communication and encryption methods mitigate risks, data is virtually unprotected during processing in the cloud. This places extremely high demands on cloud providers as well as other users who may simultaneously use resources of the same physical machine. An attacker who breaks out of his own virtual machine can access other virtual machines outside his client area and may hence be able to view and manipulate sensitive data of another client (here: the eHealth application) during the processing of the other client's data.

Communications between the platform, the application and the backend are protected by the cryptographically protected TLS protocol. In the present scenario, the Technical Guideline assumes single-ended authentication whereby the application checks the authenticity of the backend. The application adds its own randomness to the TLS connection process in order to make it more difficult for an attacker to break into the TLS connection. However, random numbers on smartphone platforms generally lack the quality necessary to protect sensitive data within an eHealth application. The residual risk during the process of establishing the connection is that the attacker can fake the authenticity of his own messages. This would allow the attacker to view and manipulate sensitive data transmitted from the application to the backend. The application note concerning O.Random_4 provides a countermeasure that can reduce this residual risk for the second TLS connection augmented with new entropy.

Due to the limitations described in chapters 2.1 and 2.2, it is generally not possible in relation to the scope of the Technical Guideline to make an all-encompassing statement regarding the security of the application, even taking into account all the objectives listed. Further reading will be necessary in order to enhance the security of the entire application. This is especially true for protection against attacks directly targeting the backend used and for the connection of eHealth applications with IoT devices.

# 3 Objectives of an eHealth application according to the Technical Guideline

## 3.1 Objectives

Testing according to the Technical Guideline covers the minimum security properties of eHealth applications. The security functionality to be tested can be divided into the following objectives:

(1) Testing the application purpose

(2) Testing the architecture

(3) Testing the source code

(4) Testing the third-party software

(5) Testing the cryptographic implementation

(6) Testing the authentication

(7) Testing the data storage and data protection

(8) Testing the paid resources

(9) Testing the platform-specific interactions

(10) Testing the network communication

(11) Testing the resilience

If the functionality to be protected is used, the manufacturer documents for each aspect of testing how the relevant requirement is ensured by the implementation.

### 3.1.1 Objective (1): Application purpose

| | |
|---|---|
| O.Zweck_1 | The manufacturer MUST disclose the primary purpose of the application and its backend and the use of personal data prior to installation (for instance, in the description in the app store) and inform the user at least at the time of first-time setting into operation. |
| O.Zweck_2 | The application and its backend MUST NOT collect and process data that does not serve the primary purpose of the application. |
| O.Zweck_3 | The application and its backend MUST obtain the user's consent before collecting or processing any personal data. |
| O.Zweck_4 | If the user has not expressly agreed to the use of certain data, such data MUST NOT be used by the application or the backend. |

| | |
|---|---|
| O.Zweck_5 | The application and its backend MUST enable the user to withdraw their consent. It MUST inform about the extent to which this changes the behaviour of the application. |
| O.Zweck_6 | The manufacturer MUST maintain a register that shows which user consents have been obtained. The user-specific part of the directory MUST be visible to the user. |
| O.Zweck_7 | If the application or its backend uses third-party frameworks and libraries, all functions used SHOULD be necessary for the primary purpose of the application. The application SHOULD safely disable other functions.<br><br>Application note/example: An API for social networks should only be used if this is compatible with the primary purpose of the application. |
| O.Zweck_8 | Sensitive data MUST NOT be shared with third parties unless necessary for the intended primary purpose of an application. The application MUST fully inform the user of the consequences of any disclosure of the data and obtain the user's consent (OPT-IN).<br><br>Application note/example: If the application uses a map visualisation of a third-party manufacturer, the user must be informed that certain data may be passed on to third parties. |
| O.Zweck_9 | The application MUST NOT display sensitive data on the screen unless this is necessary for the purpose of the application. |

### 3.1.2 Objective (2): Architecture

| | |
|---|---|
| O.Arch_1 | Security MUST be an integral part of software development and of the lifecycle of the entire application and its backend (see iOS Security Framework [iOSSF] and Security for Android Developers [SfAD], respectively). |
| O.Arch_2 | During the design phase of the application, it MUST already be taken into account that the application and its backend will process sensitive data during its productive phase. To these ends, the architecture of the application MUST map the secure collection, processing, storage and deletion of sensitive data over a data lifecycle. |
| O.Arch_3 | The lifecycle of cryptographic key material MUST follow an elaborated policy that includes properties, such as random number source, detailed information on the segregation of functions of keys, expiration of key certificates, integrity assurance through hash algorithms, etc. The policy MUST be based on recognised standards, such as [TR02102-1] and [NSP80057]. |
| O.Arch_4 | If the application uses a cloud as its backend system, the cloud MUST have a C5 (Cloud Computing Compliance Controls Catalogue) [KCC-C5] or equivalent attestation or certification. |

| | |
|---|---|
| O.Arch_5 | Backups and cloud backups controlled by the operating system MUST NOT contain unencrypted sensitive data and, in particular, key material. |
| O.Arch_6 | Security functions MUST always be implemented, both in the application and in the backend, as well as on all external interfaces and API endpoints. |
| O.Arch_7 | The application and its backend MUST provide authenticity and integrity protection for the application and its configuration. The application SHOULD regularly perform its own authenticity and integrity check of the application binary based on a digital signature with a certificate. |
| O.Arch_8 | If the application or its backend uses third-party frameworks or libraries (for instance, for object serialisation), the manufacturer MUST clearly inform the user about the scope of use and the security mechanisms used. The application and its backend MUST ensure the secure use of these functions. They MUST also ensure that unused functions cannot be activated by third parties. |
| O.Arch_9 | Interpreted code[5] that may interact with user input (webviews with JavaScript), MUST NOT have access to encrypted memories or user data, except as strictly necessary to fulfil the purpose of the application. |
| O.Arch_10 | The manufacturer MUST provide the user with a low-barrier means of reporting security problems. |
| O.Arch_11 | The backend MUST be able to force security-relevant updates of the application. |
| O.Arch_12 | The manufacturer CAN provide the application and updates via a trusted channel with its own app store. |
| O.Arch_13 | If the application and updates are not deployed through the normal mechanisms of the hardware platform's app store, they MUST be encrypted and signed using cryptographic measures. |

### 3.1.3 Objective (3): Source code

| | |
|---|---|
| O.Source_1 | User input MUST be checked prior to use in order to eliminate potentially malicious values before processing. |
| O.Source_2 | The manufacturer MUST provide structured data with an escape syntax. |
| O.Source_3 | Error messages and notifications MUST NOT contain sensitive data (such as a user identifier). |
| O.Source_4 | Potential exceptions in the program flow MUST be intercepted, handled in a controlled manner and documented. |

5 This does not include code of platform-specific programming languages (such as Java or Kotlin for Android).

| | |
|---|---|
| O.Source_5 | In the case of exceptions in the program flow with security-critical effects, the application and its backend SHOULD stop accessing sensitive data. |
| O.Source_6 | In program environments with manual memory management (i.e. the application itself can define exactly the time and place of memory read and write operations), the application and backend implementation MUST use secure function alternatives (for instance, sprintf_s rather than printf) for read and write access to memory segments. |
| O.Source_7 | All development support options (such as log calls, developer URLs, test methods, etc.) MUST be disabled in the productive version. |
| O.Source_8 | The manufacturer MUST ensure that no debugging mechanisms remain in the productive version. |
| O.Source_9 | The implementation of the application and its backend SHOULD enable state-of-the-art security mechanisms of the development environment, such as byte-code minimisation and stack protection. |

### 3.1.4 Objective (4): Third-party software

| | |
|---|---|
| O.TrdP_1 | Third-party libraries and frameworks SHOULD be used in their latest available version for the platform operating system in use. |
| O.TrdP_2 | The manufacturer MUST perform regular checks of third-party libraries and frameworks with regard to vulnerabilities. Functions from libraries and frameworks MUST NOT be used if any vulnerabilities are known. |
| O.TrdP_3 | Security updates for third-party libraries and frameworks MUST be uploaded promptly. The manufacturer MUST present a security concept that determines the tolerated continued use for the application and/or backend based on the criticality of exploitable vulnerabilities. After the grace period has expired, the application MUST refuse to work. |
| O.TrdP_4 | The user MUST be informed about mitigation measures that can be implemented by the user. |
| O.TrdP_5 | Before using third-party libraries and frameworks, their source MUST be checked for trustworthiness. |
| O.TrdP_6 | The application and its backend SHOULD not disclose sensitive data to third-party software. |
| O.TrdP_7 | Data received via third-party software SHOULD be validated.<br><br>Application note/example: Exceptions to O.TrdP_6 and O.TrdP_7 are, for example, frameworks and libraries for encryption (TLS). |
| O.TrdP_8 | Third-party software that is no longer kept up-to-date by the manufacturer or developer MUST NOT be used. |

### 3.1.5 Objective (5): Cryptographic implementation

| | |
|---|---|
| O.Cryp_1 | When using encryption in the application, hard-coded keys MUST NOT be used. This does not apply to state-of-the-art techniques that strongly hide the used key from reverse engineering (keyword: 'White Box Cryptography'). If static keys are used, at least one non-static key MUST be used in multi-layer encryption. |
| O.Cryp_2 | The application and its backend MUST use tried-and-tested implementations for implementing cryptographic primitives (see [TR02102-1]). |
| O.Cryp_3 | The choice of cryptographic primitives MUST be appropriate to the use case and conform to the state of the art (see [TR02102-1]). |
| O.Cryp_4 | Cryptographic keys MUST NOT be used for more than one purpose.<br><br>Application note/example: The purpose is differentiated according to protection through encryption and authentication. Different keys must be provided for this. |
| O.Cryp_5 | The strength of cryptographic keys MUST reflect the current state of the art (see [TR02102-1]). |
| O.Cryp_6 | All cryptographic keys MUST be located in a tamper-resistant environment (such as embedded Secure Element/Trusted Execution Environment). Variants for different hardware platforms must be considered. |

#### 3.1.5.1 Random numbers

| | |
|---|---|
| O.Random_1 | All random values MUST be generated using a secure cryptographic random number generator. |
| O.Random_2 | The application and its backend MUST obtain random numbers from a random number generator with high entropy. |
| O.Random_3 | The application and its backend SHOULD assign a seed consisting of at least three independent system parameters to the random number generator. It SHOULD not be possible to determine the parameters from outside the application. If the platform provides hardware random number generators that do not allow the assignment of seeds, these hardware random number generators CAN be used instead.<br><br>Application note/example: This concerns the random number generators on the device of the application as well as those of the backend. |

| | |
|---|---|
| O.Random_4 | The application SHOULD obtain a suitable random number from the backend when creating a seed for the random number generator.<br><br>Application note/example: Before the first TLS connection, the application introduces entropy, according to O.Random_3 (for instance, from user interaction and device sensors), through a seed into the local random number generator. It establishes an initial connection in order to obtain additional entropy from the random number source of the backend. The connection is then immediately closed again. The application considers the randomness obtained, according to O.Random_4, in the local random number generator. For the operational TLS connection, it uses randomness from the local random source, which has been augmented with the entropy of the random number source of the backend. |

### 3.1.6 Objective (6): Authentication

| | |
|---|---|
| O.Auth_1 | The manufacturer MUST document a concept for authentication (two-factor), authorisation (role concept) and termination of an application session. |
| O.Auth_2 | Suitable authentication and authorisation MUST be provided at the backend interface for connecting a backend system. |
| O.Auth_3 | The application SHOULD implement authentication mechanisms and authorisation functions separately. If the application requires different roles, authorisation MUST be implemented separately for each data access. |
| O.Auth_4 | The user MUST be authenticated by a second factor before sensitive data is processed in the application or its backend (step-up authentication). |
| O.Auth_5 | For user authentication in the application session, the second factor CAN be generated by the backend system. |
| O.Auth_6 | Additional information (such as device used, Wi-Fi access node used or time of access) SHOULD be included in the assessment of an authentication process. In the event of a deviation from expected parameters, an additional authentication measure (step-up authentication) MUST be performed. |
| O.Auth_7 | Strong password policies MUST exist for authentication based on a username and a password. |
| O.Auth_8 | For authentication based on a username and a password, the strength of the password used CAN be displayed to the user. Information regarding the strength of the chosen password MUST NOT be retained in the application memory or backend. |
| O.Auth_9 | The user MUST be able to change his password. |
| O.Auth_10 | The backend and the application MUST provide measures to prevent any trying out of login parameters (such as passwords). |

This can be achieved, for instance, by delaying subsequent login attempts or by using so-called captchas.

O.AO.Auth_11 If the application was interrupted (set to background mode), re-authentication MUST be requested.

#### 3.1.6.1 Authentication via biometrics

O.Biom_1 The use of biometric sensors SHOULD not be used as the only authentication mechanism. It is only permitted as part of two-factor authentication.

O.Biom_2 The manufacturer MUST define the minimum quality and characteristics of a biometric sensor to be used by the application.

O.Biom_3 The application MUST check the hardware of the biometric sensor against a blacklist or whitelist before use. The manufacturer SHOULD maintain a whitelist/blacklist of secure/unsecure biometric sensor equipment. A whitelist/blacklist MUST be kept available in the backend.

O.Biom_4 Before authentication using a biometric sensor, the application MUST always ensure that the available hardware meets the specified requirements.

O.Biom_5 Before the application uses a biometric sensor, it MUST ensure that biometric reference characteristics of the device user are available to the sensor for a comparison.

O.Biom_6 The application MUST determine when the biometric reference characteristics were changed and refuse enrolment if biometric reference characteristics were subsequently changed (i.e. since the authentication control mechanism in the application was activated).

O.Biom_7 The application MUST use functionalities inherently in the operating system (such as unlock KeyChain/KeyStore) in order to evaluate the biometric authentication.

#### 3.1.6.2 Stateful authentication measures

O.Sess_1 Session handling SHOULD be implemented using secure frameworks (see O.Ntwk_3).

O.Sess_2 Session identifiers MUST be created by the random number generator of the backend.

O.Sess_3 Session identifiers MUST be protected as sensitive data.

| | |
|---|---|
| O.Sess_4 | Session identifiers MUST NOT be stored unencrypted on permanent storage media. |
| O.Sess_5 | The application and its backend MUST actively terminate the application session after an appropriate session timeout, according to current best practice recommendations. |
| O.Sess_6 | When an application session is terminated, the application MUST securely delete the session identifier both in the device and on the backend. |

#### 3.1.6.3 Stateless authentication measures

| | |
|---|---|
| O.Tokn_1 | The authentication token SHOULD be kept in a secure memory area on the device (e.g. KeyChain/KeyStore). The authentication token in the device MUST be protected from easy access by third parties (for example, in the case of rooted/jailbroken devices). |
| O.Tokn_2 | Sensitive data MUST NOT be embedded in an authentication token. |
| O.Tokn_3 | An authentication token MUST include the fully qualified name of the backend. The application MUST check the fully qualified name. |
| O.Tokn_4 | The backend MUST use a suitable procedure to sign the authentication token (see [TR02102-1]). |
| O.Tokn_5 | The private key used to sign the authentication token MUST NOT be present in the device. |
| O.Tokn_6 | The backend MUST check the token. The signature type MUST NOT be none. |
| O.Tokn_7 | The backend MUST reject requests with an invalid or unsigned authentication token. |
| O.Tokn_8 | The backend MUST allow for an appropriate time-to-live when evaluating the validity of a token. |
| O.Tokn_9 | The backend MUST provide the user with existing authentication tokens when requested. |
| O.Tokn_10 | The backend MUST allow the user to invalidate one or all previously issued authentication tokens (for instance, if the device was lost). |

### 3.1.7 Objective (7): Data storage and data protection

| | |
|---|---|
| O.Data_1 | The as-supplied setting of the application MUST provide maximum data protection and security. |
| O.Data_2 | All sensitive data MUST be stored in encrypted form. This applies both to volatile storage (for instance, in the working memory) and to permanent storage |

| | |
|---|---|
| | (for instance, in a cloud environment). Included are cryptographic keys, with the exception of storage encryption. Preference SHOULD be given to hardware-supported key management of the platform. If sufficient key protection is ensured by the platform (for instance, in an embedded Secure Element/Trusted Execution Environment), the application CAN store keys in plain text there. |
| O.Data_3 | All sensitive data SHOULD be stored in an environment that is protected against access and tampering (such as embedded Secure Element/Trusted Execution Environment). In this way, the highest possible level of protection SHOULD be achieved for the specific platform and/or device. |
| O.Data_4 | The application MUST NOT make any resources available to third parties that allow access to sensitive data. |
| O.Data_5 | All collected sensitive data MUST NOT be kept in the application or its backend beyond the duration of its respective use. The application MUST respect the principles of data minimisation and purpose limitation. |
| O.Data_6 | Sensitive data SHOULD be stored and processed in the backend system. |
| O.Data_7 | If recording devices (such as cameras) are used, all metadata with data protection relevance, such as GPS coordinates of the recording location, hardware used, etc., MUST be removed. |
| O.Data_8 | When sensitive data is collected, the use of recording devices (such as cameras) MUST be prevented because other applications could access such data, for example, via a media gallery. |
| O.Data_9 | When sensitive data is entered via the keyboard, the application SHOULD prevent recordings from becoming visible to third parties. This specifically excludes auto-correction and auto-complete functions, third-party input devices and any form of storage that can be evaluated by third parties. |
| O.Data_10 | When sensitive data is entered, its temporary storage on the clipboard SHOULD be disabled. The application CAN alternatively implement its own clipboard which is protected against access by other apps. |
| O.Data_11 | Sensitive data, such as biometric data or private keys, MUST NOT be exported from the component on which they were generated. |
| O.Data_12 | The application SHOULD prevent access by third parties and the storage of screen contents (for instance, screenshots and displays for app switching) when displaying sensitive data. |
| O.Data_13 | The application and its backend MUST NOT write sensitive data in log files or other messages or notifications that have not been expressly enabled by the user (see O.Plat_4). |

| | |
|---|---|
| O.Data_14 | The application MUST ensure that all sensitive data is encrypted when the device is locked.<br><br>Application note/example: Older platform versions sometimes allow the application to be stored on external storage media that are not subject to storage encryption. The application MUST prohibit this because this would mean that the above requirement could not be fulfilled. |
| O.Data_15 | The application MUST securely link data stored locally to the device. |
| O.Data_16 | If the platform does not protect the selected storage medium against theft (for instance, unencrypted SD cards), the application MUST inform the user of the increased risk when the storage medium in question is selected.<br><br>Application note/example: The encryption of data at the application level is maintained in any case. |
| O.Data_17 | The application MUST ensure that all sensitive data and application-specific login information on the mobile device are completely deleted when the application is uninstalled. |
| O.Data_18 | The application MUST provide the user with the option of all sensitive data and application-specific login information also being completely deleted in the backend when the application is uninstalled. If the user decides not to delete the data in the backend, a maximum retention period appropriate for the purpose MUST be defined. The user MUST be informed about the length of the retention period. After the maximum retention period has expired, all sensitive data and application-specific login information MUST be completely deleted. The user MUST be given the opportunity to delete all data completely even before the end of the retention period.<br><br>Application note/example: Enabling uninstallation due to a device change without losing the history. |
| O.Data_19 | In order to counteract misuse of data after a device loss, the application CAN implement a kill switch, i.e. the intentional, secure overwriting of user data in the device at application level, triggered by the backend. The manufacturer MUST implement strong authentication mechanisms in order to prevent fraudulent operation of the kill switch by the user via the backend. |

### 3.1.8 Objective (8): Paid resources

| | |
|---|---|
| O.Paid_1 | The application MUST indicate to the user which services are subject to additional costs. |
| O.Paid_2 | The application MUST obtain the user's consent before carrying out any actions that are subject to a charge. |

| | |
|---|---|
| O.Paid_3 | The application MUST obtain the user's consent before requesting access (for instance, Android permissions) to paid resources.<br><br>Application note/example: The sending of text messages may incur costs and therefore requires consent. |
| O.Paid_4 | The application CAN obtain the user's permanent consent for access to frequently used, paid resources in as far as this is appropriate to the purpose of the application. |
| O.Paid_5 | The application MUST enable the user to withdraw previously given consent. |
| O.Paid_6 | The application SHOULD store the transaction history of paid resources and functions in the backend. The transaction history, including metadata, MUST be treated as sensitive data according to O.Zweck_8. |
| O.Paid_7 | If the application offers paid functions, the manufacturer MUST provide a concept that prevents third parties from tracing payment flows for the use of application functions.<br><br>Application note/example: Periodic payment processing is one possible approach to hide the actual intensity of use from third parties. |
| O.Paid_8 | The application MUST offer the user an overview of the costs incurred. If costs are due to individual accesses, the application MUST list an overview of the accesses. |
| O.Paid_9 | The validation of payment transactions MUST be carried out in the backend. |
| O.Paid_10 | Third-party payment procedures MUST meet the requirements for third-party software (see chapter 3.1.4). |

### 3.1.9 Objective (9): Network communication

| | |
|---|---|
| O.Ntwk_1 | All network communications of the application and its backend MUST be TLS-encrypted throughout. |
| O.Ntwk_2 | The configuration of TLS connections MUST be state-of-the-art and follow current best practice recommendations (see [TR02102-2]). |
| O.Ntwk_3 | The application and its backend MUST use either the security functionality of the operating system platform in use or security-tested frameworks or libraries to establish secure communication channels. |

| | |
|---|---|
| O.Ntwk_4 | The application MUST support certificate pinning, i.e. it MUST NOT accept certificates whose certificate chain the manufacturer does not consider to be trustworthy [RFC7469]. |
| O.Ntwk_5 | The application MUST check the server certificate of the backend. |
| O.Ntwk_6 | The backend MUST reject connections whose protocol version or cipher suite does not comply with [TR02102-2]. |
| O.Ntwk_7 | The application MUST validate the integrity of the backend responses. |
| O.Ntwk_8 | The application MUST deactivate platform-specific fallback mechanisms (for instance, clear Text traffic Opt-out and/or analogue In-App Transport Security). |
| O.Ntwk_9 | The application and its backend SHOULD keep log files on the backend for all established connections. It MUST be ensured that the HTTP headers are fully captured (for instance, X-Forwarded-for) when intermediate proxy servers are used. |
| O.Ntwk_10 | An aborted start MUST be logged as a security event in the backend. |

### 3.1.10 Objective (10): Platform-specific interactions

| | |
|---|---|
| O.Plat_1 | As a precondition for using the application, the device MUST be protected (password, pattern lock, etc.). The manufacturer MUST inform the user of the consequences of non-activated device protection. |
| O.Plat_2 | The application MUST NOT request any permissions other those necessary to fulfil its primary purpose. |
| O.Plat_3 | The application MUST advise the user of the purpose of the permissions to be requested and of the consequences if the user does not grant them. |
| O.Plat_4 | The application CAN provide the user with options to display messages and notifications, including those with sensitive content, if applicable. This MUST be deactivated in the as-supplied condition. |
| O.Plat_5 | The application SHOULD restrict access to intended file paths.<br><br>Application note/example: For example, by whitelisting absolute file paths. |
| O.Plat_6 | The application and its backend MUST implement access restrictions to all data. |
| O.Plat_7 | The application MUST restrict broadcast messages to authorised applications. |
| O.Plat_8 | The application and its backend MUST NOT send any sensitive data in broadcast messages. |

| | |
|---|---|
| O.Plat_9 | The offering of sensitive functionalities via interprocess communication SHOULD be prevented. If the offering is necessary to fulfil the purpose, the functionalities offered MUST be adequately protected. |
| O.Plat_10 | The application SHOULD prevent JavaScript from being active during use of WebView. If JavaScript is indispensable for the implementation of the application, the application MUST reject JavaScript from sources outside the manufacturer's control. |
| O.Plat_11 | If the application switches to background mode, it MUST remove all sensitive data from the current view (Views in iOS and Activities in Android, respectively). |
| O.Plat_12 | The application MUST disable any protocol handlers not needed in WebViews. |
| O.Plat_13 | The application MUST delete application-specific cookies after exiting the application. |
| O.Plat_14 | The application SHOULD safely overwrite all user-specific data in the working memory after termination. |

### 3.1.11 Objective (11): Resilience

| | |
|---|---|
| O.Resi_1 | The application MUST provide the user with low-barrier, best practice recommendations for the safe use of the application and its configuration. |
| O.Resi_2 | The application MUST identify rooted or jailbroken devices according to the current state of the art and respond appropriately. The manufacturer MUST outline the risks to the user's data if the app is continued (for example, the risk of disclosure). Another appropriate response is to terminate the application. |
| O.Resi_3 | The application MUST reliably detect and prevent the start in a development/debug environment. |
| O.Resi_4 | The application MUST abort its start if it is launched under unusual user rights (for instance, root or nobody). |
| O.Resi_5 | The application MUST verify the integrity of the device before processing sensitive data (such as Google Safety.Net). |
| O.Resi_6 | The application MUST verify the integrity of the backend before accessing it (see also O.Ntwk_4). |
| O.Resi_7 | The application MUST implement hardening measures, such as an integrity check before each processing of sensitive data within the program flow. |

| | |
|---|---|
| O.Resi_8 | The application MUST implement strong measures against reverse engineering and CAN use obfuscation measures, such as code obfuscation and string encryption. |
| O.Resi_9 | The application MUST implement access control mechanisms, taking into account different platforms and platform versions, so that misuse of resources by changing the platform version is prevented and sufficient protection of all assets is achieved in each execution environment. |

# References


[ASVS] The OWASP Foundation, “Application Security Verification Standard”, version 4.0, available at: https://www.owasp.org/images/d/d4/OWASP_Application_Security_Verification_Sta ndard_4.0-en.pdf

[GDR18] we are social, “Global Digital Report 2018”, January 2018 version, available at: https://wearesocial.com/de/blog/2018/01/global-digital-report-2018

[iOSSF] Apple Inc. “iOS Security Framework”, available at: https://developer.apple.com/documentation/security

[KCC-C5] Federal Office for Information Security, “Cloud Computing Compliance Criteria Catalogue”, 2020 version, available at: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/CloudComputing / Anforderungskatalog/2020/C5_2020.pdf?__blob=publicationFile&v=2

[MASVS] The OWASP Foundation, “Mobile AppSec Verification Standard (German Translation)”, version 1.1, available at: https://github.com/OWASP/owasp-masvs/releases/download/1.1.4/OWASP_Mobile_AppSec_Verification_Standard_1. 1.4_Document-de.pdf

[MSTG] The OWASP Foundation, “Mobile Security Testing Guide”, December 2016 version, available at: https://www.enisa.europa.eu/publications/smartphone-secure-development-guidelines-2016/at_download/fullReport

[NSP80057] National Institute of Standards and Technology “Recommendation for Key Management”, revision 4, available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[RFC7469] C. Evans, C. Palmer, R. Sleevi, Google Inc., “Public Key Pinning Extension for HTTP”, April 2015 version, available at: https://tools.ietf.org/html/rfc7469

[SfAD] Google Developers, “Security for Android Developers”, January 2020 version, available at: https://developer.android.com/topic/security

[SSDG] European Union Agency For Network And Information Security, “Smartphone Secure Development Guidelines”, December 2016 version, available at: https://www.enisa.europa.eu/publications/smartphone-secure-development- guidelines-2016/at_download/fullReport

[T10WASR] The OWASP Foundation, “Top 10-2017”, 2017 version, available at: https://www.owasp.org/images/9/90/OWASP_Top_10-2017_de_V1.0.pdf

[TR02102-1] Federal Office for Information Security, “Kryptographische Verfahren: Empfehlungen und Schlüssellängen”, 2019-01 version, available at: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=10

[TR02102-2] Federal Office for Information Security, “Kryptographische Verfahren: Empfehlungen und Schlüssellängen Teil 2 – Verwendung von Transport Layer Security (TLS)”, 2019-01 version, available at: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf?__blob=publicationFile&v=7

[SGBV33a] Federal Gazette,”"Sozialgesetzbuch (SGB) Fünftes Buch (V) – Gesetzliche Krankenversicherung – § 33a Digitale Gesundheitsanwendungen”, December

2019 version, available at: http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl119s2562.pdf

# Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| App | Application |
| A.* | Assumption |
| BSI | Bundesamt für Sicherheit in der Informationstechnik (Federal Office for Information Security) |
| GPS | Global Positioning System |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| IoT | Internet of Things |
| O.* | Objective |
| OSP.* | Organizational Security Policies |
| SD card | Secure Digital Memory Card |
| SDK | Software Development Kit |
| SGB V | German Social Security Code, Volume Five (*Sozialgesetzbuch (SGB) Fünftes Buch (V)*) |
| SMS | Short Message Service |
| SPD | Security Problem Definition |
| SSID | Service Set Identifier |
| T.* | Threat |
| TG | Technical Guideline |
| TLS | Transport Layer Security |
| URL | Uniform Resource Locator |
| WiFi | Wireless Fidelity |
| WLAN | Wireless Local Area Network |
| XML | Extensible Markup Language |