D11-QM Sicherheitskonzept für PersoApp auf Android OS

Editor:	Matthias Ritscher (Fraunhofer SIT)
Prüfung:	Rico Klimsa, Christine Neupert, André Gutwirth, Frank Rosenberger (alle Ageto), Sven Wohlgemuth (TU Darmstadt)
Typ:	Technischer Bericht
Projekt:	PersoApp
Version:	1.0
Datum:	18. November 2014
Status:	[FREIGABE]
Klasse:	ÖFFENTLICHKEIT
Datei:	D11-QM Sicherheitskonzept fuer PersoApp auf Android OS

Zusammenfassung

Aufgrund des speziellen Vertrauensmodells von Android OS und der Systemarchitektur gegenüber stationären IT-Systemen ist die Erstellung eines Sicherheitskonzeptes für PersoApp auf Android OS erforderlich. Dieses berücksichtigt spezielle Sicherheitsanforderungen für „Mobilität“ und leitet Handlungsempfehlungen für sichere elektronische Identitäten in der Laufzeitumgebung Android OS ab.

Konsortialleitung:
Prof. Dr. Ahmad-Reza Sadeghi und Dr. Sven Wohlgemuth
System Security Lab, TU Darmstadt/CASED, Mornewegstr. 32, 64293 Darmstadt
Tel.: +49-6151-16-75561 Fax: +49-6151-16-72135
E-Mail: persoapp@trust.cased.de Web: https://www.persoapp.de

Nutzungslizenz

Mitglieder des Konsortiums

1. **AGETO Service GmbH**, Deutschland
2. **Center for Advanced Security Research Darmstadt (CASED)**, Deutschland
3. **Fraunhofer Institut für Sichere Informationstechnologie (SIT)**, Deutschland
4. **Technische Universität (TU) Darmstadt**, Deutschland

Versionen

Version	*Datum*	*Beschreibung (Editor)*
0.1	2014-10-16	Gliederungsentwurf (Fraunhofer SIT, ZU Darmstadt/CASED)
0.2	2014-11-03	Systembeschreibung (AGETO)
0.3	2014-11-10	Version zur internen Prüfung (Fraunhofer SIT)
1.0	2014-11-18	Freigabeversion nach interner Prüfung (Fraunhofer SIT)

Autoren

Autoren	*Beiträge*
Matthias Ritscher (Fraunhofer SIT)	Kapitel 1, 3, 4.1, 4.2, 4.4, 5.1, 5.2. 5.3, Literatur
Rico Klimsa (AGETO)	Kapitel 2
Sven Wohlgemuth (TU Darmstadt)	Kapitel 4.3, 4.4, 5.4, Literatur

[1] http://creativecommons.org/licenses/by-sa/3.0/

Inhaltsverzeichnis

1 Einführung

Die Basis der Open-Source-Code-Bibliotheken und APIs der PersoApp auf Android OS wird Entwicklern zur Verfügung gestellt, die die genannten Bibliotheken und APIs in eigenentwickelten Apps verwenden und ergänzen möchten. Hierbei sollen Apps entstehen, die die eID-Funktionalität des Personalausweises (nPA) als Identifikationsmerkmal des Nutzer verwenden.

Um die sichere Nutzung der Identitätsmerkmale zu gewährleisten und durch eine App nicht das Schutzkonzept der eID-Gesamtlösung zu gefährden (Beispielsweise durch massenhafte Entwendung von eID-PINs), hat ein App-Entwickler, der die Open-Source-Code-Bibliotheken und APIs der PersoApp auf Android OS verwendet, ein Sicherheitskonzept für seine konkrete App aufzustellen. Alle in diesem Dokument vorgestellten Aspekte sind dabei zu berücksichtigen. Dieses Dokument dient der Hilfestellung zur Aufstellung des eigenen Sicherheitskonzepts.

Dieses umfasst zunächst als ersten Aspekt die Identifizierung jedes schützenswerten Gutes (Asset), das in der App erzeugt, verarbeitet, übertragen, empfangen oder (zwischen-) gespeichert wird. Hierbei ist aufzustellen, welchen Schutzbedarf dieses Gut hat und welche Komponenten unter welchen Voraussetzungen bzw. Prozessen Kenntnis über dieses Gut erlangen oder es modifizieren dürfen. Diese Aspekte werden exemplarisch in Kapitel 2 beschrieben.

Als zweiter Schritt sind Schutzziele aufzustellen, die für den gesamten Nutzungskontext der App, also auch in der Interaktion mit externen Komponenten, gewährleistet sein müssen. Es ist üblich zunächst die drei Schutzziele Vertraulichkeit, Integrität und Authentizität zu betrachten. Daneben lassen sich weitere, zum Teil aus diesen drei Grundwerten abgeleitete, zum Teil ergänzende Schutzziele formulieren, um einem gegebenen Anwendungskontext gerecht zu werden. Beispiele für Schutzziele sind Verfügbarkeit, Zugriffskontrolle und Nachvollziehbarkeit. Eine Aufstellung von Schutzzielen für eine generische App, die eID Funktionen nutzt, ist in Abschnitt 3.1 "Schutzziele" dargestellt.

Um die Schutzziele zu erreichen, sind konkrete Maßnahmen zu treffen. Nicht jede Maßnahme kann erfolgreich gegen alle Angreifer eingesetzt werden. Daher ist zunächst zu definieren, gegen welche Angreiferklassen die App geschützt werden soll (s. exemplarisch Abschnitt 3.2 "Angreifermodell"). Auf dieser Grundlage sind die Maßnahmen zur Nutzung jedes schützenswerten Gutes und zum Erreichen der erarbeiteten Schutzziele zu beschreiben.

Als weitere Aspekte zur Aufstellung eines vollständigen Sicherheitskonzeptes gibt dieses Dokument Hilfestellung bei der Erkennung relevanter Bedrohungen auf eine generische App, die eID Funktionen nutzt (Kapitel 4 "Bedrohungen") als auch Hilfe im Hinblick auf beachtenswerte Aspekte als State of the Art anwendbarer Methoden zu Sicherung von Apps mit hohem Schutzbedarf (Kapitel 5 "Handlungsempfehlungen").

2 Systembeschreibung

2.1 PersoApp eID-Client mobil

In diesem Abschnitt werden die verwendeten Android App Assets und die verwendeten Entitäten der PersoApp auf Android OS erörtert.

2.1.1 Android App Assets

Die Android App Assets enthalten Fehlermeldungen und Informationen für den Nutzer der PersoApp. Diese Texte sind in sog. Properties-Dateien innerhalb der App abgelegt. Da diese Texte an verschiedenen Stellen innerhalb der PersoApp zum Einsatz kommen, ist ihre Relevanz für den Programmablauf demzufolge hoch. Sie sind aber nicht exponiert, da sie innerhalb der erstellten *apk*-Datei vorliegen. Außerdem wurden diese Texte obfuskiert, um Rückschlüsse auf die Funktionalität der Anwendung im dekompilierten Quelltext zu vermeiden bzw. zu erschweren.

2.1.2 Verwendete Entitäten

Die schützenswerten Entitäten sind immaterielle Daten, die in die PersoApp eingegeben bzw. in der PersoApp erzeugt werden. Das Wissen der verschiedenen Komponenten über diese Entitäten soll an den Eingaben in die PersoApp exemplarisch gezeigt werden. Dazu zählen

- die eID-PIN
- die tcTokenURL
- das tcToken

eID-PIN

Die eID-PIN wird über das Tastenfeld der PersoApp eingegeben. Sie authentifiziert den Benutzer gegenüber dem nPA und ist seine technische Autorisierung für den Zugriff auf seine Identitätsdaten im Rahmen einer Online-Authentisierung. Die PIN-Eingabe erfolgt je nach Zweck in einem anderen Fragment[2].

Im Falle einer Online-Authentifizierung wird die eingegebene eID-PIN von der AuthenticateActivity in eine SecureHolder-Struktur verpackt. Dadurch wird die eingegebene PIN verschlüsselt und so vor unberechtigtem Zugriff geschützt. Anschließend wird die verschlüsselte eID-PIN an das mainViewFragment weitergeleitet. Das mainViewFragment leitet die erhaltene eID-PIN über die Klasse MainView zum SALService weiter. Der SALService übermittelt die verschlüsselte PIN durch den AndroidCardHandler an die eID-Karte und wird durch den AndroidCardHandler auch über das Ergebnis des PIN-Abgleichs benachrichtigt.

[2] AuthenticateFragment, ConfirmPinFragment, NewPinFragment, CurrentPinFragment, CANDialog

Im Falle einer Aktivierung, Änderung oder Entsperrung wird die eingegebene eID-PIN in der CommonChangePinActivity ebenfalls in eine SecureHolder-Struktur verpackt.
Als Nächstes wird die verschlüsselte eID-PIN (im folgenden PIN) an den internen Nachrichtenhandler gesendet. Der interne Nachrichtenhandler übermittelt die PIN an die MainViewFacade. Die MainViewFacade leitet die PIN an den AndroidCardHandler weiter, der sie an die eID-Karte übermittelt. Der Nachrichtenhandler zeigt außerdem dem MainViewFragment den Erfolg oder Misserfolg des PIN-Abgleichs an.

Die Übertragung der Daten an die eID-Karte erfolgt verbindungsbehaftet *(ISO/IEC 18092, ISO/IEC 14443 und JIS X 6319-4)* durch zwei gleichwertige aktive Transmitter.

tcTokenURL

Die tcTokenURL wird durch den Dienstanbieter mithilfe eines entsprechenden Links oder durch den Benutzer bestimmt, mit einem GET-Request an die PersoApp übertragen.

Die tcTokenURL wird innerhalb der PersoApp nicht verschlüsselt, da sie keine sensiblen Daten enthält und nur nach bestandenem Handshake mit der enthaltenen Adresse das tcToken ausgeliefert wird.

Die trusted-channel-Token-URL wird in der AuthenticateActivity gespeichert und in das MainViewFragment übermittelt. Das MainViewFragment startet den ECardWorker mit der tcTokenURL. Der ECardWorker speichert die URL in der ECardSession. Im SALService wird die tcTokenURL auf die Einhaltung der same-origins-police geprüft und deshalb noch mal abgefragt.

tcToken

Das tcToken wird von dem eService über eine gesicherte Verbindung (TLSv-1.2) ausgeliefert. Es enthält die Daten mit denen die PAOS-Kommunikation mit dem eID-Server initiiert wird. Der ECardWorker verarbeitet es und extrahiert die einzelnen Parameter. Der SessionIdentifier und der vorhandene PSK werden an den PAOSInitiator für die Initialisierung des PAOS-Protokolls zwischen eID-Server und eID-Client übermittelt.

Bei Erfolg der Online-Authentifizierung stößt der ECardWorker die Weiterleitung des Browsers zu der RefreshURL an. Bei Misserfolg der Online-Authentifizierung leitet der ECardWorker den Browser zu der CommunicationErrorAddress um.

2.2 Prozesse innerhalb des PersoApp eID-Clients mobil

Im folgenden Abschnitt werden die internen und die externen Prozesse des PersoApp eID-Clients erläutert.

2.2.1 Interne Prozesse

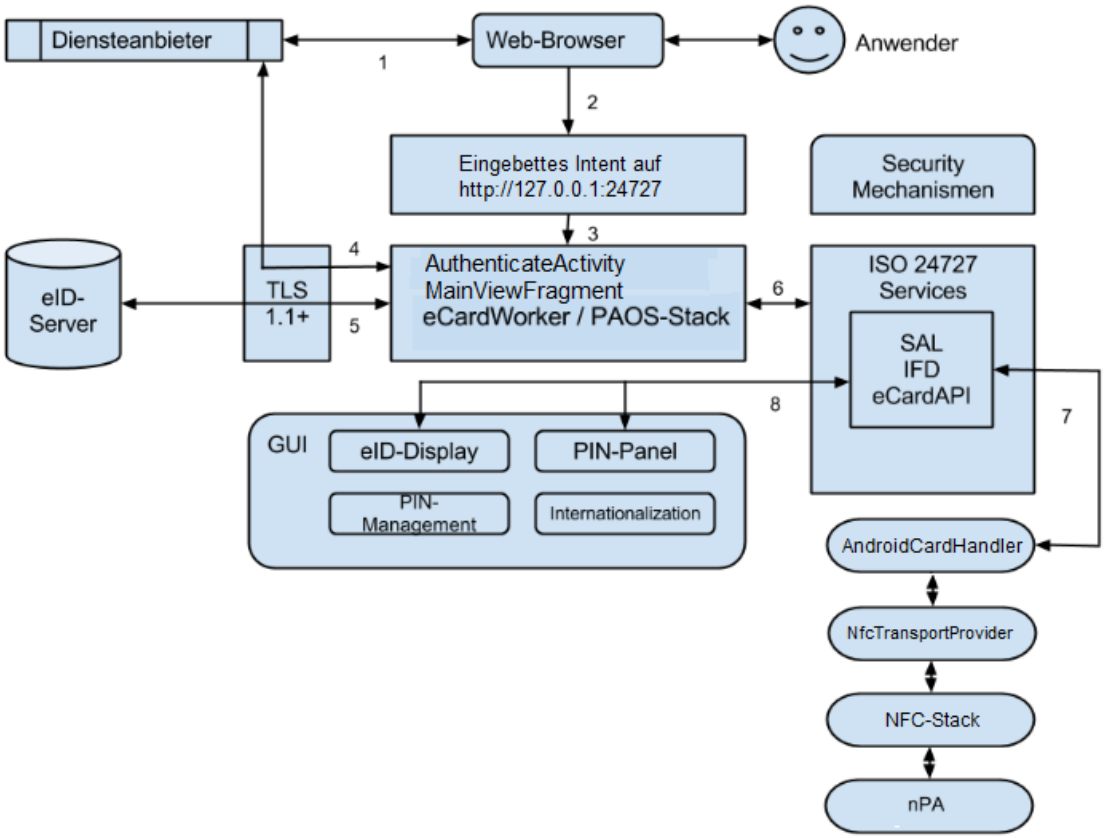

Abbildung 1 Interne Prozesse der PersoApp

1. Der Dienstanbieter stellt dem User in dem Webbrowser einen Link zur Verfügung, der die tcTokenURL enthält.
2. Der User klickt auf den Link; der Initiale GET-Request wird als Intent an die AuthenticateActivity weitergeleitet. Die Prüfung der tcTokenURL wird durch den Intentfilter durchgeführt. Die Parameter dazu werden in der AndroidManifest.xml definiert.
3. Das Intent löst die Aktion net.ageto.eid.intent.action.AUTHENTICATE aus und ruft dabei die AuthenticateActivity auf.
4. Die AuthenticationActivity meldet an das MainViewFragment den Start der Authentication. Dabei wird der ECardWorker aufgerufen, der sich das tcToken vom Dienstanbieter holt.
5. Der ECardWorker fordert den PAOS-Stack zur Kommunikation mit dem eID-Server auf.
6. Der PAOS-Stack leitet die eingehenden Requests an die lokal verfügbaren WebServices weiter und empfängt die Responses.
7. Die Web Services kommunizieren mit dem AndroidCardHandler und über diese Schnittstelle mit dem NFC-Stack und der eID-Karte

8. Dem User werden die Ergebnisse (Erfolg, Fehler, auszulesende Daten, Zertifikatsinformationen) der Kommunikation über die GUI mitgeteilt. Die Eingaben werden an den lokalen Web Service weitergeleitet und von dort an den nPA übertragen.

2.2.2 Externe Prozesse

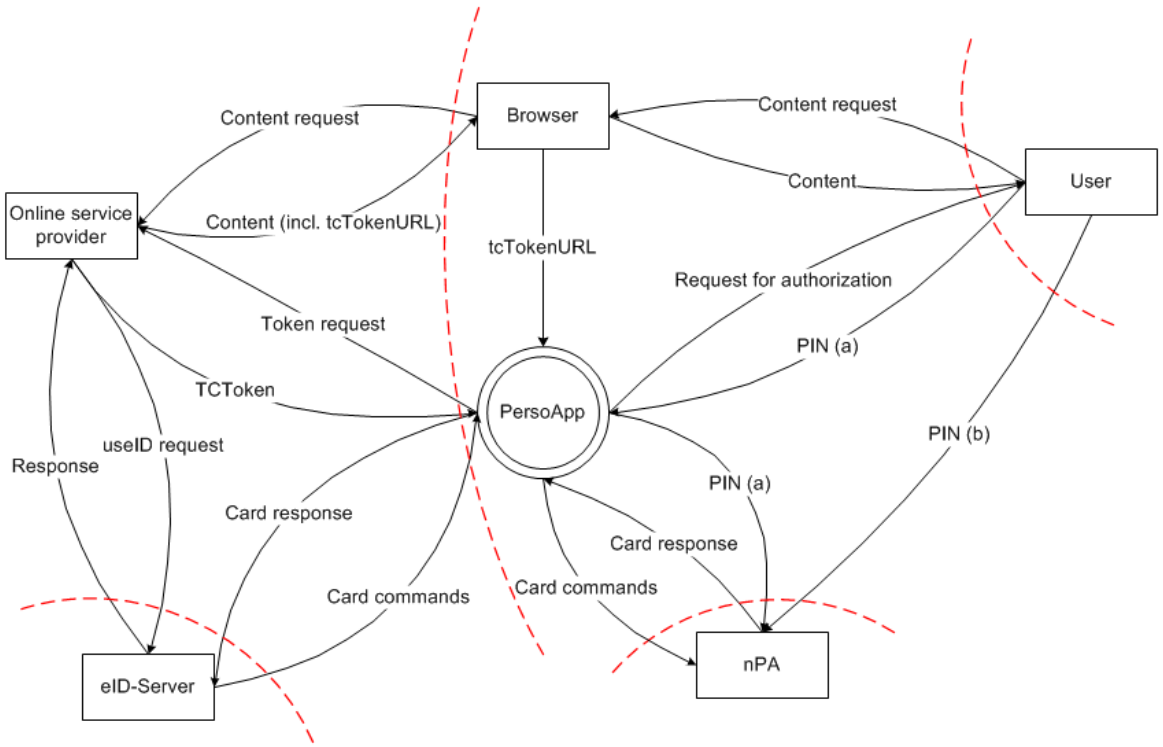

Abbildung 2 Externe Prozesse der PersoApp

Prozessbeschreibung:

1. Der Nutzer nutzt seinen Browser um Inhalt von einem Online-Dienstanbieter zu erfragen.
2. Der Online-Dienstanbieter sendet die angefragten Daten zum Browser, der die Daten dem Nutzer präsentiert. Das schließt die tcTokenURL mit ein.
3. Die tcTokenURL wird vom Browser an die PersoApp-Software weitergeleitet. Die PersoApp-Software fordert das tcToken vom Online-Dienstanbieter an.
4. Die PersoApp baut eine Verbindung zum eID-Server auf, indem es die Daten aus dem erhaltenen tcToken verwendet.
5. Die PersoApp zeigt dem Nutzer, welche Daten vom eID-Server durch den Online-Dienstanbieter angefordert werden.
6. Der Nutzer muss anschließend seine privaten Daten mit seiner geheimen PIN entsperren. Dies geschieht über die PersoApp selbst (a). Die Möglichkeit einer Eingabe durch ein spezielles Eingabegerät (b) ist für die Android-Version der PersoApp nicht möglich.
7. Die PersoApp-Software erhält die persönlichen Daten vom nPA und leitet sie zum eID-Server weiter.

8. Der eID-Server leitet die persönlichen Daten zum Online-Dienstanbieter weiter.

3 Schutzbedarf

3.1 Schutzziele

Es ist üblich als Grundwerte der Informationssicherheit die drei Schutzziele Vertraulichkeit, Integrität und Authentizität zu unterscheiden. Daneben lassen sich weitere, zum Teil aus diesen drei Grundwerten abgeleitete, zum Teil ergänzende Schutzziele formulieren, um einem gegebenen Anwendungskontext gerecht zu werden. In diesem Sicherheitskonzept werden zusätzlich zu den drei genannten Grundwerten die Aspekte Verfügbarkeit und Zugriffskontrolle sowie Nachvollziehbarkeit berücksichtigt.

Jede konkrete Implementierung einer PersoApp auf Android OS, welche die zur Verfügung gestellte Open Source Software als Basis verwendet, sollte Schutzmaßnahmen treffen, um die nachfolgend genannten Schutzziele zu erfüllen.

3.1.1 Schutzziel Authentizität (O1)

Authentizität der Interaktion zwischen der App und dem Nutzer (O1.1)

Der Nutzer muss durch geeignete Maßnahmen innerhalb der App in die Lage versetzt werden, selbst zweifelsfrei beurteilen zu können, dass er eine Interaktion mit der realen PersoApp auf Android OS durchführt. Externe Apps, die sich als die PersoApp auf Android OS ausgeben oder die Interaktion des Nutzers mit der realen PersoApp auf Android OS auf unterschiedlichen Wegen übernehmen sind daher zu detektieren oder diese Vorgehensweise ist durch geeignete Maßnahmen abzuwehren.

Zum anderen ist die PersoApp auf Android OS durch geeignete Maßnahmen in die Lage zu versetzen, selbst zweifelsfrei beurteilen zu können, dass ein Nutzer die Interaktionen mit ihr durchführt.

Authentizität der Interaktion zwischen der App und dem Online Provider (O1.2)

Die PersoApp auf Android OS hat die Identität des Online Providers, mit dem sie und deren Prozesse interagiert, zu prüfen, bevor ein Austausch (vertraulicher) Daten oder eine Ressourcenzuweisung beginnt.

In allen Fällen, in denen (vertrauliche) Daten übertragen werden, muss gegenseitig gewährleistet sein, dass die Gegenstelle authentisch ist.

Authentizität der Interaktion zwischen der App und dem eID Server (O1.3)

Die PersoApp auf Android OS hat die Identität des eID Servers mit dem sie deren Prozesse interagiert zu prüfen, bevor ein Austausch (vertraulicher) Daten oder eine Ressourcenzuweisung beginnt.

In allen Fällen, in denen (vertrauliche) Daten übertragen werden, muss gegenseitig gewährleistet sein, dass die jeweilige Gegenstelle authentisch ist.

Authentizität der Interaktion zwischen der App und lokalen Browser (O1.4)

Es muss durch die PersoApp auf Android OS sichergestellt werden, dass die durch den Browser an die PersoApp auf Android OS übertragenen Daten authentisch sind und durch den Browser versendet wurden, der im aktuellen Nutzungskontext verwendet wird.

Authentizität der App (O1.5)

Die PersoApp auf Android OS muss durch geeignete Maßnahmen innerhalb der App zur Laufzeit selbst zweifelsfrei beurteilen können, ob sie vom ursprünglichen Herausgeber erzeugt wurde.

3.1.2 Schutzziel Zugriffskontrolle (O2)

Zugriffsschutz auf Nutzerdaten (O2.1)

Der Schutz der Nutzerdaten ist eng mit Datenschutzaspekten verbunden. Die PersoApp auf Android OS hat den Zugang zu persönlichen Nutzerdaten auf autorisierte Entitäten zu beschränken.

Jeder unberechtigte Zugang und jede unbefugte Offenlegung von Benutzerdaten über die PersoApp auf Android OS als eine Art Tunnel unter Umgehung der standardisierten Sicherheitsmaßnahmen ist daher zu verhindern. Dies beinhaltet die Koordinierung der beiden folgenden Mechanismen: Freigabemechanismus auf Nutzerdaten mit der eID-PIN und geschützte Datenübertragung über PACE. Die korrekte Abstimmung dieser beiden Mechanismen ist durch die PersoApp auf Android OS in jedem Nutzungskontext zu gewährleisten.

Zugriffskontrolle auf Schnittstellen (O2.2)

Die PersoApp auf Android OS nutzt unterschiedliche Schnittstellen zur Kommunikation mit anderen Entitäten innerhalb des Nutzungskontexts. Der Zugang zu diesen Schnittstellen sollte stets mittels dezidierten Zugriffskontrollmechanismen durchgesetzt werden, um die geforderten Schutzziele zu erreichen.

Die PersoApp auf Android OS hat für jeden Kommunikationspartner zu bestimmen, ob dieser die Rechte zum Ausführen eines bestimmten Kommunikationsprozesses oder zur Nutzung einer bestimmten Schnittstelle hat.

3.1.3 Schutzziel Vertraulichkeit (O3)

Vertraulichkeit sensibler Daten (O3.1)

Sichere Datenverarbeitung ist ein Prozess der Sicherstellung, dass bestimmte Daten während, bei und nach Abschluss eines Prozesses auf sichere Art und Weise gespeichert, archiviert oder gelöscht werden. Die PersoApp auf Android OS soll Zugangsdaten, zwischengespeicherte Daten, Protokolldaten, Laufzeitdaten und temporäre Daten mit sensiblen Inhalt vor unbefugter Informationsgewinnung schützen und für diese Daten eine sichere Datenverarbeitung gemäß o.g. Definition umsetzen.

Korrekte Nutzung kryptografischer Funktionen (O3.2)

Die kryptographischen Algorithmen, die innerhalb der PersoApp auf Android OS verwendet werden, müssen ordnungsgemäß umgesetzt werden und den entsprechenden Standardalgorithmen entsprechen.

Vertraulichkeit der Interaktion zwischen der App und dem Nutzer (O3.3)

Alle Aspekte und Daten, die in der Interaktion zwischen der App und dem Nutzer erzeugt werden, sind von der PersoApp gegenüber Dritten vor unberechtigter Informationsgewinnung zu schützen.

Vertraulichkeit der Interaktion zwischen der App und dem Online Provider (O3.4)

Alle Aspekte und Daten, die in der Interaktion zwischen der App und dem Online Provider erzeugt werden, sind von der PersoApp gegenüber Dritten vor unberechtigter Informationsgewinnung zu schützen.

Vertraulichkeit der Interaktion zwischen der App und dem eID Server (O3.5)

Alle Aspekte und Daten, die in der Interaktion zwischen der App und dem eID Server erzeugt werden, sind von der PersoApp gegenüber Dritten vor unberechtigter Informationsgewinnung zu schützen.

Vertraulichkeit der Interaktion zwischen der App und lokalen Browser (O3.6)

Alle Aspekte und Daten, die in der Interaktion zwischen der App und dem lokalen Browser erzeugt werden, sind von der PersoApp gegenüber Dritten vor unberechtigter Informationsgewinnung zu schützen.

Erfüllung der Datenschutzanforderungen (O3.7)

Im Nutzungskontext werden spezifische personenbezogene eID Daten übertragen. Alle verwendeten und übermittelten personenbezogenen Daten sind vor unberechtigter Informationsgewinnung gemäß den gesetzlichen Verpflichtungen im Rahmen des Datenschutzes zu schützen.

Keine versteckten Funktionen oder Backdoors (O3.8)

Es darf keine Möglichkeit in der App bestehen, die es Dritten ermöglicht, Nutzerdaten vor oder während der Übertragung im Klartext zu lesen. Weiterhin darf die PersoApp auf Android OS ausschließlich Funktionen ausführen können, die analog in den entsprechenden BSI TRs beschrieben wurden oder in der App Beschreibung dokumentiert sind.

3.1.4 Schutzziel Integrität (O4)

Schutz vor Manipulation der App (O4.1)

Die PersoApp auf Android OS muss durch geeignete Maßnahmen innerhalb der App zur Laufzeit selbst zweifelsfrei beurteilen können, ob ein Dritter sie manipuliert hat.

Überprüfung der Integrität der Ausführungsplattform (O4.2)

Die PersoApp auf Android OS sollte durch geeignete Maßnahmen innerhalb der App zur Laufzeit selbst zweifelsfrei beurteilen können, ob ein Dritter die Ausführungsplattform (beispielsweise durch Rooting) manipuliert hat.

Detektion von Manipulationen in der Interaktion zwischen der App und dem Nutzer (O4.3)

Ein Integritätsschutz hat auf der Schnittstelle zwischen der PersoApp auf Android OS und dem Nutzer zu gewährleisten, dass durch Manipulationen keine negativen Auswirkungen im Hinblick auf die Sicherheit des Gesamtsystems verursacht werden. (Bewusst) Fehlerhafte, unterdrückte oder duplizierte Nachrichten, Daten oder Aktivitäten sind zu detektieren.

Detektion von Manipulationen in der Interaktion zwischen der App und dem Online Provider (O4.4)

Ein Integritätsschutz hat auf der Schnittstelle zwischen der PersoApp auf Android OS und dem Online Provider zu gewährleisten, dass durch Manipulationen keine negativen Auswirkungen im Hinblick auf die Sicherheit des Gesamtsystems verursacht werden. (Bewusst) Fehlerhafte, unterdrückte oder duplizierte Nachrichten, Daten oder Aktivitäten sind zu detektieren.

Detektion von Manipulationen in der Interaktion zwischen der App und dem eID Server (O4.5)

Ein Integritätsschutz hat auf der Schnittstelle zwischen der PersoApp auf Android OS und dem eID Server zu gewährleisten, dass durch Manipulationen keine negativen Auswirkungen im Hinblick auf die Sicherheit des Gesamtsystems verursacht werden. (Bewusst) Fehlerhafte, unterdrückte oder duplizierte Nachrichten, Daten oder Aktivitäten sind zu detektieren.

Detektion von Manipulationen in der Interaktion zwischen der App und lokalen Browser (O4.6)

Integritätsschutz hat auf der Schnittstelle zwischen der PersoApp auf Android OS und dem lokalen Browser zu gewährleisten, dass durch Manipulationen keine negativen Auswirkungen im Hinblick auf die Sicherheit des Gesamtsystems verursacht werden. (Bewusst) Fehlerhafte, unterdrückte oder duplizierte Nachrichten, Daten oder Aktivitäten sind zu detektieren. Schutzziel Verfügbarkeit (O5)

Robustheit gegenüber Denial-of-Service Angriffen (O5.1)

Die PersoApp auf Android OS muss einen grundlegenden Schutz vor Denial-of-Service Angriffe bieten. Dieser grundlegende Schutz beinhaltet Angriffe, die ausschließlich durch einzelne Angreifer entstanden ist. Sind mehrere Angreifer beteiligt oder verwendet ein Angreifer viele Ressourcen zur Durchführung des Angriffs, so übersteigt dieser Angriff die Möglichkeiten, die ein grundlegender Schutz vor Denial-of-Service Angriffe bieten kann und muss somit nicht betrachte werden.

Resistenz (O5.2)

Die PersoApp auf Android OS darf nicht in einen Zustand fallen, in den Benutzer nicht aktiv mit der GUI interagieren können. Ein Aufhängen der Anwendung darf

weiterhin nicht das ganze Gerät blockieren. In jedem Fall sollte die App über seine Schnittstellen im begrenzten Umfang ansprechbar bleiben.

3.1.5 Schutzziel Nachvollziehbarkeit (O6)

Providername in Datenfreigabe Freigabe (O6.1)

In der GUI, die der Nutzer zur Freigabe der eID-Daten verwendet, muss explizit und klar der Provider Name sichtbar sein, an den die eID-Daten übermittelt werden.

Deterministisches Verhalten (O6.2)

Das Verhalten von Anwendungen der PersoApp auf Android OS sollte stets deterministisch sein. Dies bedeutet, dass die gleiche Funktionalität immer wieder die gleichen Ergebnisse unter den gleichen Voraussetzungen erzeugt.

Kein Sicherheitsverlust durch falsche GUI Nutzer-Interpretation (O6.3)

Die GUI der PersoApp auf Android OS darf keine unklaren Optionen beinhalten, die den Benutzer auf Basis dieser unklaren Definition dazu führen können, versehentlich eine Einstellung mit negativem Einfluss auf die Gesamtsicherheit zu tätigen. Jede Beschreibung der sicherheitsrelevanten Einstellungen sollte deutlich ihre Wirkung darstellen und sollte keinen Auslegungsspielraum lassen.

App Verhalten gemäß der Dokumentation (O6.4)

Jede Aktion der App hat der entsprechenden Dokumentation zu entsprechen. Jedes andere nicht-deterministische Verhalten oder Funktionalität führt zu einer Verletzung dieses Schutzziels.

Zusammenfassung der Schutzziele

Die folgende Liste führt alle Schutzziele der vorangegangenen Abschnitte als Übersicht auf.

Authentizität der Interaktion zwischen der App und dem Nutzer (O1.1)
Authentizität der Interaktion zwischen der App und dem Online Provider (O1.2)
Authentizität der Interaktion zwischen der App und dem eID Server (O1.3)
Authentizität der Interaktion zwischen der App und lokalen Browser (O1.4)
Authentizität der App (O1.4)
Zugriffsschutz auf Nutzerdaten (O2.1)
Zugriffskontrolle auf Schnittstellen (O2.2)
Vertraulichkeit sensibler Daten (O3.1)
Korrekte Nutzung kryptografischer Funktionen (O3.2)
Vertraulichkeit der Interaktion zwischen der App und dem Nutzer (O3.3)
Vertraulichkeit der Interaktion zwischen der App und dem Online Provider (O3.4)
Vertraulichkeit der Interaktion zwischen der App und dem eID Server (O3.5)
Vertraulichkeit der Interaktion zwischen der App und lokalen Browser (O3.6)
Erfüllung der Datenschutzanforderungen (O3.7)
Keine versteckten Funktionen oder Backdoors (O3.8)
Schutz vor Manipulation der App (O4.1)
Überprüfung der Integrität der Ausführungsplattform (O4.2)
Detektion von Manipulationen in der Interaktion zwischen der App und dem Nutzer (O4.3)
Detektion von Manipulationen in der Interaktion zwischen der App und dem Online Provider (O4.4)
Detektion von Manipulationen in der Interaktion zwischen der App und dem eID Server (O4.5)
Detektion von Manipulationen in der Interaktion zwischen der App und lokalen Browser (O4.6)
Robustheit gegenüber Denial-of-Service Angriffen (O5.1)
Resistenz (O5.2)
Providername in Datenfreigabe Freigabe (O6.1)
Deterministisches Verhalten (O6.2)
Kein Sicherheitsverlust durch falsche GUI Nutzer-Interpretation (O6.3)
App Verhalten gemäß der Dokumentation (G6.4)

Tabelle 1 Liste der Schutzziele

3.2 Angreifermodell

In diesem Abschnitt wird das Angreifermodell anhand verschiedener Einordnungsmerkmale beschrieben. Abgeschlossen wird das Kapitel mit den für das Sicherheitskonzept relevanten Angreifertypen.

3.2.1 Mögliche Angreiferklassen

In diesem Abschnitt werden zunächst die möglichen Angreiferklassen für die PersoApp auf Android OS vorgestellt. Die einzelnen Angreiferklassen werden nach den Kategorien „Ziele“, „Form“, „Wissen“ und „Berechtigungen“ charakterisiert. Zusätzlich werden weitere Kategorisierungsmerkmale beschrieben.

Kategorisierung nach Zielen

Zunächst können Angreifer nach ihren Zielen kategorisiert werden. Die Ziele des Angreifers können beispielsweise sein:

- finanzieller Gewinn, eigene Vorteile
- Selbstwertschätzung, Reputation
- Identitätsdiebstahl
- Informationsgewinn über Dienstbetreiber/Dienstnutzer,
- Intellectual property (IP) Diebstahl
- Sabotage/Manipulation/Schwächung des Dienstanbieters
- Schmälerung der Reputation des Dienstanbieters oder des Dienstnutzers

Kategorisierung nach Form

Ebenso können Angreifer nach Organisationsform eingeordnet werden wie:

- Einzeltäter
- national/global organisiert
- Nachrichtendienste

Kategorisierung nach generellem Wissen

Bei der Kategorisierung nach generellem Wissen wird zwischen den folgenden Angreiferklassen unterschieden:

- First-Tier Angreifer
- Second-Tier Angreifer
- Third-Tier Angreifer

Bei einem *First-Tier Angreifer* handelt sich um einen Spezialisten, der aktiv neue Sicherheitslücken findet und auch ZeroDay Exploits realisieren kann. Er hat ein umfangreiches Wissen und Verständnis über aktuelle Technologien und Angriffsmethoden. Ein *Second-Tier Angreifer* hat im Gegensatz zum First-Tier Angreifer ein eingeschränkteres Wissen, welches sich auf der Höhe von Administratoren befindet. Er kann bestehende Angriffstools nutzen, versteht ihre Funktionsweise und kann sie in gewissem Rahmen weiterentwickeln. Weiterhin verfügt er über viel Zeit, um seine Angriffe durchzuführen. Ein *Third-Tier Angreifer* verfügt in der Regel über kein eigenes Wissen und kann das Risiko seiner Taten nicht einschätzen. Für seine Angriffe nutzt er existierende Angriffstools. Er trifft nur eine grobe Zielauswahl und versucht üblicherweise Massenangriffe auf bekannte Sicherheitslücken automatisiert auszuführen. Diese Angriffe sind somit in der Regel bekannt bzw. auffällig und lassen sich somit prinzipiell leicht abwehren.

Kategorisierung nach spezifischem Wissen

Bei der Kategorisierung nach spezifischem Wissen wird zwischen den folgenden Angreiferklassen unterschieden:

- Insider
- Outsider

Ein *Insider* hat spezifische Kenntnisse über die App und deren Backenddienste. Der Angreifer hat sowohl Kenntnis über die grundsätzlich eingesetzten Protokolle, Komponenten und die Architektur, als auch wie sie jeweils umgesetzt sind. So hat er detaillierte Kenntnisse über verwendete Schutzmechanismen und deren potentielle Schwachstellen oder Verwundbarkeiten in der Implementierung, Konfiguration oder der Ausführungsumgebung, die er ausnutzen kann, um Angriffe auf die eingesetzten Protokolle, Komponenten oder die Architektur durchzuführen.

Ein *Outsider* hat keine der oben genannten spezifischen Kenntnisse. Er hat jedoch möglicherweise generelle Kenntnisse im Bereich der Schutzmethoden, potentieller Schwachstellen oder Verwundbarkeiten in Implementierung, Konfiguration oder der Ausführungsumgebung, der wahrscheinlich eingesetzte Protokolle, Komponenten und der umgesetzten Architektur.

Kategorisierung nach Berechtigungen

Bei der Kategorisierung nach Berechtigungen wird zwischen den folgenden Angreiferklassen unterschieden:

- interner Angreifer
- externer Angreifer

Ein *interner Angreifer* ist prinzipiell der reguläre Benutzer des Smartphones. Er hat die notwendigen Berechtigungen zum Zugang und zum physischen Zugriff auf das Endgerät. Auch kann er lokal als regulärer Nutzer Aktivitäten auf dem Endgerät ausführen, beispielsweise zusätzliche Apps installieren oder Debugging-Maßnahmen treffen.

Ein *externer Angreifer* hat zunächst keine der genannten Berechtigungen. Externe Angreifer können mittels einer Drittanbieter App oder durch Schwachstellen auf das Endgerät gelangen. Dort sind die Auswirkungen des Angriffes zunächst durch Application Sandboxing oder andere Separierungsmechanismen des Betriebssystems limitiert. Der externe Angreifer muss sich daher durch eine Rechteausweitung erweiterte Rechte verschaffen.

Weitere Kategorisierungsmerkmale

In diesem Abschnitt werden weitere Kategorisierungsmerkmale vorgestellt, anhand derer relevante Angreifertypen zugeordnet werden können.

Weitere Kategorisierungsmerkmale sind:

- technische Ressourcen
- finanzielle Ressourcen
- sonstige Ressourcen

Technische Ressourcen umfassen verfügbare Hardware und Rechenkapazität des Angreifers, um beispielsweise eingesetzte kryptographische Mechanismen mittels Rechenkraft zu brechen. Finanzielle Ressourcen umfassen die Möglichkeit benötigte Hardware, Software oder auch menschliche Arbeitskraft zu beschaffen, um Angriffe durchzuführen. Sonstige Ressourcen umfassen alles was bisher nicht erwähnt wurde wie z.B. der Zugriff auf ein verfügbares Bot-Net um DDoS Angriffe durchzuführen.

3.2.2 Relevante Angreifertypen

Angreifer-Typ	Definition	Relevanz
Script Kiddies	**Ziel:** Angriffe ausprobieren, Selbstwertschätzung, **Form:** Einzeltäter, **Generelles Wissen:** Third-Tier, **Spezifisches Wissen:** Outsider, **Berechtigungen:** interner Angreifer, **Weiteres:** -	*relevant*
Regulärer Nutzer	**Ziel:** Vortäuschen einer anderer Identität, bzw. anderer Identitätsmerkmale, eigene Vorteile, Informationsgewinnung über Dienstbetreiber, **Form:** Einzeltäter, **Generelles Wissen:** First/Second-Tier, **Spezifisches Wissen:** Outsider, **Berechtigungen:** interner Angreifer, **Weiteres:** -	*relevant*
Bösartiger Mitarbeiter	**Ziel:** eigene Vorteile, Selbstwertschätzung, **Form:** Einzeltäter, **Generelles Wissen:** unspezifiziert, **Spezifisches Wissen:** Insider, **Berechtigungen:** unspezifiziert, **Weiteres:** Möglichkeiten der Manipulation der App und der relevanten Dienste	*teilweise relevant*
Kriminelle Einzeltäter	**Ziel:** Identitätsdiebstahl, finanzieller Gewinn, eigene Vorteile, Selbstwertschätzung, Reputation, Informationsgewinnung über Dienstbetreiber/ Dienstnutzer, Intellectual property (IP) Diebstahl, **Form:** Einzeltäter, **Generelles Wissen:** First/Second-Tier, **Spezifisches Wissen:** Outsider, **Berechtigungen:** externer Angreifer , **Weiteres:** -	*relevant*
Hacktivists	**Ziel:** Selbstwertschätzung, Informationsgewinnung über Dienstnutzer, Sabotage/Manipulation/ Schwächung des Dienstanbieters, Schmälerung der Reputation des Dienstnutzers oder des Dienstanbieters, **Form:** national/global organisiert, **Generelles Wissen:** First/Second-Tier, **Spezifisches Wissen:** Outsider, Berechtigungen: externer Angreifer, **Weiteres:** -	*relevant*
Wettbewerber	**Ziel:** Informationsgewinnung über Dienst-betreiber, Intellectual property (IP) Diebstahl, Sabotage/Manipulation/Schwächung des Dienstanbieters, Schmälerung der Reputation des Dienstanbieters, **Form:** national/global organisiert, **Generelles Wissen:** First/Second-Tier, **Spezifisches Wissen:** Outsider, **Berechtigungen:** externer Angreifer, **Weiteres:** -	*relevant*
Organsiertes Verbrechen	**Ziel:** finanzieller Gewinn, eigene Vorteile, Informationsgewinnung über Dienstnutzer, Identitäts-	*relevant*

	diebstahl, Informationsgewinnung über Dienstbetreiber/ Dienstnutzer, Intellectual property (IP) Diebstahl, Sabotage/Manipulation, **Form:** national/global organisiert, Generelles Wissen: First/Second-Tier, **Spezifisches Wissen:** Outsider, **Berechtigungen:** externer Angreifer (bei nutzbarem Exploit/Backdoor in der Ausführungsumgebung: interner Angreifer), **Weiteres:** Verfügbarkeit technischer und finanzieller Ressourcen	
Nachrichten-dienste	**Ziel:** eigene Vorteile, Identitätsdiebstahl, Informationsgewinnung über Dienstbetreiber/ Dienstnutzer, **Form:** national/global organisiert, **Generelles Wissen:** First/Second-Tier, Spezifisches **Wissen:** wahrscheinlich Innentäter Wissen vorhanden, **Berechtigungen:** externer Angreifer (bei nutzbarem Exploit/Backdoor in der Ausführungs-umgebung: interner Angreifer), **Weiteres:** hohe Verfügbarkeit technischer und finanzieller Ressourcen	*weniger relevant*

Tabelle 2 Liste der relevanten Angreifertypen

4 Bedrohungen

Im folgenden Abschnitt werden die im Projekt betrachteten Bedrohungen und Angriffsszenarien erörtert. Die verschiedenen Szenarien werden für eine differenzierte Beschreibung der später im Dokument definierten Handlungsempfehlungen benötigt.

Die Beschreibung der genannten spezifischen Bedrohungen auf die PersoApp auf Android OS und deren Ausführungsumgebung wird gemäß des STRIDE Ansatzes (**S**poofing, **T**ampering, **R**epudiation, **I**nformation Disclosure, **D**enial of Service und **E**levation of Privilege) durchgeführt. Hierbei werden ausschließlich Bedrohungen hinsichtlich der Standard Android Ausführungsumgebung berücksichtigt. Bedrohungen, die sich durch Betriebssystemmodifikationen wie bspw. Rooting ergeben, werden daher nicht berücksichtigt.

4.1 Anwendungsebene

Die PersoApp eID-Client Android App ist unterschiedlichsten Bedrohungen auf der Anwendungsebene ausgesetzt. Diese umfasst die Sicherheitscharakteristiken der PersoApp eID-Client Android App und die Interaktion der App mit anderen Apps innerhalb der Android Ausführungsumgebung.

Täuschen des Nutzers, um einen Angriff auszuführen (T1.1)

Wenn der Angreifer nicht in der Lage ist, gewünschten (Angriffs-)Aktionen mit den für ihn verfügbaren Berechtigungen durchzuführen, so kann er einen berechtigten Benutzer zur (unbeabsichtigten) Durchführung der vom Angreifer gewünschten Aktion bringen.

Täuschen des Nutzers, um Zugriff auf die eID-PIN zu erhalten (T1.2)

Diese Bedrohung besteht darin, dass der Nutzer von einer App getäuscht wird, so dass diese Kenntnis von der eID-PIN erhält. Dies kann durch Imitation oder Manipulation der regulären PersoApp eID-Client Android App geschehen, in die normalerweise die eID-PIN eingegeben wird, durch Mitlesen der Eingaben des Nutzers in die Tastatur bzw. den Events auf dem Touch-Display oder durch ähnliche Vorgänge.

Täuschen der PersoApp eID-Client Android App, um unautorisierten Zugriff auf eID Funktionen bzw. eID Daten zu erhalten (T1.3)

Die App wird von durch einem Angreifer in deren internen Prozessen in der Art getäuscht, so dass diesem ein unautorisierter Zugriff auf eID-Funktionen bzw. eID-Daten gewährt wird.

Modifikation der verarbeiteten Daten innerhalb der App (T1.4)

Innerhalb der App werden Daten auf Schnittstellen angenommen und weiterverarbeitet. Auf Basis dieser Daten werden unterschiedliche Prozesse ausgeführt oder auf Basis der Daten der Kontakt mit durch die Daten bestimmten Kommunikationspartnern initiiert. Es besteht die Bedrohung, dass Angreifer die Daten innerhalb der

App manipulieren, so dass Prozesse nicht wie gewünscht oder mit anderen Kommunikationspartnern ausgeführt werden.

Modifikation von der eID-PIN Eingabe in die App (T1.5)

Aus unterschiedlichsten Beweggründen besteht die Bedrohung, dass ein Angreifer während der eID-PIN Eingabe diese Eingabe modifiziert.

Ausführen von eID Funktionen ohne Interaktion mit dem Nutzer (T1.6)

Ist einem Angreifer die korrekte eID-PIN zu einem konkreten Personalausweis bekannt, so kann der Angreifer autark und ohne die Interaktion mit dem regulären Besitzer des Personalausweises die eID Funktionen verwenden.

Datenabfluss der eID-PIN (T1.7)

Die eID-PIN ist im Anwendungskontext der PersoApp auf Android OS das schützenswerteste Gut. Im Rahmen der Nutzung der PersoApp wird die eID-PIN auf dem Android Endgerät eingegeben. Hier ist es von Relevanz, ob Dritte direkt oder indirekt Kenntnis über die eID-PIN erhalten können. In diesem Kontext ist es gegeben, dass bei einer Eingabe der PIN über Drittanbieter Tastatur-Apps die Anbieter dieser Apps prinzipiell von diesen PIN Tastatureingaben in Kenntnis gesetzt werden können. Auch eine ungeschützte Speicherung in einem Wörterbuch ist realistisch. Im Android System können Drittanbieter Tastatur-App verwendet und deren Benutzung nicht unterbunden werden. Die Vorgehensweise die eID-PIN in Drittanbieter Tastatur-App oder in die Standard Android Tastatur einzugeben, ist nicht im Einklang mit der Schutzwürdigkeit des Datums.

Weiterhin ist es gegeben, dass sich schädliche Apps sich als Overlay über andere Apps (hier die PersoApp für Android) legen, um Bildschirmeingaben mitzulesen oder Screenshots anzufertigen. Auch auf diesem Wege kann die eID-PIN an Dritte abfließen.

Erraten oder Extraktion der eID-PIN (T1.8)

Jede Schnittstelle, die ein Angreifer zur Verfügung hat, ist als möglicher Angriffspunkt für das Abrufen von Geheimnissen wie die eID-PIN zu betrachten. Zum Beispiel kann dies durch eine unzureichend geschützte Schnittstelle ermöglicht werden, bei der das konkrete Verhalten eines Prozesses Aussagen über das Geheimnis ermöglicht (z.B. bei Seitenkanalangriffen).

Jede kleine Information, die so extrahiert wird, schränkt die Möglichkeiten ein und verkürzt oder ermöglicht am Ende einen Brute-Force-Angriff bei der Suche nach dem richtigen Geheimnis.

Ungenügende Kryptografie (T1.9)

An unterschiedlichsten Stellen werden in den Prozessen der PersoApp eID-Client Android App kryptografische Algorithmen und Prozesse verwendet. Werden schwache Algorithmen oder Prozesse verwendet oder generell kryptografische Algorithmen und Prozesse in einer nicht adäquaten oder korrekten Weise, so kann die notwendige Sicherung schützenswerter Daten nicht gewährleistet sein. Es ist daher möglich, dass diese Daten an Externe abfließen können.

Informationsabfluss von eID Informationen (T1.10)

Der Nutzer gibt mittels der App eID Daten an Dritte frei bzw. verwaltet diese. Der Angreifer kann die App verwenden, um einen unautorisierten Zugriff auf eID Daten im Generellen zu erhalten.

Informationsabfluss über Logs und Caches (T1.11)

Speichert die PersoApp eID-Client Android App schützenswerte Daten im Systemlog oder in systemweiten Caches, so können Drittanbieter-Apps möglicherweise darauf zugreifen.

Unterbrechung der eID Prozesse innerhalb der App (T1.12)

Der Angreifer kann durch Manipulation von Daten und Prozessen innerhalb der App die eID Funktion und die Weitergabe der eID Daten stören. Dies kann beispielsweise die folgenden Aktivitäten umfassen: Manipulation der eID-PIN bei der Eingabe, Manipulation der TokenURL, Unterbrechung/Umkonfiguration der Netzwerkverbindungen, Manipulation des eID Freigabe UI, etc.

App verschafft sich erweiterte Rechte auf Anwendungsebene (T1.13)

Die Umgebung für die Programmausführung kann ein Ziel für einen Angriff sein. Der Angreifer könnte in der Lage sein, aus der Ausführungsumgebung (auch als "Sandbox" bezeichnet) auszubrechen, was als Ergebnis in einer massiven Erhöhung von Berechtigungen resultiert. Diese Angriffe auf Anwendungsebene umfassen die Umgehung der Beschränkungen für die Installation neuer Anwendungen oder für die Nutzung von Anwendungen in unbeabsichtigter Weise. Auch besteht die Möglichkeit, dass Angriffe gegen den Nutzer, sein Endgerät oder andere installierte Apps direkt ausgeführt werden.

Weiterhin können durch den Angreifer auf der Systemebene erweiterte Daten und Informationen erhalten (u.a. Anwendungsdaten anderer Apps, Systemdaten und System-Logs) bzw. in der Ausführungsumgebung besonders geschützte Prozesse ausgeführt werden.

4.2 Systemebene

Täuschen des Nutzers, um erweiterten Systemzugriff zu erhalten (T2.1)

Die App täuscht den Nutzer, dass dieser Änderungen auf der Systemebene durchführt, die es der PersoApp eID-Client Android App oder einem Angreifer ermöglichen, in der Ausführungsumgebung erweiterten Systemzugriff (Root Zugriff) zu erhalten.

Veränderung des Ausführungsumgebung (T2.2)

Die App veranlasst den Nutzer, dass dieser Änderungen der Ausführungsumgebung durchführt, welche die Schutzeigenschaften der Android Ausführungsumgebung heruntersetzt. Dies betrifft beispielsweise Änderungen in den Android-Sicherheitseinstellungen (Lockscreen, Passcode, Aktivierung des USB Debugging, Erlauben der Installation von Apps aus nicht vertrauenswürdigen Quellen, etc.), oder das Installieren von (bösartigen) Drittanbieter-Apps.

Veränderung von Übergabewerten (T2.3)

Die App übernimmt Daten vom Webbrowser und verarbeitet diese weiter. Auf Basis dieser Daten werden unterschiedliche Prozesse ausgeführt oder der Kontakt mit dem durch die Daten bestimmten Kommunikationspartnern initiiert. Es besteht die Bedrohung, dass ein Angreifer die Daten schon vor Übergabe an die Schnittstelle mit der App auf Systemebene manipuliert, so dass Prozesse nicht wie gewünscht oder mit anderen Kommunikationspartnern ausgeführt werden.

Modifikation von Touch Events (T2.4)

Aus unterschiedlichsten Beweggründen besteht die Bedrohung, dass ein Angreifer (beispielsweise während der der eID-PIN Eingabe) die Touch Events auf Systemebene modifiziert.

Modifikation von App-Daten im Speicher (T2.5)

Es besteht die Bedrohung, dass ein Angreifer App-Daten zur Laufzeit im Speicher manipuliert, so dass Prozesse nicht wie gewünscht oder mit anderen Kommunikationspartnern ausgeführt werden.

Informationsabfluss durch Screenshots (T2.6)

Es besteht die Bedrohung, dass ein Angreifer (beispielsweise während der der eID-PIN Eingabe) Screenshots anfertigt und an Dritte weiterleitet. Auf diese Weise können schützenswerte Güter entwendet werden.

Informationsabfluss durch Abfrage von Touch Events (T2.7)

Es besteht die Bedrohung, dass ein Angreifer (beispielsweise während der der eID-PIN Eingabe) die Touch Events auf Systemebene abfragt. Auf diese Weise können schützenswerte Güter entwendet werden.

4.3 Hardwareebene

Modifikation der Startsequenz (T3.1)

Durch Änderung der spezifizierten und erwarteten Startsequenz des mobilen Gerätes können nicht-signierte Anwendungen installiert und gestartet werden. Solche Anwendungen können schadhaften Programmcode nachladen und Zugriff auf die eID-PIN erhalten.

Modifikation der Algorithmen für einen sicheren Systemstart (T3.2)

Eine Modifikation kann dazu führen, dass nicht-signierte Anwendungen installiert und gestartet werden. Diese wiederum können schadhaften Programmcode nachladen.

Täuschen der Geräteidentifikation (T3.3)

Die Hardwarekonfiguration eines mobilen Gerätes hat üblicherweise eine eindeutige, unveränderbare Identität. Beispiele sind IMEI, Bluetooth und WiFi-Adressen. Sollte diese Nummer bei einer Identitätsfeststellung des Gerätes geändert werden können, so ist eine Impersonierung des Gerätes möglich. Durch Nutzung der eigentlich zugeordneten Zugriffsrechten ist dann die erwartete Durchsetzung der Schutzziele bedroht.

Modifikation der Messwerte einer Startsequenz (T3.4)

Sollte ein schreibender Zugriff auf die Messwerte einer Startfrequenz möglich sein, so könnte ein Messwert für eine gestartete Anwendung gelöscht und die nachfolgenden Messwerte entsprechend geändert werden, so dass einem Prüfer die Ausführung dieser Anwendung verborgen ist. Diese Anwendung könnte schadhaften Programmcode nachladen.

Nicht autorisierter Zugriff auf den Geräteschlüssel (T3.5)

Der Geräteschlüssel ist neben kryptographischen Mechanismen ein Vertrauensanker für einen sicheren Speicher des mobilen Gerätes. Bei einem nicht-autorisierten Zugriff auf diesen Schlüssel kann der Inhalt des Speichers ausgelesen und somit dessen Vertraulichkeit verletzt werden.

4.4 Kommunikation

Impersonierung einer der Kommunikationsendpunkte (T4.1)

Der Angreifer maskiert sich erfolgreich als anderer Kommunikationsendpunkt durch die Veränderung von Daten.

Änderung der Kommunikationsinhalte (T4.2)

In Bezug auf die Kommunikationsprozesse können Kommunikationsinhalte auf unterschiedlichen Ebenen verändert werden. Zum einen kann eine Manipulation des Protokoll Payload Data erfolgen. Eine weitere Manipulationsmöglichkeit eröffnet sich durch die Manipulation von der Datenstruktur (beispielsweise Headerdaten) in jeder Übermittlung von Informationen auf den angegebenen Schnittstellen.

Weiterhin besteht die Bedrohung, dass der Protokollablauf bei der Übertragung von Daten auf den angegebenen Schnittstellen verändert wird. Der Angreifer verändert in der Kommunikation Form von Daten (Daten/Nachrichten-Abfolge), Struktur der Daten (Daten/Inhalt der Nachricht) und Herkunft der Daten (Daten/Nachrichten-Pfad).

Ungenügendes Session Handling (T4.3)

In einer verbindungsorientierten Kommunikation wird zunächst eine logische Verbindung hergestellt. In einem zweiten Schritt authentifizieren sich die Kommunikationspartner (gegenseitig), was zu einem Vertrauensverhältnis führt. Ein Ziel eines Angreifers ist es, diese logische Verbindung und das Vertrauen auszunutzen, um die gleichen Privilegien wie der rechtmäßig authentifizierte Benutzer zu erlangen.

Der Angreifer könnte versuchen, mit mitgelesenen, zufällig erzeugten oder reverse-engineerten Session-IDs die Kontrolle über die Sitzung erhalten.

Diese Bedrohung beinhaltet Session Hijacking im Netzwerk oder andere Angriffe, die auf ungenügendem Session Handling beruhen.

Umleiten des Datenverkehrs (T4.4)

Die Möglichkeit, den Datenfluss zwischen den einzelnen Komponenten des eID Prozesses umzuleiten, kann von besonderem Interesse für einen Angreifer sein. Ein Umleiten des Datenverkehrs ermöglichen Man-in-the-middle Angriffe. Dabei

durchfließen die übertragenen Daten durch eine vom Angreifer kontrollierte Komponente, was es diesem ermöglicht die zwischen den Komponenten ausgetauschten Daten zu detektieren und zu ändern. Techniken zur Umleitung des Datenverkehrs können an den verschiedenen Schnittstellen angewandt werden, um Routinginformation der übertragenen Daten zu ändern.

Modifikation der Sicherheitsparameter der Kommunikation (T4.5)

Falls ein Angreifer die Etablierung und Selektion der Sicherheitsparameter für die Kommunikation (beispielsweise SSL Cipher Suites) beeinflussen kann, besteht die Möglichkeit, dass in der Folge die Übertragung unzureichend geschützt ist. Der Angreifer kann dann die übermittelten Daten mitlesen und in den Klartext überführen.

Mitlesen auf dem Kommunikationskanal (T4.6)

In Abhängigkeit von der für die Übertragung verwendeten Schnittstelle und dem Zugriffspunkt, an dem der Angreifer die Verbindungen mitliest, können Beobachtungen der Kommunikation zwischen dem einzelnen am eID Gesamtsystem beteiligten Komponenten möglich sein. Eine detaillierte Prüfung auch von proprietären Protokollen ist möglich sowie die Erhebung von unverschlüsselten Daten. Das Mitlesen verschlüsselter Kommunikation kann dem Angreifer mit genügend Material für Brute-Force-Angriffe auf schwache Verschlüsselungsmethoden versorgen und Informationen hinsichtlich Kommunikationsmerkmale wie Quell- und Zieladressen und andere Kommunikationsstatistiken geben.

Ungenügender Transportkanalschutz bei der Datenübertragung (T4.7)

Ist die Übertragung unzureichend geschützt, kann der Angreifer die übermittelten Daten mitlesen und in den Klartext überführen.

Unterbrechung der Kommunikationsstrecken (T4.8)

Eine Unterbrechung der Kommunikationsstrecken kann dazu führen, dass ein verteiltes System nutzlos wird. Es ist damit ein potenzieller Weg zur Durchführung von Denial-of-Service Angriffen. Techniken der Unterbrechung sind:

1. Eine Überlastung des Netzes durch (Angriffs-)Instrumentalisierung von Funktionen oder Dienste des eID Dienstes. Dies wird durch Senden von Informationen in hin und her Endlos-Schleifen erreicht.
2. Absturz der Kommunikationsendpunkte: Durch die Ausnutzung von Bugs kann ein Angreifer versuchen, einzelne Komponenten des eID Systems zu beenden oder durch das Deaktivieren einiger oder des gesamten Funktionsumfang einzelner Komponenten das Gesamtsystem zum Erliegen zu bringen.
3. Protokollmanipulationen, die einen Verbindungsabbruch zur Folge haben: Erreicht werden könnte dies z.B. durch die Manipulation der Header-Daten, wodurch ein Fehlerfall im Übertragungsprotokoll ausgelöst wird oder das Senden von manipulierten Nachrichtenkontrollfluss-Daten.
4. Wiederholung von Daten: Die Wiedervorlage von ehemals gültigen Paketen, die aufgezeichnet und wiedergegeben, nachgeahmt oder mit Hilfe des Merkmals der Kommunikation von einem autorisierten Komponente erzeugt wurden, können Fehlerfälle zum Abbruch der Kommunikation auslösen.

5. Session Hijacking: Der Versuch, Befehle in eine aktive Sitzung einzuschleusen, um die Kontrolle über die Session zu erhalten, kann die Verbindung bei Detektion dieses Angriffes abbrechen lassen. Der Angreifer versucht mit mitgelesenen, zufällig erzeugten oder reverse-engineerten Session-IDs die Kontrolle über die Sitzung zu erhalten.

4.5 Zusammenfassung der Bedrohungen

Täuschen des Nutzers, um einen Angriff auszuführen (T1.1)
Täuschen des Nutzers, um Zugriff auf die eID-PIN zu erhalten (T1.2)
Täuschen der PersoApp eID-Client Android App, um unautorisierten Zugriff auf eID Funktionen bzw. eID Daten zu erhalten (T1.3)
Modifikation der verarbeiteten Daten innerhalb der App (T1.4)
Modifikation von der eID-PIN Eingabe in die App (T1.5)
Ausführen von eID Funktionen ohne Interaktion mit dem Nutzer (T1.6)
Datenabfluss der eID-PIN (T1.7)
Erraten oder Extraktion der eID-PIN (T1.8)
Ungenügende Kryptografie (T1.9)
Informationsabfluss von eID Informationen (T1.10)
Informationsabfluss über Logs und Caches (T1.11)
Unterbrechung der eID Prozesse innerhalb der App (T1.12)
App verschafft sich erweiterte Rechte auf Anwendungsebene (T1.13)
Täuschen des Nutzers, um erweiterten Systemzugriff zu erhalten (T2.1)
Veränderung des Ausführungsumgebung (T2.2)
Veränderung von Übergabewerten (T2.3)
Modifikation von Touch Events (T2.4)
Modifikation von App-Daten im Speicher (T2.5)
Informationsabfluss durch Screenshots (T2.6)
Informationsabfluss durch Abfrage von Touch Events (T2.7)
Modifikation der Startsequenz (T3.1)
Modifikation der Algorithmen für einen sicheren Systemstart (T3.2)
Täuschen der Geräteidentifikation (T3.3)
Modifikation der Messwerte einer Startsequenz (T3.4)
Nicht autorisierter Zugriff auf den Geräteschlüssel (T3.5)
Impersonierung als einer der Kommunikationsendpunkte (T4.1)
Änderung der Kommunikationsinhalte (T4.2)
Ungenügendes Session Handling (T4.3)
Umleiten des Datenverkehrs (T4.4)
Modifikation der Sicherheitsparameter der Kommunikation (T4.5)
Mitlesen auf dem Kommunikationskanal (T4.6)
Ungenügender Transportkanalschutz bei der Datenübertragung (T4.7)
Unterbrechung der Kommunikationsstrecken (T4.8)

Tabelle 3 Liste der Bedrohungen

5 Handlungsempfehlungen

Diese Handlungsempfehlungen werden Software-Entwicklern von Apps gegeben, die die Basis der Open-Source-Code-Bibliotheken und APIs der PersoApp auf Android OS für eigene Apps verwenden und ergänzen wollen.

5.1 App-Entwicklung und dessen Prozesse

Identifikation und Schutz schützenwerter Güter

Als ersten Aspekt der Aufstellung eines spezifischen Sicherheitskonzepts Ihrer App, die Open-Source-Code-Bibliotheken und APIs der PersoApp auf Android OS verwendet, identifizieren Sie jedes schützenswert Gut, das in Ihrer App erzeugt, verarbeitet, übertragen, empfangen oder (zwischen-)gespeichert wird. Stellen Sie auf, welchen Schutzbedarf dieses Gut hat und welche Komponenten unter welchen Voraussetzungen Kenntnis über dieses Gut erlangen oder es modifizieren dürfen.

Definieren Sie die Maßnahmen, wie dieses schützenswerte Gut zu behandeln ist. Dies schließt Maßnahmen zur Sicherung der Authentizität, der Integrität, der Vertraulichkeit, des Zugriffsschutzes und ähnliches ein.

Überprüfen Sie diese Anforderungen stets an Ihrer konkreten Implementierung.

Nutzung von Krypto-Funktionen und System-Krypto-APIs

Für die Maßnahme relevante Schutzziele:

- Vertraulichkeit sensibler Daten (O3.1)
- Korrekte Nutzung kryptografischer Funktionen (O3.2)
- Vertraulichkeit der Interaktion zwischen der App und dem Nutzer (O3.3)
- Vertraulichkeit der Interaktion zwischen der App und dem Online Provider (O3.4)
- Vertraulichkeit der Interaktion zwischen der App und dem eID Server (O3.5)
- Vertraulichkeit der Interaktion zwischen der App und lokalen Browser (O3.6)
- Erfüllung der Datenschutzanforderungen (O3.7)

Für die Maßnahme relevante Bedrohung:

- Datenabfluss der eID-PIN (T1.7)
- Ungenügende Kryptografie (T1.9)
- Informationsabfluss von eID Informationen (T1.10)

Falls möglich, so speichern Sie keine sensiblen Daten auf dem Gerät. Bedenken Sie, dass alle Daten, die lokal gespeichert werden, prinzipiell gefährdet sind. Für viele Anwendungen ist es funktionell und technisch möglich, auf die Speicherung von sensiblen Daten zu verzichten.

Ist dies nicht möglich, so speichern Sie sensible Daten nur verschlüsselt und verwenden Sie korrekte und adäquate Verschlüsselungsimplementierungen. Um

sensible Daten abzuspeichern, sollten Sie zunächst eine App-eigene Verschlüsselung auf Basis der Standard Crypto APIs verwenden, die Ihr System anbietet.

Verschlüsseln Sie die Benutzerdaten mit einem zufällig erzeugten Master-Schlüssel, der mit einer Passphrase durch den Benutzer entschlüsselt wird. Der Master-Schlüssel kann von der App persistent gespeichert werden. Für diese Speicherung des Hauptschlüssels sollte er mit der Benutzerpassphrase verschlüsselt werden. Es bietet sich eine Verwendung einer Schlüsselableitungsfunktion, wie PBKDF2, mit einer sehr hohen Anzahl (minimal 10.000 Iterationen) von Iterationen an. Diese durch die App-eigene Verschlüsselung gesicherten Daten übergeben Sie an system-eigene Methoden (beispielsweise Android Keychain).

Falls Sie nur die system-eigene Methoden einsetzen, so beachten Sie, dass Daten nur solange sicher sind, wie das gesetzte Gerätepasswort sicher ist, das verwendet wird, um den Schlüssel und Schlüsselmanagement abzuleiten. Viele Nutzer verwenden aus Komfortgründen kein Gerätekennwort, deshalb ist ein App-eigener Schutz schützenswerter abgespeicherter Daten wichtig.

Weiterhin überlassen Sie Kryptografie den Experten und Implementieren selbst keine Hash- oder Verschlüsselungsalgorithmen oder ähnliches. Verwenden sie nur korrekt funktionsfähige Implementierungen kryptografische Algorithmen. Zu diesen gehören die im Betriebssystem vorhandenen sowie die der bekannten großen Krypto-grafischen Bibliotheken (Beispiele sind BouncyCastle und OpenSSL).

Strikte Trennung von Release- und Debug-Code

Für die Maßnahme relevante Schutzziele:

- nicht spezifisch

Für die Maßnahme relevante Bedrohung:

- Informationsabfluss über Logs und Caches (T1.11)

In der Entwicklung kommen Situationen vor, in der gewisse Workaround-Aspekte von Entwicklern in den Code integriert werden. Diese können beispielsweise das aktive Ausschalten einer SSL-Zertifikatsprüfung sein, da das verwendete Testsystem kein offizielles Zertifikat einer Certificate Authority enthält oder das aktive Ausgeben von schützenswerter Informationen auf Log-Konsolen.

Prinzipiell ist diese Art der Entwicklung nicht verwerflich, jedoch trennen Sie strikt und stets immer solche Workarounds, die nicht in die Release-Version Ihrer App gelangen sollten, durch explizite Nutzung von Debug-Weichen.

Nutzung von schützenden Compiler-Optionen

Für die Maßnahme relevante Schutzziele:

- Authentizität der Interaktion zwischen der App und dem Nutzer (O1.1)
- Authentizität der App (O1.4)

- Zugriffsschutz auf Nutzerdaten (O2.1)
- Schutz vor Manipulation der App (O4.1)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Nutzer (O4.3)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Online Provider (O4.4)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem eID Server (O4.5)
- Detektion von Manipulationen in der Interaktion zwischen der App und lokalen Browser (O4.6)
- Resistenz (O5.2)
- Deterministisches Verhalten (O6.2)

Für die Maßnahme relevante Bedrohung:

- Täuschen des Nutzers, um einen Angriff auszuführen (T1.1)
- Modifikation der verarbeiteten Daten innerhalb der App (T1.4)
- Modifikation von App-Daten im Speicher (T2.5)

Wird eine App ausgeführt, definieren unterschiedliche interne App-Merkmale die maximale Sicherheitsstufe, welche die App erreichen kann. In diesem Zusammenhang können Sie als App-Entwickler Einstellungen vornehmen, um die Angriffsfläche der App zu reduzieren. Hierzu stehen Kompilierungsoptionen zur Verfügung, um das resultierende App Binary zu härten, z.B. gegen Attacken auf Basis von Pufferüberlauf- und Speicherkorrumpierungsproblemen.

Zusätzlich werden aufgrund von nicht verwendeten Kompilierungsoptionen die Aufwände für einen erfolgreichen Angriff auf die App deutlich verringert und heben damit das allgemeine Angriffsrisiko.

Apps in denen dieser Schutz nicht vollständig aktiviert ist, entsprechenden damit häufig nicht den Anforderungen an die Sicherheitsqualität im Anwendungsszenario, in dem der Personalausweis als Identifikationsmerkmal verwendet wird.

Anwendung des Minimal-Prinzips

Für die Maßnahme relevante Schutzziele:

- Schutz vor Manipulation der App (O4.1)

Für die Maßnahme relevante Bedrohung:

- betrifft eine Vielzahl der im Dokument benannten Bedrohungen

Sicherheitsrelevante Funktionen und Aktivitäten innerhalb der App sollten von anderen Funktionen gekapselt und getrennt werden. Diese Funktionen sollten stets nur die minimal notwendige Funktionalität enthalten und damit eine sehr geringe Code Basis haben. Diese Code Basis sollte besonders sorgfältig auf mögliche Bugs und Sicherheitsprobleme, beispielsweise in einem Code-Review unterzogen werden.

Als Ergebnis soll die Angriffsfläche minimiert werden, die es einem Angreifer ermöglichen könnte, direkt mit sicherheitsrelevanten Funktionen zu interagieren und schützenswerte Güter zu erreichen.

Dies betrifft auch die App Implementierung, diese sollte unnötige externe Bibliotheken vermeiden, da mit jeder dieser externen Bibliotheken eine unbekannte Angriffsfläche in die App hinzugefügt wird. Insgesamt kann jedoch zusammengefasst werden, dass jede Erhöhung der Komplexität zu zusätzlichen potentiellen Schwachstellen führen kann.

Etablierung von Freigabe- und Release-Prozessen

Etablieren Sie Prozesse, die genau die sukzessiven Prozessschritte spezifizieren, um eine App für die Öffentlichkeit freizugeben. Dies umfasst folgende Aufgaben der Festlegung des funktionellen Umfangs, die Festlegung des genauen Zeitplans einer Release-Freigabe, die Qualitätskontrolle zur Überwachung, die Dokumentation des Umfangs der Änderungen und die Verwaltung der Versionshistorie.

Aus IT-Sicherheitssicht ist in dem Prozessschritt der Qualitätskontrolle stets sicherzustellen, dass in den Release-Versionen der App keine Elemente von Debug(-Funktionen) enthalten sind.

Durchführung von Qualitäts- und Sicherheitstests

Sicherheitsaudits finden meist im Rahmen einer Qualitätskontrolle statt und dienen der Reduzierung von Sicherheitslücken sowie der Einführung von Best Practices. Eine Durchführung von Qualitäts- und Sicherheitstests durch (entwicklerfremde) Tester sollte stets in den App-Freigabeprozess integriert werden.

Diese Qualitäts- und Sicherheitstests sollten die im Abschnitt 3.1 genannten Schutzziele abdecken und können auf unterschiedlichen Wegen durchgeführt werden konzeptuelle Reviews, Source Code Audits, Penetrationstests, Zertifizierungen, etc.

5.2 App-Hardening

Erschwerung von App Reverse Engineering

Für die Maßnahme relevante Schutzziele:

- Authentizität der App (O1.4)
- Schutz vor Manipulation der App (O4.1)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Nutzer (O4.3)

Für die Maßnahme relevante Bedrohung:

- betrifft eine Vielzahl der im Dokument benannten Bedrohungen

Erschweren Sie durch Maßnahmen ein Reverse Engineering Ihres App Codes. So können Sie verhindern, dass ein Angreifer Einblicke in konkrete Aspekte, wie Ihre

App funktioniert, erhält. Ist die App intern komplexer aufgebaut und verschleiert es seinen Aufbau, ist aus Sicht eines Angreifers ein deutlicher Nachteil, der dazu führen kann, dass der Angreifer eine reduzierte Anzahl von möglichen Angriffsvektoren zu zur Verfügung hat.

Prinzipiell ist ein Reverse Engineering einer Android-App (APK-Datei) ziemlich leicht zu erreichen und die internen Abläufe der Anwendung können prinzipiell untersucht werden.

Sie sollten als Entwickler aktiv den Code durch Tools verschleiern, um die Schwierigkeit der Prüfung durch Angreifer zu erhöhen. Der Programmcode der App wurde sollte so obfuskiert werden, dass Variablen-, Klassen- und Methodennamen keinen Rückschluss auf die Funktion geben (beispielsweise können diese durch ein Zeichen-lange Bezeichner ersetzt werden). Trotz der einfachen Dekompilierbarkeit von Android Apps erschwert dies die Interpretation der generierten Quelltexte erheblich. Weiterhin sollte der Programmcode keinerlei Debug-Informationen enthalten, die einem Angreifer Hinweise auf Klassen- und Methodennamen vor der Obfuskierung geben könnten. Weiterhin sollte eine Obfuskierung aller Strings im App-Code durchgeführt werden.

Der Entwickler interagiert stets mit dem nicht-obfuskierten Source Code. Beim Kompilieren werden durch (Zusatz)Tools innerhalb der Entwicklungsumgebung auf Basis des nicht-obfuskierten Source Code alle genannten Obfuskierungen durchgeführt [ProGuard, DexGuard].

Wenn die Anwendung hochsensible Daten verarbeitet, erwägen Sie Integration von Anti-Debug-Techniken in Ihre App (Debugging Detection, Debugging Prevention).

Integritätsschutz der App

Für die Maßnahme relevante Schutzziele:

- Authentizität der App (O1.4)
- Schutz vor Manipulation der App (O4.1)
- Authentizität der Interaktion zwischen der App und dem Nutzer (O1.1)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Nutzer (O4.3)

Für die Maßnahme relevante Bedrohung:

- betrifft eine Vielzahl der im Dokument benannten Bedrohungen

Sichere Apps enthalten Detektionsalgorithmen, die die App-Ausführung verhindern, wenn diese detektiert haben, dass die App manipuliert wurde. Hierfür werden Prüfsummen, digitale Signaturen und andere Validierungsmechanismen in der Anwendung verwendet, um zu detektieren, ob Daten durch den Versuch die Anwendung zu manipulieren, verändert wurden.

Weiterhin durch Erzwingen einer eingebetteten digitalen Signatur in der Anwendung auch die Authentizität der einzelnen App-Inhalte (Dateien) verifizieren.

Jede dieser Kontrollen kann prinzipiell durch Reverse Engineering von einem Angreifer umgangen werden, aber eine solche Überprüfung und Manipulation erhöht signifikant den erforderlichen Aufwand und die erforderliche Zeit, um die Anwendung anzugreifen. Daher ist es zwingend notwendig, stets die Maßnahmen zum Integritätsschutz der App mit den Maßnahmen zur Erschwerung von App Reverse Engineering zu verknüpfen.

Wenn Manipulationen an der App detektiert werden, sollten weitere Maßnahmen getroffen werden, beispielsweise das Löschen aller Benutzerdaten, Schlüssel und andere wichtiger Daten bzw. Die Benachrichtigung an das Backend.

Auf Android kann Public Key Signaturen verwendet werden, um eine App zu unterzeichnen. Der öffentliche Schlüssel aus kann dem App-Zertifikat ausgelesen und verwendet werden, um die Anwendung zu überprüfen, ob sie mit dem privaten Schlüssel des Entwicklers signiert wurde. Die Prüfung digitaler Signaturen ist besonders nützlich, um zu Versuche eines Angreifers zu erkennen, wenn dieser die App gegen eine manipulierte, vom Angreifer selbst signierte App austauscht.

Sichere Datenhaltung im RAM

Für die Maßnahme relevante Schutzziele:

- Vertraulichkeit sensibler Daten (O3.1)
- Vertraulichkeit der Interaktion zwischen der App und dem Nutzer (O3.3)
- Vertraulichkeit der Interaktion zwischen der App und dem Online Provider (O3.4)
- Vertraulichkeit der Interaktion zwischen der App und dem eID Server (O3.5)
- Vertraulichkeit der Interaktion zwischen der App und lokalen Browser (O3.6)
- Erfüllung der Datenschutzanforderungen (O3.7)

Für die Maßnahme relevante Bedrohung:

- Datenabfluss der eID-PIN (T1.7)
- Erraten oder Extraktion der eID-PIN (T1.8)
- Informationsabfluss von eID Informationen (T1.10)

In Android verbleibt jede Anwendung (auch nach ihrem Gebrauch) im Speicher, bis der Speicher vom System zurückgefordert wurde. Schützenswerte Daten können daher über lange Zeit im Speicher verweilen. Ein Angreifer, der ein Endgerät findet oder stiehlt, kann prinzipiell ein Speicherabbild des RAM Speichers erzeugen [FROST].

Halten Sie daher sensible Daten wie Entschlüsselungsschlüssel nicht länger als erforderlich im RAM. Die Variablen, denen schützenswerte Daten zugewiesen wurden, sollten nach Gebrauch direkt auf `null` zurückgesetzt werden.

Geschützte Eingabe der eID-PIN

Für die Maßnahme relevante Schutzziele:

- Authentizität der Interaktion zwischen der App und dem Nutzer (O1.1)
- Zugriffsschutz auf Nutzerdaten (O2.1)
- Vertraulichkeit sensibler Daten (O3.1)
- Vertraulichkeit der Interaktion zwischen der App und dem Nutzer (O3.3)
- Erfüllung der Datenschutzanforderungen (O3.7)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Nutzer (O4.3)

Für die Maßnahme relevante Bedrohung:

- Täuschen des Nutzers, um Zugriff auf die eID-PIN zu erhalten (T1.2)
- Täuschen der PersoApp eID-Client Android App, um unautorisierten Zugriff auf eID Funktionen bzw. eID Daten zu erhalten (T1.3)
- Modifikation von der eID-PIN Eingabe in die App (T1.5)
- Ausführen von eID Funktionen ohne Interaktion mit dem Nutzer (T1.6)
- Datenabfluss der eID-PIN (T1.7)
- Erraten oder Extraktion der eID-PIN (T1.8)
- Informationsabfluss von eID Informationen (T1.10)
- Modifikation von Touch Events (T2.4)
- Informationsabfluss durch Abfrage von Touch Events (T2.7)

Die Eingabe der eID-PIN muss besonders geschützt sein. Die Eingabe der eID-PIN über die normale Android Tastatur entspricht nicht dem erforderlichen Sicherheitsanforderungen (insbesondere wenn Drittanbieter Tastatur-Apps vom Anwender verwendet werden).

Von daher ist in der GUI der PersoApp für Android eine eigene PIN Eingabe Maske zu etablieren, auf der der Nutzer durch Touch-Events auf eingeblendete Schaltfelder die eID-PIN eingibt. Um die PIN Eingabe zu schützen, ist dieses GUI Eingabefeld durch Maßnahmen gegen Touchjacking und Overlay Framing zu sichern.

Können Touchjacking und Overlay Framing nicht durch die App-Implementierung ausgeschlossen werden, so sollten zumindest die Schaltfelder bei jeder PIN-Eingabe zufällig stets andere Konfigurationen annehmen, so dass von der (aufgenommenen) Touch-Koordinate nicht zwingend auf die korrekte PIN Ziffer geschlossen werden kann.

Verhinderung von Screenshots/Screencasts

Für die Maßnahme relevante Schutzziele:

- Vertraulichkeit sensibler Daten (O3.1)

Für die Maßnahme relevante Bedrohung:

- Datenabfluss der eID-PIN (T1.7)
- Erraten oder Extraktion der eID-PIN (T1.8)
- Informationsabfluss von eID Informationen (T1.10)
- Informationsabfluss durch Screenshots (T2.6)

Das Android Betriebssystem erlaubt es mehrere Apps gleichzeitig auszuführen. Wechselt der Benutzer die App wird die vorherige Vordergrund-App weiterhin im Hintergrund ausgeführt. Zusätzlich wird ein Screenshot vom letzten Zustand der App gemacht, der in der Recents-Liste angezeigt wird.

Derzeitig ist keine Vorgehensweise bekannt, wie man Zugriff aus einer App heraus auf die in der Recents-Liste dargestellten Screenshots zugreifen könnte. Jedoch sollten Apps, die mit sensiblen Daten umgehen, prophylaktisch das automatische Nutzen eines Screenshots in der Recents-Liste unterbinden.

Dies geschieht dadurch, dass man alle in der App verwendeten sollten mit dem Flag WindowManager.LayoutParams.FLAG_SECURE versehen werden. Dies deaktiviert die Möglichkeit Screenshots über die Funktionen des Betriebssystems anzufertigen, so lange das entsprechende Fenster dargestellt wird. Dadurch werden sowohl manuelle Screenshots, als auch automatische Screenshots in der Recents-Liste deaktiviert. Stattdessen wird nur ein schwarzes Bild angezeigt.

Generelle Screenshots und Screencasting (Filmen des Bildschirminhalts) stellen höhere Anforderungen an die durchzuführenden Maßnahmen. Hier sind kombinierte Maßnahmen notwendig, um zum einen zu detektieren, ob ein Screencasting überhaupt möglich ist (beispielsweise Rooting-Status der Ausführungsumgebung und installierte Apps) und weiterhin Techniken, um aufgenommene Screenshots bzw. Screencasts zu verhindern bzw. unkenntlich zu machen. In [ScreenProtect] wird vorgestellt, mit optischen Tricks zu arbeiten, die die Eigenheiten der menschlichen Wahrnehmung für die genannten Zweck der Unkenntlichmachung aufgenommener Screenshots/Screencasts ausnutzt.

Validierung jedes Eingabewertes

Für die Maßnahme relevante Schutzziele:

- Schutz vor Manipulation der App (O4.1)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Nutzer (O4.3)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Online Provider (O4.4)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem eID Server (O4.5)
- Detektion von Manipulationen in der Interaktion zwischen der App und lokalen Browser (O4.6)

- Robustheit gegenüber Denial-of-Service Angriffen (O5.1)
- Deterministisches Verhalten (O6.2)

Für die Maßnahme relevante Bedrohung:

- Änderung der Kommunikationsinhalte (T4.2)

Eine Vorbedingung im Hinblick auf die App-Sicherheit ist, dass alle Eingaben in die App auch von nicht vertrauenswürdigen Entitäten stammen könnten. Eingaben in Dienste sind stets gründlich zu filtern und zu validieren vor der Übernahme in die internen App-Prozesse. Input-Validierung sollte auch zwischen den einzelnen Modulen innerhalb der App durchgeführt werden, auch dort können Daten auf dem Transportweg abgefangen und manipuliert werden.

Validieren Sie jede Benutzereingaben vor der Übertragung in die internen App-Prozesse, ob sie plausibel, zum erwarteten Datentyp gehört, in einem vorgegebenen Wertebereich oder einer vorgegebenen Wertemenge liegt. Weiterhin überprüfen Sie ob die Benutzereingabe unerwünschte Teile enthält, die bspw. für Injection-Angriffe verwendet werden können. Ist dies der Fall, so verwerfen Sie die Eingabe oder filtern die ungewünschten Teile aus dem Eingabewerten heraus.

Umgang mit gerooteten Geräten

Für die Maßnahme relevante Schutzziele:

- Überprüfung der Integrität der Ausführungsplattform (O4.2)

Für die Maßnahme relevante Bedrohung:

- Überprüfung der Integrität der Ausführungsplattform (O4.2)
- Täuschen des Nutzers, um einen Angriff auszuführen (T1.1)
- App verschafft sich erweiterte Rechte auf Anwendungsebene (T1.13)
- Täuschen des Nutzers, um erweiterten Systemzugriff zu erhalten (T2.1)
- Veränderung des Ausführungsumgebung (T2.2)

Apps auf gerooteten Geräte können erweiterte Zugriffrechte erlangen, und damit außerhalb ihrer Application-Sandbox und im System Daten einsehen bzw. verändern.

Falls die Sicherheitsarchitektur einer App nicht auf die Gefährdungen durch gerootete Geräte eingeht, sollte durch geeignete Mechanismen festgestellt werden, ob die Ausführungsumgebung modifiziert bzw. gerootet ist. Ist das Ergebnis des Tests positiiv, soll die Programmausführung gestoppt und weitere Maßnahmen getroffen werden, beispielsweise das Löschen aller Benutzerdaten, Schlüssel und andere wichtiger Daten bzw. Die Benachrichtigung an das Backend.

5.3 Kommunikationssicherheit

Verifikation aller Kommunikationspartner

Für die Maßnahme relevante Schutzziele:

- Authentizität der Interaktion zwischen der App und dem Nutzer (O1.1)
- Authentizität der Interaktion zwischen der App und dem Online Provider (O1.2)
- Authentizität der Interaktion zwischen der App und dem eID Server (O1.3)
- Vertraulichkeit sensibler Daten (O3.1)

Für die Maßnahme relevante Bedrohung:

- Umleiten des Datenverkehrs (T4.4)
- Änderung der Kommunikationsinhalte (T4.2)
- Modifikation der Sicherheitsparameter der Kommunikation (T4.5)
- Mitlesen auf dem Kommunikationskanal (T4.6)
- Ungenügender Transportkanalschutz bei der Datenübertragung (T4.7)
- Unterbrechung der Kommunikationsstrecken (T4.8)

Nur ein kleinerer Anteil der Smartphone-Apps agiert vollständig autark ohne externe Ressourcen. Die Mehrheit der Apps interagiert mit externen Entitäten über Netzwerk- oder lokaler Kommunikation. Falls eine Smartphone-App mit Servern interagiert und diese Kommunikation sensible Daten beinhaltet, sollte in der App die Kommunikationssicherheit in der richtigen Art und Weise umgesetzt werden.

Die Umsetzung korrekter Secure Socket Layer (SSL) oder Transport Layer Security (TLS) Kommunikation kann in der App-Entwicklung prinzipiell einfach mit den Standardfunktionen des Smartphone-Betriebssystems durchgeführt werden. In der Entwicklungsphase einer Smartphone-App wird jedoch die SSL/TLS-Konfiguration oder ihre Prozesse häufig modifiziert, um das Debugging oder die Funktion in einer Testumgebung ohne gültige Zertifikate zu ermöglichen. Dies wird benötigt, wenn die Test-Umgebung oder -weitaus schlimmer- die Produktivumgebung kein Server-Zertifikat verwendet, das durch eine Certificate Authority (CA) unterzeichnet wurde. App-Entwickler lösen dieses Problem durch die Deaktivierung oder Änderung der SSL/TLS-Sicherheitsmaßnahmen.

Eine dieser Änderungen, die das Sicherheitsniveau senkt, ist die teilweise oder vollständige Deaktivierung der Zertifikatsvalidierung. Eine solche Änderung eröffnet Möglichkeiten für Man-in-the-Middle Angriffe, bei denen der Angreifer die ausgetauschten Daten mitlesen und manipulieren kann. Die Vertraulichkeit, Authentizität und Integrität der Daten der Kommunikation sind daher bei der festgestellten unsicheren Zertifikatsvalidierung grundlegend gefährdet.

Da Anwendungen genau wissen, mit welcher Gegenstelle sie sich verbinden, ist es oftmals akzeptabel, ein sogenanntes Certificate Pinning zu verwenden. Hier sind je wird das verwendete Server Zertifikat der Gegenstelle in die App verankert. Wird

eine solche Lösung implementiert, dann ist zwingend ein Integritätsschutz des inkludierten Zertifikats innerhalb der App notwendig.

Nutzen Sie daher die systemeigenen Funktionen innerhalb von Android, um eine Zertifikatsvalidierung der Serverzertifikate durchzuführen.

Keine gemischte Nutzung von HTTP und HTTPS

Für die Maßnahme relevante Schutzziele:

- Vertraulichkeit sensibler Daten (O3.1)
- Vertraulichkeit der Interaktion zwischen der App und dem Online Provider (O3.4)
- Vertraulichkeit der Interaktion zwischen der App und dem eID Server (O3.5)

Für die Maßnahme relevante Bedrohung:

- Informationsabfluss von eID Informationen (T1.10)
- Änderung der Kommunikationsinhalte (T4.2)
- Umleiten des Datenverkehrs (T4.4)

Immer wieder wird sowohl geschützte als auch ungeschützte Kommunikation innerhalb einer App für die Kommunikation mit demselben Server verwendet. Oft wird von den Entwicklern argumentiert, dass der ungeschützte Zugriff über HTTP nicht problematisch sei, da die übertragenen Informationen nicht vertraulich wären. Dies berücksichtigt jedoch nicht, dass ein Angreifer jede ungeschützte Kommunikation nicht nur lesen sondern auch manipulieren kann.

Dies gibt einem potentiellen Angreifer die Möglichkeit Server-Anfragen oder -Antworten zu verändern (oder mit eigenen Funktionen zu ergänzen) und damit die auswertende App-Umgebung zu einem anderen Verhalten (im Bezug auf das Verhalten mit unmodifizierten Daten) zu bewegen. Dies kann u.a. verwendet werden, um das Vertrauen des Benutzers in eine App auszunutzen, bspw. durch eine hinzugefügte Dialogbox mit Passwortabfrage, deren Eingaben an den Angreifer gesendet werden.

Darüber hinaus ist es denkbar, dass vergessen wurde ein 'S' an eine 'HTTP'-URL hinzuzufügen, da die zugrunde liegenden Schnittstellen nicht zwischen einer ungeschützten und geschützten Kommunikation unterscheiden und damit kein Fehlverhalten sichtbar wird. Ungeschützte Kommunikation und damit auch die enthaltenen Autorisierungsgeheimnisse gegenüber Servern können jedoch z.B. bei öffentlichen WLANs ganz einfach mitgelesen werden. So könnte ein Angreifer die Kontrolle über die Verbindung erlangen und damit auch Zugriff auf die Dienstkontodaten des App-Benutzers mit allen gespeicherten Informationen erhalten.

Verwenden Sie daher bei jedem Kommunikationsvorgang, unabhängig von den transportierten Daten eine HTTPS Verbindung, wenn die Gegenstelle diese Sicherung unterstützt.

Vertrauenswürdiger Kommunikationsmanager

Für die Maßnahme relevante Schutzziele:

- Zugriffskontrolle auf Schnittstellen (O2.2)
- Vertraulichkeit sensibler Daten (O3.1)
- Korrekte Nutzung kryptografischer Funktionen (O3.2)
- Vertraulichkeit der Interaktion zwischen der App und dem Online Provider (O3.4)
- Vertraulichkeit der Interaktion zwischen der App und dem eID Server (O3.5)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem Online Provider (O4.4)
- Detektion von Manipulationen in der Interaktion zwischen der App und dem eID Server (O4.5)

Für die Maßnahme relevante Bedrohung:

- Impersonierung als einer der Kommunikationsendpunkte (T4.1)
- Änderung der Kommunikationsinhalte (T4.2)
- Ungenügendes Session Handling (T4.3)
- Umleiten des Datenverkehrs (T4.4)
- Modifikation der Sicherheitsparameter der Kommunikation (T4.5)
- Mitlesen auf dem Kommunikationskanal (T4.6)
- Ungenügender Transportkanalschutz bei der Datenübertragung (T4.7)
- Unterbrechung der Kommunikationsstrecken (T4.8)

Eine durchgängige Zugriffskontrolle, welche die Erstellung und den Schutz von Netzwerkverbindungen regelt, ist bei Android OS nicht gegeben. Android OS regelt einen Zugriff auf Netzwerke über die Rechte im Android OS Application Layer. Netzwerkzugriff durch Linux-Dienste und “Native” Bibliotheken wie auch deren Verwendung von Sicherheitsprotokollen wird durch Android OS nicht betrachtet. Die Empfehlung ist einen vertrauenswürdigen Kommunikationsmanager auf Android OS Geräten zu integrieren, der diese Zugriffskontrolle durchsetzt. Die Schnittstellen des vertrauenswürdigen Kommunikationsmanager sind:

- Zugriffskontrolle auf Netzwerk-Ports
- Zugriffskontrolle auf Android Anwendungen, die eine Netzwerkverbindung herstellen
- Korrektheit von Sicherheitsanwendungen

Der erste Punkt bezieht sich auf eine Erweiterung des Android OS Linux Kernels um eine Firewall. Eine Option ist netfilter/iptables. Da netfilter/iptables im Standardmodus mit Root-Rechten ausgeführt wird, sollte dieser Linux-Dienst durch eine typensichere Zugriffskontrolle gehärtet werden.

Der zweite Punkt bezieht sich auf die Schnittstelle des Android OS Referenzmonitors für Android Anwendungen und dem Netzwerkzugang des Linux Kernels. Es sollte

nicht möglich sein, dass eine Android OS Anwendung, die für einen Netzwerkzugriff autorisiert sind, eine andere Android OS Anwendung verwendet, die ebenfalls für einen Netzwerkzugriff autorisiert ist. Als Beispiel sollte der Web Browser nur TSL/SSL Verbindungen akzeptieren. Dies kann durch sein MANIFEST spezifiziert werden. Dies soll Spoofing-Angriffe, z.B. durch Phishing, verhindern.

Der dritte Punkt bezieht sich auf die Implementierung der Sicherheitsanwendung, z.B. der Firewall mit netfilter/iptables und der TSL/SSL Implementierung. Falls eine Sicherheitsschwachstelle existiert, so kann sie durch schadhaften Code ausgenutzt werden. In diesem Fall übernimmt der erfolgreiche Angreifer die Kontrolle über die Sicherheitsanwendung und damit über den Netzwerkverkehr des Android OS Gerätes. Sicherheitsschwachstellen finden sich nicht nur in der Implementierung (s. Heartbleed) sondern auch in der Konfiguration (s. Rechteeskalierung) von Sicherheitsanwendungen. Daher sollten der vertrauenswürdige Kommunikationsmanager und weitere Sicherheitsanwendungen sowohl in seiner Implementierung als auch Konfiguration auf Schwachstellen hin geprüft werden. Für die Kommunikationsprotokolle werden folgende Verbesserungen vorgeschlagen:

Bluetooth

Länge der PIN: Damit es einem Angreifer erschwert wird, Zugriff auf die geheimen Schlüssel des Bluetooth-Protokolls zu erhalten, sollte die PIN ausreichend zufällig und lang sein.

Schutz des Geräteschlüssels (Unit Key): Um zu vermeiden, dass ein Gerät den Schlüssels eines anderen Gerätes lernt, welcher im ersten Schlüsselaustauschprotokoll verwendet wird, sollte das Gerät für jede Kommunikation mit einem anderen Gerät einen eigenen Sitzungsschlüssel generieren. Dazu könnte der Geräteschlüssel als Eingabe für einen Zufallszahlengenerator dienen, um den Bezug zu dem Gerät nachweisen zu können.

Sicherheit auf dem Application Layer: Sicherheitsprotokolle des Application Layer, bspw. IPSec, sollte zum Schutz der Kommunikation mit Bluetooth verwendet werden. Sollten übliche zertifikatsbasierte Methoden eingesetzt werden, so kann sich das Gerät gegen einen Mann-in-the-Middle Angriff schützen. Um den Einsatz solcher Methoden sicherzustellen ist eine typensichere Durchsetzung der Zugriffskontrolle durch ein erweitertes Android OS eine notwendige Bedingung.

WiFi

Zum Schutz von Vertraulichkeit, Integrität und Zurechenbarkeit einer HTTP-Verbindung wird ein ständiger Einsatz der *de facto* Standardprotokolle TLS/SSL und IPSec empfohlen, wobei deren Implementierung und Konfiguration auf Schwachstellen hin geprüft werden sollte. Um den Einsatz solcher Methoden sicherzustellen ist eine typensichere Durchsetzung der Zugriffskontrolle durch ein erweitertes Android OS eine notwendige Bedingung.

Telekommunikationsnetzwerk (3G usw.)

Die Empfehlung ist der Einsatz des de facto Standardprotokolls IPSec, um Vertraulichkeit, Integrität und Zurechenbarkeit unter bekannten Schwachstellen zu

erzielen. Um den Einsatz solcher Methoden sicherzustellen ist eine typensichere Durchsetzung der Zugriffskontrolle durch ein erweitertes Android OS eine notwendige Bedingung.

5.4 OS-Hardening

Basis-Hardening des Android OS Application Layer

Für die Maßnahme relevante Schutzziele:

- Authentizität der Interaktion zwischen der App und dem Nutzer (O1.1)
- Zugriffsschutz auf Nutzerdaten (O2.1)
- Authentizität der App (O1.4)
- Schutz vor Manipulation der App (O4.1)
- Überprüfung der Integrität der Ausführungsplattform (O4.2)

Für die Maßnahme relevante Bedrohung:

- Täuschen der PersoApp eID-Client Android App, um unautorisierten Zugriff auf eID Funktionen bzw. eID Daten zu erhalten (T1.3)
- Modifikation der verarbeiteten Daten innerhalb der App (T1.4)
- App verschafft sich erweiterte Rechte auf Anwendungsebene (T1.13)
- Täuschen des Nutzers, um erweiterten Systemzugriff zu erhalten (T2.1)
- Veränderung des Ausführungsumgebung (T2.2)

Ein unbefugter Zugriff kann auf einem Android OS Gerät durch folgende Gegenmaßnahmen erschwert werden:

- Schutz des Missbrauchs einer SIM-Karte: Aktivierung des SIM-Lock
- Schutz des Softwaresystems gegen Zugriff durch einen nicht-autorisierten Nutzer: Setzen einer systemweitern PIN oder eines systemweiten Passwortes
- Schutz vor Installation von Android Anwendungen durch nicht-autorisierte Dienste: Installation nur durch eine vertrauenswürdige und durch einen Vertrauensanker durch Hardware überprüfbare Installationsanwendung (Secure Booting).

Minimierung und Hardening der Systemprozesse

Diese Empfehlungen beziehen sich auf den Linux-Kernel von Android OS:

- Schutz vor nicht-autorisierten Zugriff via USB: Der USB Debugging Mode sollte deaktiviert bzw. Zugriffsrechte für dessen Nutzung nur für bestimmte und identifizierbare USB-Geräte gewährt werden,
- Im Fall einer dedizierten Nutzung des Android OS Gerätes sollten nicht benötigte Module des Linux-Kernels deaktiviert bzw. entfernt werden.
- Reduzierung der Code-Komplexität der Systemdienste, indem nicht benötigte Abhängigkeiten zwischen Bibliotheken entfernt werden oder gehärtete Software-Bibliotheken verwendet werden.
- Reduzierung von Autorisierungen für Systemdienste wie der Anforderung einen Systemdienst unter dem Nutzer „root“ auszuführen.

Um diese Empfehlungen umzusetzen ist es notwendig, die Konfiguration des Android OS Linux-Kernel entsprechend zu analysieren und eine angepasste Version zu erstellen, prüfen und einzusetzen. Der Software-Code von Systemdiensten sollte in seinen Kontroll- und Datenflüssen überprüft werden, um diese Beziehungen und weitere bisher unbekannte Schwachstellen zu entdecken. Für eine Härtung ist ein Prinzip die des „Least Privilege“ [Saltzer].

Sicherer Speicher

Für die Maßnahme relevante Schutzziele:

- Zugriffsschutz auf Nutzerdaten (O2.1)
- Zugriffskontrolle auf Schnittstellen (O2.2)
- Vertraulichkeit sensibler Daten (O3.1)
- Erfüllung der Datenschutzanforderungen (O3.7)

Für die Maßnahme relevante Bedrohung:

- Modifikation der verarbeiteten Daten innerhalb der App (T1.4)
- Datenabfluss der eID-PIN (T1.7)
- Informationsabfluss von eID Informationen (T1.10)
- Veränderung von Übergabewerten (T2.3)
- Modifikation von App-Daten im Speicher (T2.5)

Eine weitere Empfehlung ist Schutz von Daten wie u.a. temporär ausgelagerter Speicherinhalt als Datei auf dem Dateisystem von Android OS. Während ein sicherer Bootmechanismus Integritätsverletzungen von Android OS Anwendungen erkennt, schützt dieser nicht vor einer Verletzung der Vertraulichkeit von Daten, wie bspw. einer PIN. Obwohl Android OS ab Version 3.0 eine Android Device Encryption unterstützt, ist dies nicht ausreichend. Diese Sicherheitsfunktionalität von Android OS verschlüsselt nur den Verzeichnisbaum */data*. Andere Dateien wie Systemdateien liegen weiterhin unverschlüsselt vor. Daher sollte der gesamte Verzeichnisbaum / verschlüsselt werden.

Zu dessen Umsetzung kann ein dediziertes Android Device Encryption in den Linux-Kernel von Android OS integriert werden, um direkt und sowohl für den Nutzer als auch für die Anwendungen Dateien transparent zu ver- und entschlüsseln. Das Android Device Encryption Modul sollte einen hardwarebasierten Schlüsselspeicher verwenden, was eine Interaktion mit dem Prozess des sicheren Bootings erfordert. Letzteres soll sicherstellen, dass nur ein Android Device Encryption Modul, dessen Integrität und Zurechenbarkeit geprüft wurde, Zugriff auf diese (kryptographischen) Schlüssel hat.

Schutz von sicherheitsrelevanten Anwendungen und Löschen des Speichers

Für die Maßnahme relevante Schutzziele:

- Zugriffsschutz auf Nutzerdaten (O2.1)

- Zugriffskontrolle auf Schnittstellen (O2.2)
- Vertraulichkeit sensibler Daten (O3.1)
- Erfüllung der Datenschutzanforderungen (O3.7)

Für die Maßnahme relevante Bedrohung:

- Modifikation der verarbeiteten Daten innerhalb der App (T1.4)
- Datenabfluss der eID-PIN (T1.7)
- Informationsabfluss von eID Informationen (T1.10)
- Veränderung von Übergabewerten (T2.3)
- Modifikation von App-Daten im Speicher (T2.5)

Der Schutz von sicherheitsrelevanten Anwendungen, wie einem eID Client und Systemprozessen, vor einem Beenden durch den Android OS Dienst *Ashmen* ist eine weitere Empfehlung. Weiterhin sollte der Speicherbereich, der von sicherheitsrelevanten Anwendungen verwendet wurde, nach Beenden der betreffenden sicherheitsrelevanten Anwendung speziell gelöscht werden, damit andere Anwendungen, denen dieser Speicherbereich zugeordnet wird, auf frühere Daten wie bspw. die PIN nicht zugreifen können. Dazu wird empfohlen *Ashmen* um Funktionalitäten gegenüber einer solchen Schwachstelle für einen Denial-of-Service und einem solchen Informationsabfluss zu überarbeiten. Eine Markierung von sicherheitsrelevanten Anwendungen und Prozessen ist unterstützend.

Separation von Systemprozessen durch Type Enforcement

Für die Maßnahme relevante Schutzziele:

- Zugriffskontrolle auf Schnittstellen (O2.2)
- Schutz vor Manipulation der App (O4.1)
- Robustheit gegenüber Denial-of-Service Angriffen (O5.1)

Für die Maßnahme relevante Bedrohung:

- App verschafft sich erweiterte Rechte auf Anwendungsebene (T1.13)
- Veränderung des Ausführungsumgebung (T2.2)

Um eine Ausnutzung von Sicherheitsschwachstellen in Native Code zu reduzieren, ist dessen Ausführbarkeit einzuschränken. Optionen dazu sind eine Anpassung des Android OS Linux-Kernels an Mandatory Access Control (MAC) und Type Confinement der Systemdienste. Damit werden nicht alle Systemdienste unter dem Nutzer *root* ausgeführt, d.h. sie haben nicht uneingeschränkte Zugriffsrechte. Anstatt dessen erhalten die Systemdienste nur die Zugriffsrechte, die sie tatsächlich benötigen. Obwohl dies nicht eine Ausnutzung von Sicherheitsschwachstellen in Native Code verhindert, z.B. in Software-Bibliotheken und die Dalvik VM, so schränkt es eine Ausbreitung von resultierenden Fehlern ein.
Da einige Zugriffsrechte für Android Anwendungen Teil ihres Sourcecodes sind, sollten diese Zugriffsrechte zur Laufzeit untersucht und ggf. geändert werden. So könnte einen Trennung von kritischen und nicht-kritischen Anwendungen erreicht werden.

Literatur

[DexGuard] DexGuard URL:https://www.saikoa.com/dexguard

[FROST] Hilgers, C. ; Macht, H. ; Muller, T. ; Spreitzenbarth, M. Post-Mortem Memory Analysis of Cold-Booted Android Devices. IT Security Incident Management & IT Forensics (IMF), 2014, (Mai 2014), 62—75, ISBN: 978-1-4799-4330-2, DOI= http://dx.doi.org/10.1109/IMF.2014.8

[ProGuard] ProGuard, URL:http://developer.android.com/tools/help/proguard.html

[Saltzer] Saltzer, J.H. 1974. Protection and the control of information sharing in mutics. *COMMUN ACM* 17, 7 (July 1974), 388—402. DOI=http://dx.doi.org/10.1145/361011.361067.

[ScreenProtect] Metz, R. Wie Snapchat und Co. sicherer werden könnten, Technology Review, URL: http://www.heise.de/tr/artikel/Wie-Snapchat-und-Co-sicherer-werden-koennten-2415639.html