Raspberry PI Security Application

Thonas Herzog, Philipp Wurm

14. Juni 2017

Zusammenfassung

1 Einleitung

Diese Dokument stellt die Dokumentation des Projekts *Raspberry PI Security Application*, in weiterer Folge *RPISec* genannt, dar, das für den Kurs *Mobile und ubiquitäre Systeme* realisiert wurde. In diesem Projekt wurde eine Heimsicherheitsanwendung mit Raspberry PI, Docker und Spring realisiert, die bei einem Sicherheitsverstoß in der Lage ist, bekannte mobile Endgeräte von registrierten Benutzern über diesen Sicherheitsverstoß zu informieren.

2 Problemdarstellung

Dieser Abschnitt behandelt die Problemdarstellung, welche die Grundlage für die zu implementierende *Raspberry PI Security Application* ist. Bei einem Auslösen eines Bewegungssensors in einem Haushalt sollen alle Bewohner über ihre mobilen Endgeräte wie Handy und Tablet über den Vorfall informiert werden sowie ein Foto erhalten, das den Sicherheitsbereich, zum Zeitpunkt wann der Bewegungsmelder ausgelöst wurde, zeigt. Des weiteren soll es zu jedem Zeitpunkt möglich sein sich ein aktuelles Foto des Sicherheitsbereichs über ein mobiles Endgerät zu beziehen.

Da es sich um eine Sicherheitsanwendung handelt, soll die Benutzerverwaltung sowie die Authentifizierung *In-House* gehalten werden, also die Sicherheitsanwendung selbst soll in der Lage sein die Benutzer zu verwalten und die Authentifizierungen durchzuführen. Da sich die mobilen Endgeräte in irgendwelchen Netzen ans Internet anbinden können, wie zum Beispiel über einen Mobilfunkanbieter, Internetanbieter oder öffentlichen *Hot-Spot*, wird ein *Messaging* Dienst benötigt über den die mobilen Endgeräte erreicht werden können. Dieser muss es erlauben, dass die Benutzerverwaltung von einem anderen Dienst übernommen werden kann, da wir diesen *Messaging* Dienst nicht vertrauen wollen und daher den *Messaging* Dienst auch nicht die Benutzerverwaltung überlassen wollen.

3 Funktionsweise

4 Hardware

Dieser Abschnitt behandelt die verwendete Hardware für *RPISec*. Für den Testaufbau wurden folgende Hardwarekomponenten verwendet.

- Ein *Raspberry PI 3 Model B*[1],
- *AZDeliveryCamRasp*[2] und ein

[1] `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`
[2] `https://az-delivery.de/products/raspberrykamerav1-3`

- *HC-SR501*[3] *Bewegungssensor.*

[3] https://www.mpja.com/download/31227sc.pdf

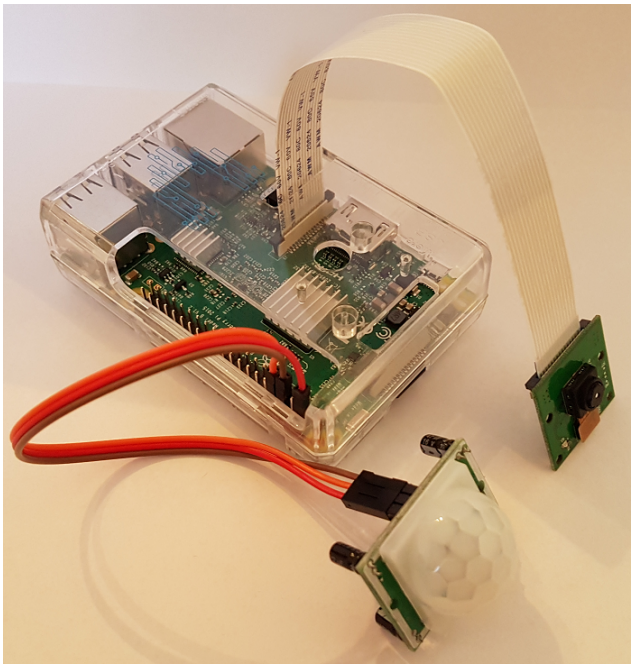

Abbildung 1: Testaufbau der Applikation

Wie in Abbildung 1 zu ersichtlich ist, wurde die Kamera über CSI *(Camera-Serial-Interface)*) und der Bewegungssensor über GPIO *(General Purpose Input/Output)* an den *Raspberry PI* angeschlossen.

5 Software

Dieser Abschnitt behandelt die verwendete bzw. implementierte Software für *RPISec*.

5.1 Betriebssysteme

Dieser Abschnitt behandelt die verwendete Betriebssysteme für den *Raspberry PI*. Die Applikation *RPISec* wurde einerseits mit dem Betriebssystem *hypriotos-rpi* und andererseits mit *Raspian* realisiert. Das Betriebssystem *hypriots* basiert auf *Debian Jessie* und wird von dem *OpenSource* Projekt *hypriot*[4] zur Verfügung gestellt wird. Das Ziel von *hypriots* ist es ein Betriebssystem für *Raspberry PI* zur Verfügung stellen, das bereits Docker vorinstalliert und betriebsbereit hat. Mit dem Betriebssystem *Raspian* muss Docker selbst installiert, wobei Docker als Paket im *Repository* zur Verfügung steht und daher sich die Installation als unkompliziert gestaltet.

Wenn Docker installiert und betriebsbereit ist, dann spielt es keine Rolle auf welchem Betriebssystem die Applikation *RPISec* betrieben wird.

Da die Applikation *RPISec* auf eine aktive Internetverbindung angewiesen ist, muss das Betriebssystem so konfiguriert werden, dass der *Raspberry PI* entweder über *Ethernet* oder *Wlan* an ein Netzwerk angebunden ist, das Zugriff auf das Internet erlaubt. In einem produktiven Betrieb muss der *Raspberry PI* über das Internet erreichbar sein, damit die mobilen *Clients* Anfragen an die gehosteten *Microservice* absetzen können.

[4] `https://blog.hypriot.com/`

5.2 Services und *Cloud*

Dieser Abschnitt behandelt die auf dem *Raspberry PI* gehosteten Services. Die Services wurden mit *Spring Boot* als *Microservices* implementiert, was möglich war, da Oracle eine ARM Implementierung der Java-JDK bereitstellt und die *Microservices* schlank implementiert wurden, sodass die zur Verfügung stehenden Ressourcen ausreichen, um diese Services auf einen *Raspberry PI* zu betreiben.

Es wurden die beiden *Microservices rpisec-auth-service* für die Benutzerverwaltung und OAuth2 Authentifizierung und *rpisec-app-service* für die Interaktion mit der Sensorik und der Interaktion mit dem *Cloud*-Diensten implementiert, wobei der *Microservice rpisec-auth-service* im Zuge des Projekts für die Lehrveranstaltung *Service Engineering* implementiert wurde. Es hätte auch ausgereicht die Benutzerverwaltung in den *Microservice rpisec-app-service* zu verpacken, obwohl dann der *Microservice* für zwei Aspekte verantwortlich gewesen wäre was im Widerspruch zu einem *Microservice* steht, der nur für einen Aspekt verantwortlich sein soll.

Die beiden *Microservices* müssen Daten persistent halten und sind daher auf eine Datenbank angewiesen, wobei im Entwicklungsbetrieb auf einen Entwicklerrechner H2 und im produktiven Betrieb auf einen *Raspberry PI* PostgreSQL verwendet wird. Die Datenbank PostgreSQL konnte verwendet werden, da PostgreSQL die ARM Architektur unterstützt.

Als *Cloud* Anbieter wurde *Google* gewählt, welcher die Plattform *Firebase* anbietet, die eine JSON-Datenbank und einen *Cloud Messaging* Dienst anbietet. Für diesen Dienst gibt es eine Java Implementierung das sogenannte *firebase-admin-sdk*, das eine API zum Interagieren mit der JSON-Datenbank und eine API zum Erstellen von Authentifizierungstoken für die *Client*-Authentifizierung bei Firebase zur Verfügung stellt. In der Java Implementierung wird zurzeit keine API für die Interaktion mit dem *Messaging* Dienst zur Verfügung gestellt, was aber kein Problem darstellt, da es sich hierbei um eine einfache Anfrage an eine *REST-API* handelt, die mit Spring *RestTemplate* durchgeführt wird.

5.3 Docker Infrastruktur

Dieser Abschnitt behandelt die Docker Infrastruktur, welche die Service und deren Abhängigkeiten hosted und verwaltet. Da der Umgang mit Docker und einer umfangreicheren Infrastruktur mit viel Shell-Skripten verbunden ist, wird das Python basierte Tool Docker-Compose verwendet, das es erlaubt eine Infrastruktur, die aus einer Menge von untereinander abhängigen Services besteht, deklarativ über eine *YAML*-Konfigurationsdatei definiert werden kann.

Der Quelltext 1 zeigt den Inhalt der *docker-compose.yml*, welche die Docker Infrastruktur für *RPISec* amd *Raspberry PI* definiert. Die in der Datei vorkommenden Textfragmente im Format *${...}* stellen Variablen dar, die *Docker Compose* entweder aus einer Datei mit dem Namen *.env*, die auf derselben Ebene wie die *docker-compose.yml* platziert werden muss, oder aus den Umgebungsvariablen des Benutzers, mit dem die Infrastruktur erstellt wird, auflöst. Sollten Variablen nicht auflösbar sein, so wird eine entsprechende Meldung auf die Konsole ausgegeben.

Quelltext 1: docker-compose.yml für RPISec am *Raspberry PI*

```
version: "2.1"
services:
    rpisec-app-db:
        container_name: rpisec-app-db
        image: tobi312/rpi-postgresql:9.6
        environment:
            - POSTGRES_USER=rpisec-app
            - POSTGRES_PASSWORD=rpisec-app
            - POSTGRES_DATABASE=rpisec-app
        mem_limit: 64m
        cpu_shares: 2
        volumes:
            - ${APP_DB_VOLUME}:/var/lib/postgresql/data:rw
```

```
    rpisec-oauth-db:
        container_name: rpisec-oauth-db
        image: tobi312/rpi-postgresql:9.6
        environment:
            - POSTGRES_USER=rpisec-oauth
            - POSTGRES_PASSWORD=rpisec-oauth
            - POSTGRES_DATABASE=rpisec-oauth
        mem_limit: 64m
        cpu_shares: 2
        volumes:
            - ${OAUTH_DB_VOLUME}:/var/lib/postgresql/data:rw
    rpisec-app:
        container_name: rpisec-app
        build:
            context: ./app
            args:
               - VIDEO_GUID=44
               - UID=1000
        volumes:
            - ${APP_APP_VOLUME}:/home/rpisec/app:rw
            - ${APP_CONF_VOLUME}:/home/rpisec/conf:ro
            - ${APP_LOG_VOLUME}:/home/rpisec/log:rw
            - ${APP_IMAGE_VOLUME}:/home/rpisec/image:rw
        environment:
            - APP_JAVA_OPTS=-Xms128m -Xmx200m
            - RPISEC_VERSION=${RPISEC_VERSION}
        mem_limit: 200m
        cpu_shares: 4
        privileged: true
        depends_on:
            - rpisec-app-db
    rpisec-oauth:
        container_name: rpisec-oauth
        build:
            context: ./oauth
        volumes:
            - ${OAUTH_APP_VOLUME}:/home/oauth/app:rw
            - ${OAUTH_CONF_VOLUME}:/home/oauth/conf:ro
            - ${OAUTH_LOG_VOLUME}:/home/oauth/log:rw
        environment:
            - OAUTH_JAVA_OPTS=-Dadmin.email=fh.ooe.mus.rpisec@gmail.com -Xms128m -Xmx200m
            - RPISEC_VERSION=${RPISEC_VERSION}
        mem_limit: 200m
        cpu_shares: 4
        depends_on:
            - rpisec-oauth-db
            - rpisec-app
    rpisec-nginx:
        container_name: rpisec-nginx
        image: rpisec-nginx
        build:
            context: ./nginx
        volumes:
            - ${NGINX_LOG_VOLUME}:/var/log/nginx:rw
            - ${NGINX_CERT_VOLUME}:/cert:ro
        mem_limit: 64m
        cpu_shares: 4
        ports:
            - 443:443
            - 80:80
        depends_on:
            - rpisec-app-db
            - rpisec-oauth-db
            - rpisec-app
            - rpisec-oauth
```

5.4 Sensorik

5.5 Mobiler *Client*