Wydział Podstawowych Problemów Techniki
Politechnika Wrocławska

Kontrola dostępu do smartfona przy wykorzystaniu urządzenia smartband

Karolina Bąk
Nr indeksu: 244917

Praca inżynierska napisana
pod kierunkiem
Dr inż. Przemysława Błaśkiewicza

Politechnika
Wrocławska

Wrocław 2021

Spis treści

Wstęp

Praca swoim zakresem obejmuje projekt systemu stałej autoryzacji wykorzystujący prostą analizę behawioralną w czasie rzeczywistym na podstawie danych gromadzonych przez inteligentną opaskę dla smartfonów działających pod Androidem. Klucze bezpieczeństwa są rzadkim zjawiskiem w mobilnych systemach. Niska popularność tego rozwiązania może wynikać z ceny tych urządzeń, często niekompatybilnych ze smartfonem metodach połączenia oraz zastępowalności innymi metodami wieloskładnikowej autoryzacji.

Celem pracy jest zaprojektowanie i stworzenie aplikacji o następujących założeniach funkcjonalnych:

- Aplikacja wykorzystuje inteligentną opaskę (pot. smartband) jako klucz bezpieczeństwa;
- Aplikacja pozwala blokować dostęp do innych aplikacji;
- Aplikacja komunikuje się z inteligentną opaską przy wykorzystaniu protokołu Bluetooth Low Energy;
- Aplikacja analizuje zachowanie użytkownika w czasie rzeczywistym;
- Aplikacja działa w tle;
- Aplikacja pozwala wybrać, które aplikacje obejmie ochroną;
- Wszystkie dane użytkownika są przechowywane lokalnie;
- Aplikacja jest odporna na popularne ataki.

Praca składa się z sześciu rozdziałów. W rozdziale 2 przedstawiono dogłębnie problem autoryzacji przy użyciu sprzętowych kluczy w smartfonach oraz braku prywatności wrażliwych danych gromadzonych przez smartbandy. Omówiono szczegółowo dane pobierane z sensorów w opasce oraz smartfonie. Scharakteryzowano zdarzenia, przy których ograniczany jest dostęp do urządzenia. Opisano rozwiązania podjęte w celu uniemożliwienia sabotażu aplikacji. Przeprowadzono analizę porównawczą istniejących rozwiązań z realizowanym systemem.

W rozdziale 3 przedstawiono szczegółowy projekt aplikacji w notacji UML. Wykorzystano diagramy przypadków użycia, komponentów oraz stanów. Omówiono dokładnie projekt bazy danych. Wyczerpująco opisano protokoły komunikacji z opaską. Opisano w pseudokodzie i omówiono algorytmy blokujące dostęp do aplikacji.

W rozdziale 4 określono technologie użyte w implementacji aplikacji: wybrany język programowania, wykorzystywane biblioteki, model opaski oraz typ bazy danych. Przedstawiono dokumentację techniczną wybranych kodów źródłowych.

W rozdziale 5 przedstawiono wymagania aplikacji co do środowiska. Określono także sposób instalacji oraz konfiguracji aplikacji. Rozdział zawiera również przykłady działania dla użytkownika.

Końcowy rozdział jest podsumowaniem uzyskanych wyników.

Analiza zagadnienia

W niniejszym rozdziale omówiono mankamenty związane z uwierzytelnianiem na urządzeniach mobilnych oraz kwestie prywatności wrażliwych danych pochodzących z urządzeń typu smartband. Przedstawiono zarys systemu. Określono, jakie dane będą rejestrowane przez aplikację oraz cel ich gromadzenia. Określono sposoby analizy danych pod kątem wykrywania sytuacji, w których smartfon jest pozostawiony bez nadzoru. Opisano mechanizmy podjęte w celu zabezpieczenia systemu oraz przechowywanych danych. Porównano istniejące rozwiązania z proponowanym w pracy, wskazując na innowacje oraz różnice.

2.1 Przedstawienie problemu

Kwestia bezpieczeństwa smartfonów jest w dzisiejszych czasach niezwykle ważną sprawą. Powszechnie stosowane metody ograniczenia dostępu, takie jak uwierzytelnienie przy użyciu hasła bądź odcisku palca, są niewystaraczające. Szczególnie jest to widoczne przy atakach fizycznych, gdzie na przykład można:

- wykorzystać nieprzytomność użytkownika, by użyć jego odcisku palca;
- poznać hasło w postaci symbolu na podstawie śladów palców na ekranie dotykowym;
- uzyskać dostęp, gdy użytkownik pozostawi odblokowane urządzenie bez opieki.

Zważywszy na fakt, iż telefony komórkowe stają się coraz bardziej powszechne [13] oraz zastępują komputery jako urządzenia wykorzystywane do łączenia się z siecią (około połowa ruchu sieciowego pochodzi z urządzeń mobilnych [4]), przechowują one wiele wrażliwych danych o swoich użytkownikach. Dlatego koniecznością jest wprowadzenie dodatkowego systemu zabezpieczeń, w szczególności wykorzystanie uwierzytelnienia wielopoziomowego, w celu zabezpieczenia urządzenia przed dostępem osób niepowołanych. Często można spotkać się z wykorzystaniem smartfonów jako autentykatorów (kody SMS oraz dedykowane aplikacje) do innych systemów informatycznych. Natomiast do autoryzacji dostępu do smartfona nie wykorzystywane są żadne dodatkowe autentykatory. Głównymi przeszkodami do ich implementacji są:

- niepraktyczność;
- monofunkcyjność.

Nieporęczność fizycznych kluczy bezpieczeństwa objawia się szczególnie w ich formie - są to niewielkie urządzenia przypominające pamięć USB lub kartę płatniczą. Dzięki temu łatwo je zgubić lub o nich zapomnieć, co uniemożliwia użytkownikowi dostęp do systemu. Często też w smartfonie brakuje niezbędnej do odczytania klucza infrastruktury, na przykład czytnika smart cardów czy modułu NFC [6]. Do niepraktyczności tego rozwiązania przyczynia się także wyżej wynieniona monofunkcyjność kluczy. Oferują one jedynie autoryzację użytkownika przy użyciu kluczy kryptograficznych bądź jednorazowych kodów i służą do logowania na stronach internetowych oraz przy autoryzacji w niektórych aplikacjach, co znacznie ogranicza pole ich zastosowania. Dlatego potrzebne jest świeże spojrzenie na tą technologię, które pozwoli stworzyć przystępne systemy stałej autoryzacji bazujące na wielu czynnikach środowiskowych oraz wykorzystujące powszechnie używane urządzenia by skutecznie zabezpieczyć dane szerokiej bazy użytkowników smartfonów.

Platformy mobilne jako dynamicznie rozwijające się technologie wspierają wachlarz urządzeń peryferyjnych, które mogły by zostać użyte jako klucz bezpieczeństwa. Jednym z nich jest **inteligentna opaska**,

potocznie zwana “smartband” bądź “fitness tracker”. Jest to urządzenie o kształcie zegarka na rękę monitorujące aktywność użytkownika taką, jak: ilość wykonanych kroków, puls czy sen. Dane gromadzone przez opaski są przesyłane protokołem Bluetooth do aplikacji towarzyszącej udostępnionej przez producenta, skąd zostają przesłane na zewnętrzne serwery. Nie jest to bezpieczne rozwiązanie, zważywszy na: wrażliwość powyższych informacji, powiązanie ich z danymi osobowymi użytkownika oraz fakt, że mogą zostać udostępnione osobom trzecim [1]. Z tego powodu konieczny jest rozwój systemów przechowywania informacji o aktywności pochodzących z inteligentnych opasek, które zapewnią użytkownikowi prywatność i nie będą bezpośrednio powiązane z producentem danego urządzenia.

2.2 Opis aplikacji

Zniwelowanie słabych punktów autentykacji przy użyciu kluczy sprzętowych jest niezwykle ważne przy implementacji tego rozwiązania w urządzeniach mobilnych. Dlatego w pracy skupiono się na usprawnieniu poniższych niedoskonałości tej technologii:

- prawdopodobieństwo utraty urządzenia autoryzującego;
- niska powszechność kluczy sprzętowych;
- ograniczona możliwość stałej autoryzacji.

Proponowana aplikacja opiera się na wykorzystaniu smartbanda jako inteligentnego klucza sprzętowego. Autoryzacja użytkownika odbywa się poprzez analizę danych o aktywności pobieranych z opaski w krótkich odstępach czasu. Po wykryciu sytuacji, gdzie smartfon jest prawdopodobnie poza nadzorem użytkownika, następuje uruchomienie blokady wybranych aplikacji do momentu wprowadzenia poprawnego hasła w aplikacji. Uniemożliwienie dostępu dokonuje się poprzez monitorowanie, która aplikacja znajduje się na pierwszym planie systemu Android. W przypadku wykrycia niedozwolonego programu użytkownik zostaje przeniesiony do aktywności odpowiedzialnej za autoryzację.

Wybór smartbanda do pełnienia funkcji klucza został podyktowany dużą popularnością urządzeń tego typu. Na rynku dostępnych jest wiele niskobudżetowych modeli, które gromadzą dane wystarczające do dość dokładnego określenia aktywności użytkownika. Z tego powodu inteligentne opaski są idealne, by oprzeć o nie system stałej autoryzacji. Dużą zaletą smartbanda jest jego niepozorna forma, czyli zegarek na rękę. Użytkownicy noszą go przez dużą część dnia, a nawet w nocy, przez co znacznie zmniejsza się ryzyko jego utraty bądź kradzieży. Kolejnym atutem tego urządzenia jest fakt, iż pełni ono znacznie więcej funkcji niż klucz sprzętowy. Dzięki temu smartband jest znacznie bardziej praktyczny dla użytkownika.

2.2.1 Charakterystyka gromadzonych danych

Aplikacja opiera się w głównej mierze o informacje rejestrowane przez inteligentną opaskę. Zaliczają się do nich:

- liczba wykonanych kroków danego dnia;
- aktualna wartość pulsu;
- moment zaśnięcia;
- moment zdjęcia opaski.

Powyższe informacje pozwalają określić stan fizyczny użytkownika, co jest kluczowe dla działania aplikacji. Dodatkowo dane o aktywności są uzupełniane o wartość sensora liczącego kroki w telefonie. Dzięki temu możliwa jest detekcja sytuacji, w których osoba eksploatująca może nie być w stanie nadzorować swojego telefonu, na przykład podczas snu bądź po pozostawieniu go na biurku w pracy. Przechowywane są także podstawowe informacje o opasce takie, jak: adres MAC oraz stan baterii. Umożliwia to ponowne połączenie z opaską oraz monitorowanie stanu urządzenia w aplikacji.

Oprócz informacji o aktywności aplikacja przechowuje także listę zainstalowanych aplikacji. Pozwala to użytkownikowi dostosować jej działanie do własnej preferencji. Najważniejszą przechowywaną informacją jest hasz hasła użytkownika, które jest wymagane do odblokowania dostępu do wybranych wcześniej aplikacji.

2.2.2 Analiza aktywności

Ważną częścią pracy jest wykrywanie sytuacji, w których smartfon jest poza nadzorem. Aby było to możliwe aplikacja bada aktywność użytkownika, korzystając z określonych w powyższej podsekcji danych, pod kątem czterech zdarzeń:

- opaska traci połączenie ze smartfonem;
- użytkownik zasypia;
- występują znaczne rozbieżności pomiędzy zarejestrowanymi krokami;
- użytkownik zdejmuje opaskę.

Utrata połączenia wykrywana jest na podstawie metod nasłuchujących zmiany w statusie połączenia Bluetooth. Sen wykrywany jest poprzez otrzymanie powiadomienia z opaski o zarejestrowaniu odpowiedniego zdarzenia. Rozbieżności w rejestrowanych krokach monitorowane są przez porównanie tempa wzrostu kroków mierzonych przez smartbanda oraz telefon, a zdjęcie opaski rozpoznaje się poprzez brak wykrywanego pulsu, bądź poprzez otrzymanie powiadomienia ze smartbanda. W przypadku wykrycia jednej z powyższych sytuacji następuje automatyczne uruchomienie blokady aplikacji.

2.2.3 Zabezpieczenie aplikacji

Aby proponowany system zapewniał ochronę przed dostępem przez osoby niepowołane musi działać nieprzerwanie i być odporny na wyłączenie go przez atakującego. W tym celu usługi aplikacji są zaimplementowane jako *Foreground Service*, by działać stale w tle w zgodzie z limitami obowiązującymi od Androida Oreo[10]. Wykorzystano technologię *Wake Lock*[12] w celu umożliwienia aplikacji pozostania w stanie pełnej sprawności w przypadku, gdy telefon przechodzi w *Doze Mode*[11]. Wdrożono także *BroadcastReceiver*, który jest odpowiedzialny za monitorowanie restartów urządzenia. Po wykryciu ukończonego uruchomienia smartfona, usługa blokująca oraz gromadząca dane są restartowane według stanu sprzed wyłączenia urządzenia.

System jest także odporny na najpopularniejsze podatności w aplikacjach mobilnych związanych z danymi medycznymi[3]. Dzięki lokalnemu przechowywaniu informacji zapewniona jest odporność na ataki za pośrednictwem sieci. Aplikację zabezpieczono przed *Intent spoofing*, dzięki wykorzystaniu jedynie dokładnie sprecyzowanych Intentów oraz zabezpieczeniu komponentów przed exportem do innych aplikacji. By zapewnić bezpieczeństwo gromadzonych danych baza danych oraz plik przechowujący hasło zostały zaszyfrowane przy użyciu algorytmu szyfrowania AES. Natomiast mniej ważne informacje są przechowywane w *SharedPreferences*, do których dostęp ma tylko projektowany system.

2.3 Analiza porównawcza istniejących systemów

Na rynku znajduje się wąskie grono rozwiązań o podobnych funkcjonalnościach. Poniżej zaprezentowano najciekawsze z nich. Określono ich zalety oraz wady, a także porównano je z systemem zaprezentowanym w pracy.

2.3.1 Yubikey

Yubikey [18] to seria nowoczesnych kluczy sprzętowych produkowanych przez Yubico, wykorzystywanych jako część wieloskładnikowej autoryzacji bądź autentykacji bazowanej na jednorazowych hasłach w szerokim gronie serwisów internetowych oraz systemów operacyjnych. Wspierają wiele protokołów kryptograficznych i

autentykacyjnych, w tym: WebAuthn, FIDO2, U2F, smart cardy kompatybilne z PIV oraz Yubico OTP. Modele dedykowane urządzeniom mobilnym do komunikacji ze smartfonem wykorzystują moduł NFC, USB-C oraz złącze Lightning. Autoryzacja odbywa się poprzez umieszczenie klucza w złączu USB-C bądź przystawienie go do tyłu telefonu dla urządzeń z włączonym NFC.

Yubikey posiada wiele zalet. Jest wspierany przez dużą liczbę serwisów i systemów, dzięki czemu wachlarz aplikacji autentykacyjnych oraz kodów SMS czy wiadomości e-mail można zastąpić jednym urządzeniem. Pomaga uniknąć wykradnięcia haseł poprzez phishing czy przechwycenie SMSa. Jest prosty w użyciu dla użytkownika i nie wymaga ładowania. Jest również odporny na wodę oraz zgniecenie.

Dużą wadą Yubikey jest jego cena. Modele zapewniające autoryzację na smartfonach kosztują na tą chwilę minimum 45€ bez podatku VAT[17], czyli około 200 zł. Dla zwykłego użytkownika może być to zbyt duża kwota, gdy może skorzystać z darmowych wariantów dwuskładnikowej autoryzacji. Forma klucza (małe urządenie przypominające pendrive) sprzyja jego łatwemu zgubieniu, co uniemożliwia dostęp do serwisów, które z niego korzystały. By temu zaradzić producent zaleca posiadać zapasowy klucz, co wiąże się z dodatkowym wydatkiem rzędu 200 zł. Nie należy także zapominać o tym, że nie wszystkie telefony wspierają NFC oraz USB-C. Podczas, gdy rynek smartfonów dąży do wdrożenia powszechnie standardu USB-C, w przypadku NFC nie wszędzie jest on potrzebny. Ów moduł służy głównie do płatności mobilnych, dlatego na przykład w Chinach, gdzie powszechny jest system płatności przez kody QR[15], jest po prostu zbędny. Zważając na popularność chińskich telefonów na światowym rynku prawdopodobnym jest, iż nawet nowe modele nie będą wspierać technologii NFC, przez co utrudnią, a nawet uniemożliwią korzystanie z kluczy Yubikey.

W proponowanym rozwiązaniu jako klucz sprzętowy zostało wykorzystane urządzenie, które eliminuje wymienione wyżej wady Yubikey. Inteligentna opaska jest przeznaczona do noszenia na ręce, dzięki czemu ciężej ją zgubić lub ukraść. Smartband komunikuje się ze smartfonem poprzez wykorzystywany powszechnie moduł Bluetooth, co pozwoli wdrożyć system w znacznie szerszym gronie urządzeń. Kolejnym atutem wybranego urządzenia jest jego cena. Inteligentną opaskę można nabyć za mniej niż 100 zł, co sprawia, że jest przystępna dla wielu użytkowników. Najważniejszą różnicą między Yubikey a proponowanym systemem jest sposób autentykacji. Dzięki zastosowaniu opaski można stale autoryzować użytkownika bazując na wielu zmiennych czynnikach w przeciwieństwie do Yubikey, które jest jedynie nośnikiem przechowującym klucz, który jest wykorzystywany przy pojedynczych logowaniach bądź jako dodatkowa autoryzacja przy wrażliwych czynnościach.

2.3.2 Haven

Haven [7] jest darmową aplikacją open-source dla urządzeń działających pod systemem Android zaprojektowaną w celu monitorowania aktywności wokół urządzenia, korzystając z jego wbudowanych sensorów. Przy wykryciu zmian w środowisku aplikacja gromadzi zdjęcia oraz nagrania dźwięku, po czym wysyła je poprzez Signala bądź Tora do użytkownika. Aplikacja została stworzona z myślą o dziennikarzach śledczych, którzy są narażeni na ataki ze strony policji bądź innych intruzów.

Główną zaletą Haven jest to, iż potrafi zastąpić drogie fizyczne systemy bezpieczeństwa. Do korzystania z tego rozwiązania wystarczy stary telefon z Androidem oraz opcjonalnie karta SIM, by zapewnić dostęp do mobilnego Internetu. Zapewnia to użytkownikowi tani, a także łatwy w przenoszeniu system pozwalający monitorować na przykład pokój w hotelu. Pozwala to zdobyć dowody w przypadku ataku typu “evil maid”[14], czyli gdy osoba trzecia uzyskuje fizyczny dostęp do urządzenia, wykorzystując nieobecność właściciela, w celu wykradnięcia danych bądź zainstalowaniu szpiegującego oprogramowania.

Wadą Haven jest zdecydowanie fakt, że nie zapobiega ona atakom, tylko zdobywa dowody ich wystąpienia. Kolejnym mankamentem aplikacji jest jej nadmierna czułość. Wykrywane są mikroruchy smartfona oraz drobne dźwięki, przez co korzystanie z Haven w głośniejszych środowiskach, jak na przykład w biurze może wiązać się z setkami fałszywie wykrytych zdarzeń.

Zaproponowany w pracy system również skupia się na atakach wykonywanych poprzez fizyczny dostęp

do urządzenia, lecz w zupełnie inny sposób. Proponowana aplikacja ma na celu wykorzystanie danych ze smartbanda w celu zabezpieczenia smartfona, gdy użytkownik nie jest w stanie nadzorować go samodzielnie. W przeciwieństwie do Haven, które wykorzystuje telefon do zbierania informacji, ale w żadnym stopniu nie korzysta z nich żeby uniemożliwić dostęp do urządzenia. Oba rozwiązania monitorują przeróżne wydarzenia rejestrowane przez dostępne sensory. Podczas, gdy Haven skupia się na środowisku, proponowana aplikacja skupia się na samym użytkowniku. Haven również w żadnym stopniu nie analizuje zbieranych danych, gdyż jego głównym zadaniem jest jedynie raportowanie tego, co dzieje się wokoło. Z kolei proponowane rozwiązanie w pewnym stopniu bierze pod lupę gromadzone dane i na ich podstawie określa, kiedy uruchomić blokadę urządzenia.

2.3.3 Android Management API

Android Management API [8] jest częścią Android Enterprise, inicjatywy dostarczającej deweloperom narzędzi pozwalających budować rozwiązania dla przedsiębiorstw w celu zarządzania flotą mobilnych urządzeń. Program ten jest dedykowany dostawcom usług zarządzania mobilnością w przedsiębiorstwie (EMM). Deweloperzy zapewniają swoim klientom lokalną bądź opartą na chmurze konsolę EMM. Wewnątrz konsoli klienci generują tokeny rejestracji urządzeń oraz tworzą zasady zarządzania (policies). Zasada zarządzania reprezentuje grupę ustawień rządzących zachowaniem zarządzanego urządzenia oraz zainstalowanymi aplikacjami. Następnie urządzenia są zapisywane do systemu przy użyciu wcześniej stworzonych tokenów. Podczas rejestracji, każde urządzenie instaluje aplikację towarzyszącą API, Android Device Policy. Kiedy do danego urządzenia są przyporządkowane zasady, powyższa aplikacja automatycznie wdraża je.

Android Management API daje niespotykane poza aplikacjami systemowymi możliwości zarządzania urządzeniem. Pozwala między innymi na:

- wyłączenie określonych modułów komunikacji takich, jak Bluetooth, Wi-Fi, SMS, rozmowy czy USB;
- blokadę instalacji bądź dezinstalacji aplikacji;
- dostosowanie aplikacji dostępnych w sklepie Play;
- wymuszenie określonych ustawień sieci;
- masowe nadanie pozwoleń aplikacjom;
- określenie sposobów autoryzacji oraz wymogów hasła;
- włączenie aplikacji w trybie Kiosk;
- zdalne wymazanie danych z urządzenia.

Więcej informacji na temat dostępnych polityk można znaleźć w [9].

Główną wadą tego API jest brak możliwości zastosowania go poza przedsiębiorstwami. Urządzeniom korzystającym z tego rozwiązania zasady narzucane są odgórnie przez administratora, więc przy ogromnej liczbie smartfonów, gdzie każdy wymagałby innego zestawu zasad oraz ich aktualizacji na bieżąco, zarządzanie byłoby kłopotliwe. Jest to fundamentalny problem, przez który proponowany system nie może wykorzystać możliwości zabezpieczających Android Management API.

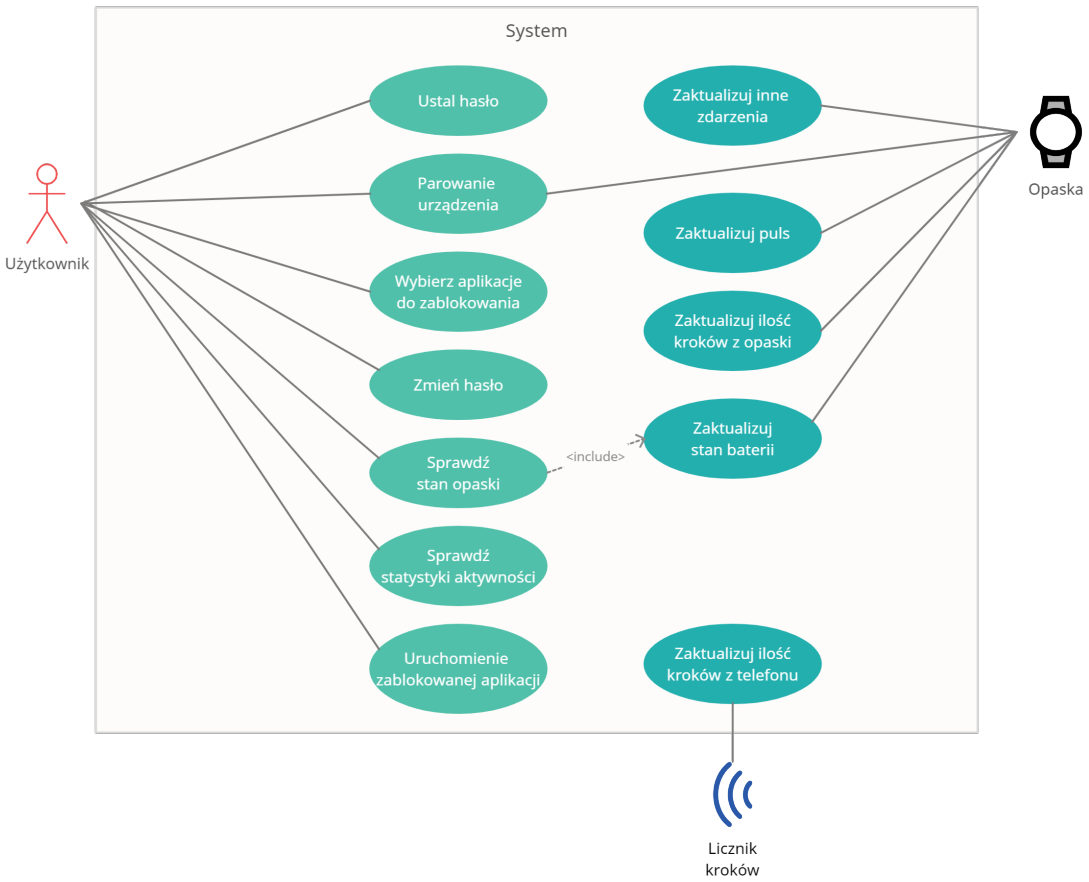
Porównując oba rozwiązania można zauważyć, iż są swoimi przeciwieństwami. Tworzony system jest z założenia czysto lokalny oraz tworzony z myślą o zwykłych użytkownikach, dlatego wszystkie mechanizmy zabezpieczające również muszą być aplikowalne bez udziału zewnętrznych serwisów, a także konfigurowalne przez użytkownika. Z kolei przy wykorzystaniu tego API konieczna jest rejestracja urządzenia w systemie EMM danego przedsiębiorstwa, a zasady dotyczące bezpieczeństwa są narzucane z góry.

Projekt aplikacji

W tym rozdziale przedstawiono szczegółowy projekt systemu korzystając z notacji UML oraz uwzględniając założenia funkcjonalne z rozdziału 1. Scharakteryzowano przypadki użycia oraz towarzyszące im scenariusze. Przedstawiono ogólną strukturę aplikacji w diagramie komponentów. Określono tryby działania aplikacji poprzez diagram stanów. Przedstawiono projekt bazy danych. Opisano dokładnie protokół komunikacji z MiBandem 3.

3.1 Przypadki użycia

Poniżej przedstawiono ogólny diagram przypadków użycia 3.1. Szczegółowe scenariusze zostały zdefiniowane w odpowiednich podsekcjach tego podrozdziału.



Rysunek 3.1: Diagram przypadków użycia w systemie.

3.1.1 Ustal hasło

W tym przypadku użytkownik aplikacji ustala hasło, które jest potrzebne przy odblokowywaniu dostępu do zablokowanych aplikacji. Aktorami są Użytkownik oraz System. Obecny przypadek użycia jest inicjowany przy pierwszym uruchomieniu aplikacji, kiedy nie istnieje jeszcze zaszyfrowany plik, w którym będzie przechowywane hasło. Po wykonaniu przypadku w systemie zostaje zarejestrowane hasło użytkownika, które będzie później wykorzystane w celu dezaktywacji blokady aplikacji. Scenariusz składa się z następującego przepływu głównego:

1. Aplikacja wyświetla formularz zawierający pola Hasło oraz Powtórz hasło.
2. Użytkownik wprowadza identyczne hasła do podanych pól oraz zatwierdza wprowadzone dane przyciskiem znajdującym się poniżej.
3. System sprawdza zgodność haseł oraz czy spełniają wymagane kryteria.
4. System tworzy zaszyfrowany plik i zapisuje w nim hasz hasła.

Alternatywnie:

3. Jeśli wprowadzone hasła nie są zgodne:
    1. Zostaje wyświetlony komunikat o błędzie.
    2. Następuje powrót do kroku numer 2 w głównym przepływie.

3.1.2 Parowanie urządzenia

W tym przypadku użytkownik aplikacji skanuje otoczenie, korzystając z modułu Bluetooth w poszukiwaniu najbliższej opaski MiBand, a następnie nawiązuje z nią pierwsze połączenie. Aktorami są Użytkownik, System oraz Opaska. Przypadek ten występuje przy pierwszym uruchomieniu systemu, kiedy zostało już ustalone hasło odblokowujące. Po zakończeniu system jest sparowany z opaską MiBand, z którą komunikacja jest kluczowym punktem działania systemu. W systemie zostaje także zapisany adres MAC opaski, dzięki czemu będzie można z łatwością ponownie połączyć się z nią. Główny przepływ składa się z następujących kroków:

1. Użytkownik naciska przycisk “Skan”.
2. System rozpoczyna skanowanie urządzeń Bluetooth Low Energy w celu znalezienia Opaski.
3. System wyświetla znalezione Opaski w formie listy.
4. Użytkownik wybiera Opaskę, do której się podłączy poprzez naciśnięcie na jej nazwę.
5. System tworzy więź z wybraną Opaską i inicjuje pierwsze połączenie.

Alternatywnie:

3. Jeśli nie znaleziono żadnej Opaski:
    1. Zostaje wyświetlony komunikat o błędzie.
    2. Następuje powrót do kroku numer 1 w głównym przepływie.

5. Jeśli wystąpi błąd połączenia z Opaską:
    1. Następuje powrót do kroku numer 4 w głównym przepływie.

3.1.3 Wybierz aplikacje do zablokowania

W tym przypadku użytkownik aplikacji wybiera z listy zainstalowanych aplikacji te, które będą blokowane przez system kiedy zostanie uruchomiona blokada. Aktorami są Użytkownik oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu System posiada informację, na których aplikacjach uruchamiać blokadę. Główny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji przechodzi do Ustawień.
2. Użytkownik wybiera opcję Zablokowane aplikacje.
3. Użytkownik zaznacza aplikację z listy, którą chce zablokować bądź odblokować.
4. System zapisuje aktualny stan aplikacji.

3.1.4 Zmień hasło

W tym przypadku użytkownik aplikacji zmienia hasło odblokowujące dostęp do zablokowanych aplikacji. Aktorami są Użytkownik oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu System posiada informację o nowym haśle, które będzie wykorzystywane od tej pory do odblokowywania dostępu. Główny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji wybiera opcję Ustawienia w dolnej nawigacji.
2. Użytkownik wybiera opcję Zmień hasło.
3. Użytkownik wpisuje odpowiednie wartości do pól Stare hasło, Nowe hasło oraz Powtórz nowe hasło oraz zatwierdza wprowadzone dane.
4. System zapisuje hasz nowego hasła w zaszyfrowanym pliku.

Alternatywnie:

3. Jeśli stare hasło jest błędne lub nowe hasło nie zgadza się z powtórzonym:
    1. System wyświetla komunikat o błędnych wprowadzonych danych.
    2. Następuje powrót do kroku numer 3 w głównym przepływie.
4. Jeśli wystąpi błąd zapisu:
    1. System wyświetla komunikat o błędzie zapisu.
    2. Następuje powrót do kroku numer 3 w głównym przepływie.

3.1.5 Sprawdź stan opaski

W tym przypadku użytkownik aplikacji sprawdza podstawowe informacje na temat opaski oraz jej stan baterii. Aktorami są Użytkownik, System oraz Opaska. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu Użytkownik zna stan baterii oraz inne informacje o Opasce, a System zyskuje odświeżony stan baterii Opaski. Głowny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji wybiera opcję Opaska w dolnej nawigacji.
2. System wyświetla zapisane wcześniej informacje o Opasce.
3. Przejdź do przypadku Zaktualizuj stan baterii

4. System aktualizuje wartość baterii na ekranie.

Alternatywnie:

2. Jeśli nie zapisano wcześniej informacji o Opasce:
    1. System wyświetla w brakujących polach wartość Nieznane.
    2. Następuje przejście do kroku numer 3 w głównym przepływie.

3. Jeśli nastąpi nagła utrata połączenia z Opaską:
    1. Pomiń następny krok w głównym przepływie.

3.1.6 Sprawdź statystyki aktywności

W tym przypadku użytkownik aplikacji sprawdza proste statystyki zarejestrowanej dziennej aktywności. Aktorami są Użytkownik oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana, nie ma uruchomionej blokady oraz Użytkownik uruchomił aplikację. Po zakończeniu Użytkownik zna ilość zarejestrowanych kroków oraz ostatnią zarejestrowaną wartość pulsu. Główny przepływ składa się z następujących kroków:

1. Użytkownik z poziomu głównego ekranu aplikacji wybiera opcję Statystyki w dolnej nawigacji.
2. System wyświetla ostatnie zapisane wartości dziennych kroków oraz pulsu.
3. System aktualizuje wyświetlane wartości, gdy zostaną zapisane nowe.

3.1.7 Uruchomienie zablokowanej aplikacji

W tym przypadku użytkownik aplikacji próbuje uruchomić aplikację z listy zablokowanych podczas, gdy uruchomiona jest blokada. Aktorami są Użytkownik i System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana oraz aplikacja pracuje w trybie blokady. Po zakończeniu Użytkownik posiada autoryzację do interakcji z blokowanymi aplikacjami, a System przechodzi w tryb monitorowania. Główny przepływ składa się z następujących kroków:

1. Użytkownik uruchamia aplikację z listy zablokowanych.
2. System przenosi Użytkownika w ekran wprowadzania hasła.
3. Użytkownik wprowadza hasło i zatwierdza.
4. System wyłącza blokadę i uruchamia usługę monitorującą.

Alternatywnie:

3. Jeśli Użytkownik wprowadził błędne hasło:
    1. System wyświetla komunikat o błędnym haśle.
    2. Następuje ponowne wykonanie kroku 3 z głównego przepływu.

3.1.8 Zaktualizuj inne zdarzenia

W tym przypadku opaska rejestruje jedno z podejrzanych zdarzeń i przesyła informację o tym do aplikacji. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje sygnał do uruchomienia blokady. Główny przepływ składa się z następujących kroków:

1. Opaska rejestruje jedno z zaprogramowanych zdarzeń.

2. Opaska przesyła kod zdarzenia do Systemu.
3. System porównuje otrzymany kod z zapisanymi kodami podejrzanych zdarzeń.
4. System uruchamia blokadę.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. Następuje przejście do kroku numer 4 w głównym przepływie.
3. Jeśli otrzymany kod nie znajduje się na liście podejrzanych:
    1. Pomiń następny krok głównego przepływu.

3.1.9 Zaktualizuj puls

W tym przypadku opaska wykonuje automatyczny pomiar pulsu i przesyła jego wartość do aplikacji. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje aktualne dane na temat pulsu do analizy stanu użytkownika. Główny przepływ składa się z następujących kroków:

1. Opaska mierzy puls.
2. Opaska przesyła zarejestrowaną wartość do Systemu.
3. System zapisuje otrzymaną wartość.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. System uruchamia blokadę.
3. Jeśli otrzymana wartość wynosi 0:
    1. System uruchamia blokadę.

3.1.10 Zaktualizuj ilość kroków z opaski

W tym przypadku opaska rejestruje wykonane kroki oraz przesyła uaktualnioną wartość dziennych wykonanych kroków do aplikacji. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje aktualne dane na temat wykonanych kroków, które zostaną wykorzystane później do analizy zachowania użytkownika. Główny przepływ składa się z następujących kroków:

1. Opaska rejestruje wykonanie kroków.
2. Opaska przesyła zaktualizowaną wartość dziennych kroków do Systemu.
3. System zapisuje otrzymaną wartość.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. System uruchamia blokadę.
3. Jeśli otrzymana wartość jest równa ostatniemu pomiarowi:
    1. System nie zapisuje otrzymanej wartości.

3.1.11 Zaktualizuj stan baterii

W tym przypadku aplikacja odczytuje aktualny stan baterii z opaski. Aktorami są Opaska oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywne połączenie między nią a aplikacją. Po zakończeniu System zyskuje aktualną wartość stanu baterii. Główny przepływ składa się z następujących kroków:

1. System przesyła do Opaski zapytanie o stan baterii.
2. Opaska przesyła do Systemu aktualny stan baterii.
3. System zapisuje nową wartość stanu baterii.

Alternatywnie:

2. Jeśli nastąpi nagła utrata połączenia:
    1. System uruchamia blokadę.

3.1.12 Zaktualizuj ilość kroków z telefonu

W tym przypadku sensor licznika kroków w smartfonie rejestruje wykonane kroki, a następnie informuje aplikację o aktualizacji swojej wartości. Aktorami są Licznik kroków oraz System. Przypadek ten występuje, gdy istnieje plik z hasłem, opaska została sparowana i jest aktywna usługa monitorująca zachowanie użytkownika. Po zakończeniu System zyskuje informację na temat wykonanych kroków, która zostanie później wykorzystana do analizy zachowania użytkownika.

1. Licznik kroków rejestruje wykonanie kroków.
2. System odczytuje zaktualizowaną liczbę kroków Licznika kroków.
3. System zapisuje odczytaną wartość.

3.2 Diagram komponentów

W tej sekcji należy przedstawić diagramy komponentów dla odpowiednich elementów systemu zidentyfikowane na podstawie wcześniejszych rozważań

3.3 Diagramy stanów

W tej sekcji należy przedstawić diagramy stanów w których może znaleźć się system. Diagramy te są szczególnie istotne przy projektowaniu systemów czasu rzeczywistego.

3.4 Projekt bazy danych

W tej sekcji należy przedstawić projekt bazy danych. Należy omówić wycinek rzeczywistości i odpowiadające mu zidentyfikowane elementy systemu, których wartości będą podlegać utrwalaniu. Należy przedyskutować wybór typów danych dla atrybutów poszczególnych obiektów.

3.5 Komunikacja z MiBand 3

W projektowanym systemie główną rolę gra inteligentna opaska. Komunikuje się ona ze smartfonem przy użyciu Bluetooth Low Energy protokołem ATT korzystając z GATT. BLE w porównaniu do klasycznego połączenia Bluetooth wykorzystuje znacznie niższe zasoby energii zachowując podobny zasięg, dzięki czemu znalazło szerokie zastosowanie w urządzeniach peryferyjnych. W poniższych podrozdziałach opisano pokrótce pojęcia ATT i GATT oraz dokładnie przedstawiono zaimplementowany protokół komunikacji z opaską Mi-Band 3.

3.5.1 ATT

Protokół Attribute umożliwia urządzeniu, określonemu jako *serwer*, odsłonić zbiór atrybutów i powiązanych z nimi wartości urządzeniu równorzędnemu, określonemu jako *klient*. Atrybuty odsłonięte przez serwer mogą być odkryte, odczytane bądź nadpisane przez klienta, a także mogą być rozgłaszane przez serwer w ramach powiadomienia lub zasygnalizowania. Atrybut jest dyskretną wartością o trzech właściwościach powiązanych ze sobą:

- typie,
- uchwycie,
- zestawie pozwoleń, które są zdefiniowane przez specyfikację wyższej warstwy wykorzystującą dany atrybut.

Typ atrybutu określa, co reprezentuje dany atrybut poprzez UUID (Universally Unique Identifier), ktore może zostać utworzone przez każdego, a następnie zostać opublikowane. Pozwala to rozpoznać atrybut niezależnie od nadanego mu przez serwer uchwytu. Uchwyt atrybutu jest unikalną, niezerową, 16-bitową wartością, która jednoznacznie identyfikuje dany atrybut w obrębie serwera, pozwalając klientowi odnieść się do niego podczas operacji odczytu oraz zapisu. Pozwolenia mogą być nadane atrybutowi w celu ograniczenia klientowi dostępu do zapisu lub odczytu.

Urządzenie może jednocześnie implementować zarówno rolę serwera jak i klienta oraz obie role mogą funkcjonować współbieżnie oraz komunikować się między sobą. Na każdym urządzeniu Bluetooth może znajdować się maksymalnie jedna instancja serwera.

Wszystkie prośby protokołu Attribute są przesyłane poprzez *nosiciela ATT*. Między urządzeniami może być wielu nosicieli, gdzie każdy z nich korzysta z osobnego kanału L2CAP oraz może mieć inną konfigurację. W przypadku BLE, wykorzystywany jest pojedynczy nosiciel ATT, który używa stałego kanału dostępnego od ustanowienia połączenia ACL. Można jednak skonfigurować dodatkowych nosicieli używając L2CAP. Więcej informacji na temat protokołu ATT można znaleźć w [2].

3.5.2 GATT

Profil Generic Attribute (GATT) definiuje framework wykorzystujący protokół Attribute, określający procedury i formaty danych znajdujących się wewnątrz profilu. Zdefiniowane procedury obejmują odkrywanie, odczyt, zapis, powiadamianie oraz sygnalizację. Profil ten został zaprojektowany do wykorzystania przez aplikacje bądź inny profil, aby umożliwić klientowi komunikację z serwerem poprzez opakowanie protokołu ATT w bardziej przystępną formę.

Profil GATT określa strukturę, w której odbywa się wymiana danych. Najwyższym poziomem jest profil zawierający liczne *usługi* będące zbiorem danych oraz przypisanych im zachowań niezbędnych do zapewnienia określonej funkcji. Usługi składają się z *charakterystyk*, z których każda zawiera określoną wartość oraz opcjonalne informacje na jej temat. Usługi oraz charakterystyki wraz ze swoimi komponentami zawierają dane profilu, które są przechowywane w Atrybutach na serwerze. Dzięki wykorzystaniu określonej struktury danych przez GATT możliwe jest przeglądanie dostępnych Usług oraz Charakterystyk, nawet gdy klient nie jest wyspecjalizowany pod dany serwer. Więcej informacji na temat GATT można znaleźć w [2].

3.5.3 Wykorzystane usługi i charakterystyki

Do uzyskania potrzebnych danych z opaski zostały użyte poniżej opisane usługi. Dla każdej z nich przedstawiono listę wykorzystanych charakterystyk z krótkim opisem, za co odpowiadają. Poniższe wartości zostały zidentyfikowane na podstawie przechwytywania pakietów ATT komunikacji MiBanda 3 z aplikacją Gadgetbridge [16] przy użyciu open-sourcowego sniffera Wireshark [5] oraz analizie kodu źródłowego Gadgetbridge.

Usługa 0000fee0-0000-1000-8000-00805f9b34fb

Jest to usługa odpowiadająca w głównej mierze za podstawowe funkcjonalności opaski MiBand 3. Pozwala zmodyfikować ustawienia urządzenia, zapisać informacje o użytkowniku oraz odczytać dane o stanie baterii i aktywności użytkownika. Jest to najczęściej wykorzystywana usługa w aplikacji.

Charakterystyka	Opis
00000003-0000-3512-2118-0009af100700	Charakterystyka pozwalająca na konfigurację ustawień opaski.
00000006-0000-3512-2118-0009af100700	Charakterystyka zawierająca informacje o stanie baterii opaski.
00000007-0000-3512-2118-0009af100700	Charakterystyka przechowująca liczbę wykonanych kroków danego dnia.
00000008-0000-3512-2118-0009af100700	Charakterystyka zawierająca dane użytkownika.
00000010-0000-3512-2118-0009af100700	Charakterystyka zawierająca informacje na temat zdarzeń wykrywanych przez opaskę.
00002a2b-0000-1000-8000-00805f9b34fb	Charakterystyka przechowująca aktualną datę i godzinę,

Tablica 3.1: Wykorzystane charakterystyki z usługi 0000fee0-0000-1000-8000-00805f9b34fb.

Usługa 0000fee1-0000-1000-8000-00805f9b34fb

Jest to usługa zawierająca przede wszystkim charakterystykę wykorzystywaną w procesie autentykacji połączenia oraz parowania. W usłudze tej znajduje się też sporo niezidentyfikowanych charakterystyk, które nie są wymagane do działania aplikacji. Dlatego więc zostały pominięte.

Charakterystyka	Opis
00000009-0000-3512-2118-0009af100700	Charakterystyka wykorzystywana do autentykacji połączenia między opaską a klientem

Tablica 3.2: Wykorzystane charakterystyki z usługi 0000fee1-0000-1000-8000-00805f9b34fb.

Usługa 0000180d-0000-1000-8000-00805f9b34fb

Jest to usługa zdefiniowana przez Bluetooth Special Interest Group odpowiadająca za komunikację między sensorem akcji serca a innym klientem GATT. Za jej pomocą można uzyskać informację o pulsie użytkownika oraz skonfigurować automatyczne pomiary.

Charakterystyka	Opis
00002a37-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu aktualnej wartości pulsu użytkownika.
00002a39-0000-1000-8000-00805f9b34fb	Charakterystyka odpowiedzialna za konfigurację sensora akcji serca w opasce.

Tablica 3.3: Wykorzystane charakterystyki z usługi 0000180d-0000-1000-8000-00805f9b34fb.

Usługa 0000180a-0000-1000-8000-00805f9b34fb

Jest to usługa również zdefiniowana przez Bluetooth Special Interest Group. Odpowiada za dostarczenie informacji o urządzeniu. W tym wypadku jest to informacja o numerze seryjnym opaski oraz wersjach sprzętu i oprogramowania.

Charakterystyka	Opis
00002a25-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu numeru seryjnego opaski.
00002a27-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu wersji sprzętowej opaski.
00002a28-0000-1000-8000-00805f9b34fb	Charakterystyka wykorzystywana do odczytu wersji oprogramowania opaski.

Tablica 3.4: Wykorzystane charakterystyki z usługi 0000180a-0000-1000-8000-00805f9b34fb.

3.5.4 Autentykacja połączenia

Aby konfigurować oraz odczytywać informacje o aktywności z MiBanda wymagana jest prosta autoryzacja. W przeciwnym razie ma dostępu do większości charakterystyk urządzenia, a połączenie zostanie zerwane po 30 sekundach. Sekwencja autentykacyjna wygląda w następujący sposób. Najpierw do opaski należy wysłać klucz, który zostanie wykorzystany do autentykacji połączenia. Wykonuje się to przy pierwszym połączeniu z urządzeniem. Następnie wysyłana jest prośba do MiBanda o podanie losowej liczby. Po jej otrzymaniu należy zaszyfrować ją algorytmem AES korzystając z klucza podanego przy parowaniu opaski, odesłać i przesłać aktualną datę do odpowiedniej charakterystyki. Po otrzymaniu potwierdzenia połączenie jest zatwierdzone i można przejść do sekwencji konfigurującej. Aktualna data przesyłana jest jako tablica bajtów, gdzie wartości to: rok $\wedge$ 0xff, (rok $\gg$ 8) $\wedge$ 0xff, miesiąc $\wedge$ 0xff, dzień miesiąca $\wedge$ 0xff, godzina $\wedge$ 0xff, minuty $\wedge$ 0xff, sekundy $\wedge$ 0xff, dzień tygodnia $\wedge$ 0xff oraz 0.

Algorithm 3.1: Parowanie z MiBandem 3

Data: Tablica bajtów zawierająca klucz wykorzystywany do szyfrowania K, Zmienna logiczna *pair* mówiąca, czy urządzenie jest sparowane

```
1  begin
2  |  authorisationChar ← 00000009 − 0000 − 3512 − 2118 − 0009af100700
3  |  Włącz powiadomienia na authorisationChar
4  |  if pair then
5  |  |  Wyślij do authorisationChar ⟶ ByteArray(0x01, 0x00) + K
6  |  |  if Otrzymano odpowiedz ← ByteArray(0x10, 0x01, 0x01) then
7  |  |  |  Przejdź do autentykacji
8  |  |  end if
9  |  end if
10 end
```

Algorithm 3.2: Autentykacja z MiBandem 3

Data: Tablica bajtów zawierająca klucz wykorzystywany do szyfrowania K

```
1  begin
2  |  authorisationChar ← 00000009 − 0000 − 3512 − 2118 − 0009af100700
3  |  timeChar ← 00002a2b − 0000 − 1000 − 8000 − 00805f9b34fb
4  |  Włącz powiadomienia na authorisationChar
5  |  Wyślij do authorisationChar ⟶ ByteArray(0x02, 0x00)
6  |  if Otrzymano odpowiedz ← ByteArray(0x10, 0x02, 0x01, ...) then
7  |  |  numToEncrypt ← odpowiedz [3 : 19]
8  |  |  Wyślij do authorisationChar ⟶ ByteArray(0x03, 0x00) + AES(K, numToEncrypt)
9  |  |  Wyślij do timeChar aktualną datę
10 |  end if
11 |  if Otrzymano odpowiedz ← ByteArray(0x10, 0x03, 0x01, ...) then
12 |  |  Przejdź do konfiguracji
13 |  end if
14 end
```

3.5.5 Sekwencja konfigurująca

Po udanej autoryzacji połączenia należy skonfigurować działanie opaski. Odbywa się to za pomocą sekwencji operacji zapisu, głównie do charakterystyki o UUID równym “00000003-0000-3512-2118-0009af100700” oraz włączeniu powiadamiania na odpowiednich charakterystykach. Po skonfigurowaniu urządzenie będzie automatycznie przesyłać odpowiednie dane o aktywności w najmniejszych możliwych odstępach czasu. Czynności konfiguracyjne zostały zidentyfikowane na podstawie porównania przechwyconych pakietów ATT z kodem źródłowym Gadgetbridge. Poniżej opisano szerzej niektóre z wysyłanych danych, które były zbyt obszerne, aby przedstawić je w pseudokodzie.

Format daty jest przestawiony jako tablica bajtów, zawierająca każdy znak z “dd/MM/yyyy” skonwertowany na bajt. Wartość celu fitness obliczana jest jako dwuelementowa tablica bajtów, gdzie wartości to wartość celu $\wedge$ 0xff oraz (wartość celu $\gg 8$) $\wedge$ 0xff. Dane użytkownika są formatowane do wysłania jako tablica bajtów zawierająca: 0x4f, 0x00, 0x00, rokUrodzenia $\wedge$ 0xff, (rokUrodzenia $\gg 8$)$\wedge$ 0xff, miesiąc, dzien, płeć (0x00 lub 0x01), wzrost $\wedge$ 0xff, (wzrost $\gg 8$) $\wedge$ 0xff, waga $\times$ 200 $\wedge$ 0xff, (waga $\times$ 200 $\gg 8$) $\wedge$ 0xff, id $\wedge$ 0xff, (id $\gg 8$) $\wedge$ 0xff, (id $\gg 16$) $\wedge$ 0xff oraz (id $\gg 24$) $\wedge$ 0xff.

Algorithm 3.3: Konfiguracja MiBanda 3 - Część 1

begin

$settingsChar \leftarrow 00000003 - 0000 - 3512 - 2118 - 0009af100700$

$batteryChar \leftarrow 00000006 - 0000 - 3512 - 2118 - 0009af100700$

$eventsChar \leftarrow 00000010 - 0000 - 3512 - 2118 - 0009af100700$

$userChar \leftarrow 00000008 - 0000 - 3512 - 2118 - 0009af100700$

$hrControlChar \leftarrow 00002a39 - 0000 - 1000 - 8000 - 00805f9b34fb$

$hrChar \leftarrow 00002a37 - 0000 - 1000 - 8000 - 00805f9b34fb$

$stepsChar \leftarrow 00000007 - 0000 - 3512 - 2118 - 0009af100700$

$serialNumChar \leftarrow 00002a25 - 0000 - 1000 - 8000 - 00805f9b34fb$

$hardwareChar \leftarrow 00002a27 - 0000 - 1000 - 8000 - 00805f9b34fb$

$softwareChar \leftarrow 00002a28 - 0000 - 1000 - 8000 - 00805f9b34fb$

Włącz powiadomienia na $settingsChar, batteryChar$ oraz $eventsChar$

Odczytaj wartość z $serialNumChar, hardwareChar, softwareChar$ oraz $batteryChar$

Ustaw język angielski wysyłając $ByteArray(0x06, 0x17, 0x00, 0x65, 0x6e, 0x5f, 0x55, 0x53)$ do $settingsChar$

Wyłącz odblokowywanie ekranu w opasce wysyłając $ByteArray(0x06, 0x16, 0x00, 0x00)$ do $settingsChar$

Wyłącz tryb nocny wysyłając $ByteArray(0x1a, 0x00)$ do $settingsChar$

Ustaw format daty wysyłając $ByteArray(0x06, 0x1e, 0x00) + format$ do $settingsChar$, gdzie $format$ to opisany powyżej przetworzony łańcuch znaków

Ustaw format wyświetlanej daty wysyłając $ByteArray(0x06, 0x0a, 0x00, 0x03)$ do $settingsChar$

Ustaw 24-godzinny zegar wysyłając $ByteArray(0x06, 0x02, 0x00, 0x01)$ do $settingsChar$

Ustaw przykładowe dane o użytkowniku wysyłając $userInfo$ do $settingsChar$, gdzie $userInfo$ to przetworzone dane o użytkowniku opisane powyżej

Ustaw jednostki metryczne wysyłając $ByteArray(0x06, 0x03, 0x00, 0x00)$ do $settingsChar$

Włącz powiadomienia na $userChar$

Ustaw lokalizację noszenia opaski na lewą rękę wysyłając $ByteArray(0x20, 0x00, 0x00, 0x02)$ do $userChar$

Ustaw cel fitness wysyłając $ByteArray(0x10, 0x00, 0x00, x[0], x[1], 0x00, 0x00)$ do $userChar$, gdzie x to przetworzona wartość celu

Ustaw elementy menu opaski wysyłając $ByteArray(0x0a, 0x7f, 0x30, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08)$ do $settingsChar$

Wyłącz w opasce tryb “Nie przeszkadzać” wysyłając $ByteArray(0x09, 0x82)$ do $settingsChar$

Wyłącz gest obrotu nadgarstka do zmiany pozycji w menu wysyłając $ByteArray(0x06, 0x0d, 0x00, 0x00)$ do $settingsChar$

Wyłącz gest podniesienia nadgarstka do włączenia ekranu opaski wysyłając $ByteArray(0x06, 0x05, 0x00, 0x00)$ do $settingsChar$

Włącz wyświetlanie ID dzwoniącego w opasce wysyłając $ByteArray(0x06, 0x10, 0x00, 0x00, 0x01)$ do $settingsChar$

Wyłącz powiadomienia o celu fitness wysyłając $ByteArray(0x06, 0x06, 0x00, 0x00)$ do $settingsChar$

Wyłącz powiadomienia o nieaktywności wysyłając $ByteArray(0x08, 0x00, 0x3c, 0x00, 0x04, 0x00, 0x15, 0x00, 0x00, 0x00, 0x00, 0x00)$ do $settingsChar$

Włącz wspomaganie detekcji snu przez pomiar pulsu wysyłając $ByteArray(0x15, 0x00, 0x01)$ do $hrControlChar$

end

Algorithm 3.4: Konfiguracja MiBanda 3 - Część 2

begin
  Włącz powiadomienie o utracie połączenia wysyłając $ByteArray(0x06, 0x0c, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00)$ do $settingsChar$
  Włącz powiadomienie o nawiązaniu połączenia wysyłając $ByteArray(0x06, 0x01, 0x00, 0x01)$ do $settingsChar$
  Ustaw interwał automatycznego pomiaru pulsu na minimalny wysyłając $ByteArray(0x14, 0x01)$ do $hrControlChar$
  Poproś o alarmy wysyłając $0x0d$ do $settingsChar$
  Włącz powiadomienia na $hrChar$ oraz $stepsChar$
end

Implementacja aplikacji

4.1 Opis technologii

Do implementacji systemu został użyty język Kotlin w wersji 1.3.61. (aktualizacja wersji i lepszy opis jezyka) Interfejs graficzny zaprojektowano w oparciu o komponenty pochodzące z biblioteki AndroidX oraz Material. Do nawigacji w głównej aktywności aplikacji wykorzystano NavigationUI. Do implementacji bazy danych użyto biblioteki Room(moze coś wiecej) zapewniającej poziom abstrakcji nad SQLite. Wykorzystano bibliotekę Hilt/Dagger w celu wstrzykiwania dependencji w obrębie aplikacji w celu zredukowania tak zwanego "boilerplate code" (może więcej).

W systemie wykorzystano inteligentną opaskę Xiaomi MiBand 3. Komunikacja z nią została zaimplementowana na podstawie nieoficjalnego SDK(większośc trzeba było napisać na nowo lub wyrzucić) opartego o bibliotekę RxAndroid pozwalającą tworzyć asynchroniczne programy bazujące na wydarzeniach korzystając z obserwowalnych sekwencji.

4.2 Omówienie wybranych kodów źródłowych

Instrukcja obsługi

W tym rozdziale omówniono założenia co do środowiska, w którym realizowana będzie instalowana. Przedstawiono procedurę instalacji i wdrożenia systemu.

5.1 Instalacja i konfiguracja

W tej sekcji omówię wymagania środowiskowe aplikacji oraz kwestie konfiguracyjne jak parowanie opaski czy ustawienie hasła.

5.1.1 Wymagania sprzętowe

System android 9; moduł BT; opaska MiBand 3;

5.1.2 Instalacja

5.1.3 Pierwsze uruchomienie

- dodać screeny do punktów

1. Gdy aplikacja jest uruchamiana po raz pierwszy należy ustawić jej w ustawieniach "Usage stats permission". Jeśli przy uruchomieniu nie ma tego pozwolenia aplikacja przekieruje w odpowiednie miejsce w ustawieniach jednak będzie to wymagało ponownego uruchomienia aplikacji.

2. Następnie jeśli aplikacja jest uruchamiana z aktywnym "Usage stats permission"to użytkownik jest proszony o pozwolenie na lokalizację. Jest ono niezbędne do poprawnego działania BT.

3. Kiedy uzyskane zostaną wszystkie potrzebne pozwolenia użytkownik zostaje przeniesiony do aktywności, w której jest tworzone hasło, które będzie wykorzystywane przy odblokowywaniu trybu Łockdown".

4. Użytkownik wprowadza dwa razy hasło i dotyka przycisk (drzwi ze strzałką)

5. Po utworzeniu hasła użytkownik zostaje przekierowany do aktywności odpowiadającej za parowanie opaski MiBand 3.

6. Użytkownik klika przycisk scan for devices

7. Następnie uruchamiany jest skan urządzeń ble z filtrem na mi band 3

8. Jeśli urządzenie zostanie odnalezione pojawi się na ekranie. Jeżeli nie zostanie odnalezione należy powtórzyć skan dotykając przycisk scan for devices

9. Po naciśnięciu na odpowiednie urządzenie na liście znalezionych urządzeń zostaje uruchomiona usługa odpowiadająca za komunikację z MiBand i użytkownik przechodzi do głównego widoku aplikacji

5.1.4 Wybór aplikacji do zablokowania

5.2 Przykłady użycia

W tej sekcji przedstawię jak działa aplikacja od strony użytkownika poprzez opis czynności potrzebnych do wykonania określonych zadań, np. sprawdzenie stanu opaski, zmiana blokowanych aplikacji czy sprawdzenie statystyk.

Podsumowanie

W podsumowanie należy określić stan zakończonych prac projektowych i implementacyjnych. Zaznaczyć, które z zakładanych funkcjonalności systemu udało się zrealizować. Omówić aspekty pielęgnacji systemu w środowisku wdrożeniowym. Wskazać dalsze możliwe kierunki rozwoju systemu, np. dodawanie nowych komponentów realizujących nowe funkcje.

W podsumowaniu należy podkreślić nowatorskie rozwiązania zastosowane w projekcie i implementacji (niebanalne algorytmy, nowe technologie, itp.).

Propozycje rozwoju w przyszłości:

- Wyłączenie łączności wifi i danych mobilnych podczas blokowania (niemożliwe jeśli nie masz roota bądź systemowej aplikacji albo korzystasz z android 10+)
- Wsparcie dla większej ilości urządzeń
- Dokładniejsze statystyki aktywności
- Maskowanie aplikacji - zmodyfikowanie wyglądu i komunikatów by przypominała zwykłą aplikację towarzyszącą opasce

Bibliografia

[1] F. Almenárez-Mendoza, L. Alonso, A. Marín-López, P. Cabarcos. Assessment of fitness tracker security: A case of study. *Proceedings*, 2:1235, 10 2018.

[2] Bluetooth SIG. *Bluetooth Core Specification Version 5.2*, 12 2019. Vol 3, Części F i G.

[3] Y. Cifuentes, L. Beltrán, L. Ramírez. Analysis of security vulnerabilities for mobile health applications. *International Journal of Health and Medical Engineering*, 9(9):1067 – 1072, 2015.

[4] J. Clement. Share of global mobile website traffic 2015-2021. https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/.

[5] G. Combs. Wireshark. Uzyskano ze strony produktu https://www.wireshark.org/.

[6] E. De Cristofaro, H. Du, J. Freudiger, G. Norcie. Two-factor or not two-factor? a comparative usability study of two-factor authentication. *USEC*, 09 2013.

[7] Freedom of the Press Foundation, Guardian Project. Haven: Keep watch. Strona internetowa projektu https://guardianproject.github.io/haven/.

[8] Google. Android Management API. Uzyskano z poradnika w dokumentacji Android Management API https://developers.google.com/android/management/introduction.

[9] Google. Android Management API Policies. Uzyskano z dokumentacji Android Management API https://developers.google.com/android/management/reference/rest/v1/enterprises.policies.

[10] Google. Background Service Limitations. Uzyskano z opisu wersji Androida Oreo https://developer.android.com/about/versions/oreo/background.html#services.

[11] Google. Optimize for Doze and App Standby. Uzyskano z poradnika w dokumentacji Androida https://developer.android.com/training/monitoring-device-state/doze-standby.

[12] Google. Wake Lock. Uzyskano z dokumentacji Androida https://developer.android.com/reference/kotlin/android/os/PowerManager.WakeLock.

[13] S. O'Dea. Number of smartphone users worldwide from 2016 to 2026. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[14] J. Rutkowska. Evil Maid goes after TrueCrypt! Uzyskano z bloga autorki https://blog.invisiblethings.org/2009/10/15/evil-maid-goes-after-truecrypt.html.

[15] H. Shen, C. Faklaris, H. Jin, L. Dabbish, J. I. Hong. 'I Can't Even Buy Apples If I Don't Use Mobile Pay?': When Mobile Payments Become Infrastructural in China. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW2), Paz. 2020.

[16] A. Shimokawa, C. Pfeiffer, D. Gobbetti. Gadgetbridge. Uzyskano z repozytorium projektu https://codeberg.org/Freeyourgadget/Gadgetbridge.

[17] Yubico Inc. YubiKey 5 NFC. Uzyskano ze strony producenta 08.06.2021 https://www.yubico.com/pl/product/yubikey-5-nfc/.

[18] Yubico Inc. Yubikey for mobile. Uzyskano ze strony producenta https://resources.yubico.com/53ZDUYE6/as/q4bsus-2mej80-29grce/YubiKey˙for˙Mobile˙Solution˙Brief.pdf.

Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.