RONALDO CESAR MENGATO JUNIOR

MODELO DE AUTENTICAÇÃO MULTICANAL BASEADO EM PROXIMIDADE

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial à obtenção do título de mestre em Informática.

Orientador: Prof. Dr. Altair Olivo Santin

CURITIBA

2018

RESUMO

Atualmente, os sistemas de infraestrutura crítica (IC) podem ser facilmente acessados remotamente. No entanto, esta exposição tem vários riscos que podem causar danos importantes. A localização do usuário é um atributo de autenticação importante para a segurança em muitos contextos, porque pode bloquear um intruso da Internet de acessar um IC. No entanto, técnicas de autenticação baseadas em localização podem ser forjadas (imitadas). Este trabalho propõe um método baseado em proximidade, exigindo que um usuário esteja fisicamente próximo a um dispositivo de referência (âncora) com um fator para estar habilitado à autenticação. O método baseado em proximidade considera um leitor multicanal (óptico e wireless) para obter o desafio de autenticação. Então, consideramos um método baseado em proximidade como um fator e um leitor de canal como outro fator de autenticação. O protótipo implementa como âncora um Beacon para o método wireless e um display LCD para leitura ótica. Adicionalmente, é utilizado um smartwatch (com ancora móvel) para exibir um QR Code para leitura óptica. Foi feita uma avaliação de desempenho para avaliar a escalabilidade da proposta e criado um modelo de adversário para avaliar os riscos da proposta; o protótipo se mostrou promissor para autenticação baseada em proximidade e pode ser usado em ambientes com grande número de usuários.

Palavras-chave: Autenticação Multifator; Autenticação baseada em proximidade; IdM.

ABSTRACT

Currently the critical infrastructure (CI) systems can be easily accessed remotely. However, this expose has several risks which can cause important risks for the CI. The user's location is an authentication attribute is important to security in many contexts, because it can block an intruder from Internet to access a CI. However, location-based authentication techniques can be forged (counterfeit). This work proposes a proximity-based method, requiring a user be physically close to a reference (anchor) device to be enable for authentication. The proximity-based method considers a multichannel reader (optical and wireless) to get the authentication challenge. Thus, we consider a proximity-based method as a factor and a channel reader as another authentication factor. The prototype implements as anchor a beacon for the wireless and a LCD-display for optical reading. Additionally, as mobile anchor we apply a smartwatch to display the QRCode for optical reading. We create an adversary model to assess the risks of the proposal. A performance evaluation with the prototype showed that the model is promising for proximity-based authentication and can be used in environments with large numbers of users.

Keywords: Multifactor Authentication; Proximity-based Authentication; IdM.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

App	Aplicativo
CP	Credencial de Proximidade
CWE	Common Weakness Enumeration
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
EID	Ephemeral Identifier
GPS	Global Positioning System
IC	Infraestrutura Crítica
IdM	Identity Management
IdP	Identity Provider
IP	Internet Protocol
LBS	Location Based Service
LCA	Location Certification Authority
MAC	Media Access Control
QRCode	Quick Response Code
OTP	One-Time Password
OWASP	Open Web Application Security Project
SP	Service Provider
SSID	Service Set Identifier
SSO	Single Sign-On
TOTP	Time-based One-Time Password

Capítulo 1

Introdução

Neste capítulo são discutidas a contextualização, motivação, hipótese, objetivos e as contribuições. Em seguida, a organização do restante do trabalho também é apresentada.

1.1. Contextualização

Atualmente está sendo disponibilizado, virtualmente, uma ampla diversidade de serviços, que oferecem diferentes nuances de acesso à diferentes tipos de dispositivos. Dispositivos estes que controlam informações críticas acessíveis na internet e podem expor vulnerabilidades causando prejuízos incalculáveis [1], isso porque o hacker pode controlar o sistema remotamente, como se fosse um usuário legítimo, explorando brechas de segurança.

Para mitigar esse problema, é necessário implementar mecanismos de autenticação robustos para que, mesmo o hacker explorando uma vulnerabilidade, não possa comandar o sistema a partir da internet. A autenticação multifator combina diferentes fatores para prover robustez na autenticação, pois caso um fator seja comprometido isoladamente, o invasor não consegue se autenticar no sistema por depender de outro(s) fatore(s).

Na literatura, diversos autores propõem novas técnicas/fatores de autenticação [2, 3 e 4], sendo que a autenticação baseada em georreferenciamento tem sido frequentemente utilizada. Essa técnica tem o objetivo de somente autenticar um usuário se o mesmo estiver em um determinado local físico. Por exemplo, um médico que realiza cirurgia por telepresença deve estar dentro de um hospital para acessar o sistema remoto; o mesmo vale para um operador de uma usina de energia elétrica, que só pode operar o sistema se estiver na central de comando do mesmo.

Entretanto, as técnicas baseadas em georreferenciamento encontradas na literatura, e em soluções de mercado, ou são facilmente forjáveis (imitadas) [3 e 4] por utilizarem de mecanismos simples como sinais de rádio frequência ou GPS, ou então possuem demandas

específicas que inviabilizam a solução proposta [5 e 6]. Ou ainda, opções que, apesar de serem robustas e não forjáveis, não apresentam suporte a vários fatores de autenticação [7].

Uma alternativa a esse método é adotar um requisito de autenticação por estar próximo a um dispositivo de referência (dispositivo âncora), que é posicionado estrategicamente em algum local físico, ou portado por alguém escolhido pelo mesmo critério estratégico.

Como a proximidade é um fator físico que exige a presença do usuário próximo ao dispositivo âncora, é possível protegê-lo posicionando dentro da instalação (i.e., edifícios, salas, etc.), que inclusive podem estar protegidos por um serviço de vigilância patrimonial, como o aplicado na proteção de valor dos bancos, por exemplo. O dispositivo âncora também pode ser carregado ou vestido por alguém. Nesse caso, devido à mobilidade, a pessoa que carrega o dispositivo pode possuir um serviço de proteção pessoal para protegê-lo fisicamente.

É importante utilizar diferentes dispositivos (canais) nos fatores referentes a autenticação por proximidade, para evitar que uma vulnerabilidade num único dispositivo (canal) deixe vulnerável todo o esquema. O ideal é que cada dispositivo seja implantado em uma plataforma diferente, pois nesse caso o hacker terá o desafio de controlar, dois ou mais dispositivos (múltiplos canais), para conseguir completar a autenticação por proximidade. Na prática, o efeito desta abordagem impõe o desafio da presença física e do controle de dois ou mais dispositivos diferentes para conseguir comprometer a abordagem.

1.2. Motivação e Hipótese

A autenticação multifator é um procedimento de grande importância na garantia da segurança de sistemas computacionais, isso se justifica pelo grande número de trabalhos propostos, e as diferentes abordagens utilizadas no processo de autenticação [8]. Uma vertente deste tipo de pesquisa busca avançar na autenticação contínua, a fim de garantir que o usuário permanece no estado de autenticado enquanto utiliza o recurso. Tal situação se faz útil quando se verifica a localização do usuário em tempo real, como é o caso desta proposta.

Outra abordagem que aumenta o nível de segurança do sistema é a escolha de diferentes canais para realizar a autenticação. Neste contexto, um canal é um meio de comunicação onde dados sensíveis são trafegados, assim, para que um atacante tenha acesso ao sistema diferentes canais devem ser comprometidos. Este tipo de abordagem é indicado contra-ataques *man-in-the-middle*, *phishing* e *malwares* [9].

Este trabalho faz uso das técnicas multifator e multi-canal, propondo três métodos de autenticação por proximidade que usam localidade como referência, podendo ser utilizados isoladamente ou combinados. São eles: baseado em rede sem fio, baseado em leitura óptica e um método que usa mobilidade, o baseado em *Wearable*.

O método baseado em rede sem fio transmite o desafio (segredo) de autenticação baseado em proximidade para um Beacon (dispositivo âncora), que através de uma rede sem fio de curto alcance, Bluetooth, revela no display (smartphone) do usuário a resposta ao desafio de autenticação. O desafio de autenticação não é facilmente forjável, porque usa criptografia de *Elliptic Curve* (EC) para proteção da mensagem e, por assegurar que quem gerou o desafio, é o dispositivo âncora que está sincronizado (pareado) com o um sistema de gestão de identidades (IdM, Identity Management).

O método óptico exibe um desafio de autenticação através de um display do dispositivo âncora no formato de um QRCode. A resposta ao desafio de autenticação por proximidade não é facilmente forjável, porque há sincronização (pareamento) com o IdM e o usuário tem um tempo curto para ler opticamente o QRCode e transmiti-lo ao IdM usando o Time-based One Time Password (TOTP). Neste caso, a resposta ao desafio é transmitida ao IdM como One Time Password (OTP).

O método baseado em *Wearable* usa um smartwatch para exibir um desafio de autenticação que deve ser lido por smartphone. Assim, como no Optical-based a resposta ao desafio de autenticação por proximidade não é facilmente forjável, porque há sincronização (pareamento) com o IdM e usuário tem um tempo para ler opticamente o QRCode e transmiti-lo ao IdM usando o TOTP. A vantagem deste método é que usuários habilitados podem carregar consigo o dispositivo âncora e fazer operações críticas, mesmo não estando presentes fisicamente numa localidade. O grau de segurança da abordagem continua robusto porque, para que o fator de proximidade seja possível, é necessário possuir o dispositivo âncora móvel e estar próximo do mesmo.

Com isso, este trabalho considera a hipótese de que pela combinação dos diferentes canais apresentados, os fatores de identidade e de proximidade e um sistema IdM, seja possível oferecer um método de autenticação robusto, com credenciais não forjáveis e eficiente na defesa de ataques remotos. O resultado final proporciona ao usuário um modo de acesso diferente, visto que ele não precisa carregar nenhum dispositivo extra (*token* físico) para o segundo fator da autenticação.

1.3. Objetivos

O objetivo geral do trabalho é o desenvolvimento de um modelo de autenticação multicanal e multifator baseado na proximidade do usuário aos dispositivos âncoras, garantindo (ou dificultando ao máximo) o não forjamento das credenciais. A proposta pretende restringir a autenticação apenas a usuários que estão em locais físicos permitidos, sendo que a definição destes locais e permissões ficam a critério do administrador. Além da robustez e da segurança, a usabilidade e a escalabilidade também são requisitos do modelo.

Objetivos específicos:

- Projetar e desenvolver o modelo e o funcionamento dos dispositivos âncoras, considerando suas limitações e facilidades;
- Desenvolver um IdM customizado para uma autenticação multifator com base na credencial de proximidade (CP);
- Implementar um modelo de pareamento entre os dispositivos âncoras e o servidor com o IdM;
- Definir e implantar dispositivos âncoras com diferentes canais (meios) de comunicação;
- Implementar um protótipo com base nas especificações do modelo proposto;
- Avaliar a performance e segurança do protótipo.

1.4. Contribuições

O trabalho proposto tem as seguintes contribuições: (I) construção de um modelo de autenticação robusto aplicável a praticamente qualquer sistema de gerenciamento de identidade; (II) a utilização de um método de sincronização seguro entre cliente e servidor sem a troca de chaves sensíveis pela rede; (III) desenvolvimento de um módulo de autenticação de proximidade para servidores IdM construídos em Java e (IV) criação de um simulador Beacon para o protocolo Eddystone-EID para o sistema operacional Android.

1.5. Organização

Para fins estruturais, o trabalho está organizado da seguinte forma: o capítulo 2 aborda a fundamentação teórica, o capítulo 3 aponta os trabalhos relacionados existentes, enquanto que no 4 está descrito o modelo de autenticação proposto. Em seguida, o capítulo 5 apresenta os detalhes da construção de um protótipo e sua avaliação. Por fim, no capítulo 6 estão as conclusões do estudo.

Capítulo 2

Fundamentação

Este capítulo apresenta assuntos correlatos com a proposta do trabalho, tais como: autenticação, gestão da identidade, dispositivos âncoras, algumas considerações pertinentes sobre criptografia de chaves de curva elíptica e os conceitos sobre o algoritmo TOTP.

2.1. Autenticação

A autenticação está presente no mundo há muitos anos e em diferentes formas. Durante séculos, as pessoas utilizam meios (como guardas, paredes e etc.) para bloquear o acesso de pessoas não autorizadas a algum tipo de recurso.

Em um sistema computacional isso não é diferente, sendo a autenticação o ato de determinar a identidade de algum elemento, como: pessoa, grupo de pessoas, computador ou servidor [10]. O processo de autenticação é de suma importância para a segurança dos sistemas, uma vez que ela controla as requisições dos usuários definindo se eles podem, ou não, acessar tal função.

Os sistemas de autenticação, em sua grande maioria, possuem alguns elementos em comum, como cita [11]:

- Pessoa/Entidade: pessoa ou grupo de pessoas a ser autenticado;
- Característica distintiva: atributo que diferencia a pessoa ou grupo dos demais;
- Proprietário: é o responsável pela aplicação sendo utilizada, onde o sistema de autenticação está presente;
- Mecanismo de autenticação: é o elemento que verifica a presença da característica distintiva;
- Mecanismo de controle de acesso: concede ou não os privilégios dependendo do resultado da autenticação.

Com o objetivo de trazermos esses elementos para um exemplo mais concreto, podemos imaginar no cenário bancário, onde há um sistema rodando em algum caixa eletrônico. Nesse caso, a pessoa/entidade é alguém que possui uma conta no banco (cliente), a característica distintiva pode ser o cartão ou senha, o proprietário é o banco, mecanismo de autenticação é o software de validação do cartão (ou senha) e o mecanismo de controle de acesso é a permissão, ou não, da transação bancária.

Portanto, os sistemas de autenticação utilizam de fatores de autenticação (ou característica distintiva) para provar a identidade do usuário. Em geral, eles podem ser [12]:

1. Alguma coisa que o usuário sabe: senha;
2. Alguma coisa que o usuário possui: cartão inteligente, *token*;
3. Algo que o cliente é: biometria (impressão digital, voz).

Os primeiros sistemas utilizavam apenas o fator que o usuário conhece, pois, a utilização de senhas era mais fácil e mais barata de implementar. No entanto, diversos problemas de segurança ocorriam: a) os usuários esquecem facilmente; b) são enviadas por e-mail, passíveis de interceptação; e c) podem ser adivinhadas com pouco esforço.

A autenticação com cartão inteligente é considerada mais segura, pois além da senha que o usuário conhece, ele deve possuir o cartão; porém, não anula a ação de um atacante, caso o cartão seja roubado. Por fim, o fator biometria é considerado o mais seguro, pois é parte do usuário e único a cada um. No entanto, este também possui desvantagens, isso porque a coleta dos dados biométricos está sujeita a problemas naturais (impressões digitais podem falhar), o que dificulta a comparação exata [12]. Por isso, um sistema de autenticação multifator, que combine um ou mais desses elementos é desejável, pois garante mais segurança na correta identificação dos usuários.

2.1.1. Autenticação Multifator

Autenticação multifator ocorre quando um sistema utiliza múltiplos métodos, ou fatores, para autenticar um usuário, ou seja, o usuário precisa informar a sua senha, apresentar o cartão de identificação e sua impressão digital.

Apesar desses três fatores serem os mais comuns, a autenticação multifator não se limita apenas a eles, uma vez que os aplicativos atuais estão adotando soluções cada vez mais personalizadas. Um exemplo é o Instagram, aplicativo de compartilhamento de fotos que possui cerca de 700 milhões de usuários, que adotou um método de autenticação de dois fatores. A

solução deles funciona da seguinte forma: o usuário cadastra o número de seu celular junto com as informações de perfil (e-mail, senha e etc.); assim, na hora de fazer o login o aplicativo pede primeiro seu e-mail e senha, em seguida um código é enviado por mensagem de texto ao celular do usuário, que digita esse código no Instagram para se autenticar [13]. Outros aplicativos famosos, como Facebook e Google, usam abordagens parecidas.

Esse tipo de autenticação é considerado forte e robusto, pois, mesmo que um invasor tenha posse do e-mail e da senha do usuário, ele precisa ter o seu celular (seguindo o exemplo do Instagram). Porém, por adicionar mais etapas no processo, a usabilidade é prejudicada, e consequentemente alguns usuários optam por não utilizar a autenticação multifator.

Neste trabalho, o método proposto utiliza os conceitos citados sobre o uso de diversos fatores no processo de autenticação, pois, os cenários onde serão utilizados a proposta são críticos, e ainda assim, alguns métodos não comprometem a usabilidade.

2.2. Gestão da Identidade (IdM, *Identity Management*)

Uma identidade é alguma informação utilizada para representar uma entidade dentro do contexto dos sistemas de tecnologia da informação e comunicação (TIC) [14]. Ela pode ser descrita como a representação, prova ou a credencial de um usuário ou entidade de algum tipo de serviço ou aplicação, por isso, ela é utilizada para distinguir os usuários e oferecer diferentes privilégios [15]. Portanto, é uma prática comum no mundo digital utilizar técnicas de identidade do usuário, como por exemplo perfis de atributos, biometria, *tokens* e identificadores únicos (e-mails, documentos e etc.).

Pensando no contexto de uma aplicação com milhares de usuários, o que é comum hoje em dia, a dificuldade de manter a identidade de cada um deles é imensa; assim como, com a diversa gama de serviços disponíveis online, os usuários também precisam se identificar em diversos aplicativos de diferentes domínios, o que pode incorrer na adoção de senhas fracas. É nesse contexto que sistemas de gerenciamento de identidade (IdM) podem auxiliar. Esse tipo de sistema possui políticas, regras e métodos que implementam a autenticação da identidade, o gerenciamento da autorização e o controle de acesso a recursos dos provedores de serviços [15].

Um sistema IdM possui três elementos principais:

- **Provedor de serviço (*Service Provider* – SP)**: aplicações que oferecem algum tipo de serviço para determinada entidade, como por exemplo um e-commerce ou internet banking.

- **Provedor de identidade (*Identity Provider* – IdP)**: é o coração do IdM, ele fornece serviços aos usuários como registro, verificação e armazenamento da identidade.
- **Usuário**: é o cliente tanto do SP quanto do IdP, este elemento possui uma identidade que é requerida ao usar algum serviço.
- **Identidade**: é o conjunto de atributos pertencentes a um usuário. Exemplo: nome, CPF, apelido.

Os sistemas de gerenciamento de identidade podem ser divididos nos seguintes modelos: tradicional, centralizado, centrado no usuário e federado. Cada um deles será explorado em detalhes nas próximas seções.

2.2.1. Modelo Tradicional

É o modelo mais fácil de ser implementado e o mais antigo; nele, o servidor provedor do serviço também é o provedor da identificação do usuário. As operações de registro, remoção e modificação da identidade, assim como os processos de autenticação e de autorização, são todos realizados pelo mesmo dispositivo responsável por hospedar a aplicação. A Figura 1 mostra a arquitetura do modelo tradicional, onde existem dois provedores de serviços diferentes e cada um fornece os atributos, credenciais e identificadores de cada usuário.

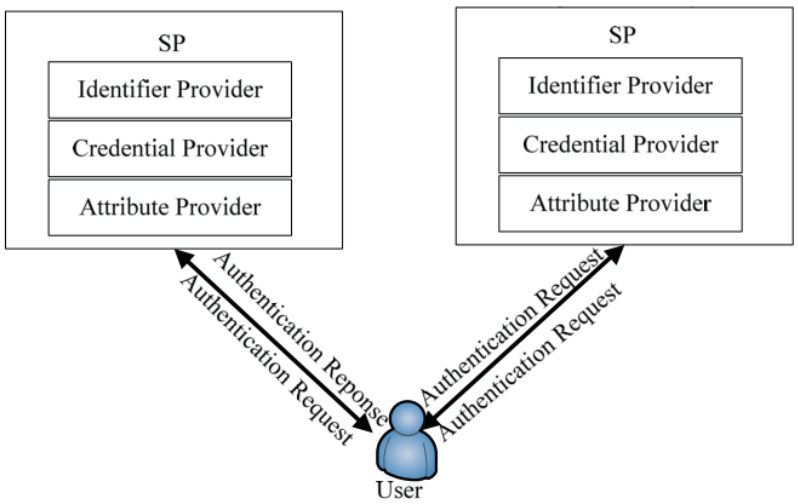

Figura 1 - Modelo Tradicional de IdM - Adaptado de [15]

O principal problema do modelo isolado é a dificuldade em gerenciar um grande número de usuários, pois, quanto maior a base de atributos de identidade maior é a complexidade das

operações. Portanto, em sistemas online, onde a escalabilidade é um requisito primordial, esse modelo não é recomendável.

2.2.2. Modelo Centralizado

Este modelo é implementado em um esquema cliente/servidor; ele separa as funções de serviços das funções de gerenciamento de identidade, assim operações como: armazenamento de identidade e autenticação são executadas em um servidor central (IdP). Com isso, os usuários se beneficiam de uma maior flexibilidade, pois, esse modelo habilita o conceito de autenticação única SSO (*Single Sign-On*), onde o processo de autenticação ocorre apenas uma vez e os usuários podem utilizá-la em outros provedores de serviços até que suas credenciais se expirem. Na Figura 2 está representado a troca de mensagens entre os elementos do modelo centralizado.

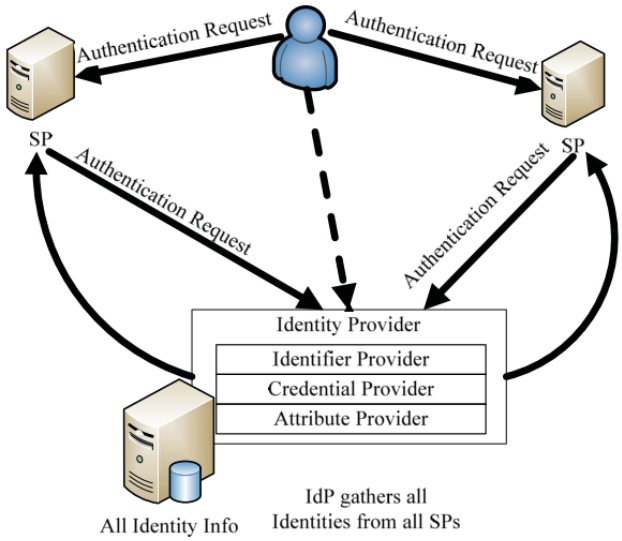

Figura 2 - Modelo Centralizado de IdM - Adaptado de [15]

Apesar de prover maior flexibilidade e maior eficiência na gestão de grande quantidade de usuários do que o modelo tradicional, o modelo centralizado possui algumas desvantagens: a) todos os dados de identidades são armazenados em um único provedor de identidade; b) ponto de falha, pois, se o IdP não estiver disponível o SP não pode ser acessado; e c) todos os usuários devem ser do mesmo domínio.

2.2.3. Modelo Federado

Superando a limitação do modelo centralizado, este tem a capacidade de autenticar usuários em diferentes serviços dentro do mesmo domínio administrativo, como empresas, governos e universidades [16]. O conceito do modelo é que as políticas e os padrões de uso sejam estabelecidos entre os provedores de serviços, assim, eles definem uma identidade, emitida por um IdM de um determinado domínio, que seja aceita entre os SP de outros domínios.

Assim, o servidor de identidade pode compartilhar uma credencial entre os provedores de serviços que compõem o elo de confiança, sendo necessário de políticas e protocolos bem definidos para que isso ocorra. Na Figura 3 está representada a arquitetura deste modelo. Alguns sistemas que implementam o modelo são: Liberty Alliance, Shibboleth e OASIS.

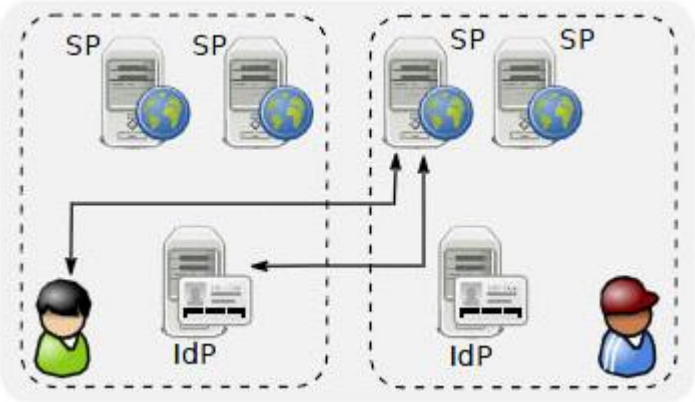

Figura 3 - Modelo Federado - Adaptado de [16]

Alguns desafios podem ser encontrados nesse modelo, como por exemplo: em mal uso de dados (um serviço pode acessar mais atributos do que deveria), manter políticas de privacidade (o usuário deve permitir todos os serviços do domínio) entre os SPs e a semântica dos atributos [10].

2.2.4. Modelo Centrado no Usuário

Diferente dos modelos citados, nesse o usuário detém o controle dos dados de identificação, tendo a capacidade de criar, atualizar, apagar e utilizar da maneira que o convém. Entretanto, o modelo centrado no usuário é comumente implementado com base em algum outro, geralmente o modelo federado.

A principal característica do modelo é o consentimento do usuário, ou seja, é ele quem decide o que, e com quem compartilhar uma informação. Com isso, há uma inversão de paradigma, sendo que os SPs não mais possuem o controle dos dados [15].

O sistema que implementa esse modelo fornece ao usuário algumas vantagens, como: maior usabilidade, flexibilidade no controle dos dados e autenticação mútua [17]. Os sistemas OpenID, Higgins e Microsoft CardSpace são exemplos desse tipo de modelo. A Figura 4 mostra um exemplo de arquitetura.

A implementação mais atual deste modelo é o protocolo *User-Managed Access* (UMA), que é baseado no OAuth2.0. UMA foi idealizado para padronizar o modo que um usuário autorize que seus dados seja utilizado por uma parte interessada. O exemplo mais prático é o de um usuário web em uma plataforma autorizando que um aplicativo mobile nativo (por exemplo, um jogo) utilize a sua credencial [18]. A Figura 4 ilustra o funcionamento do protocolo, como podemos ver um usuário pode gerenciar, autorizar e controlar os seus recursos.

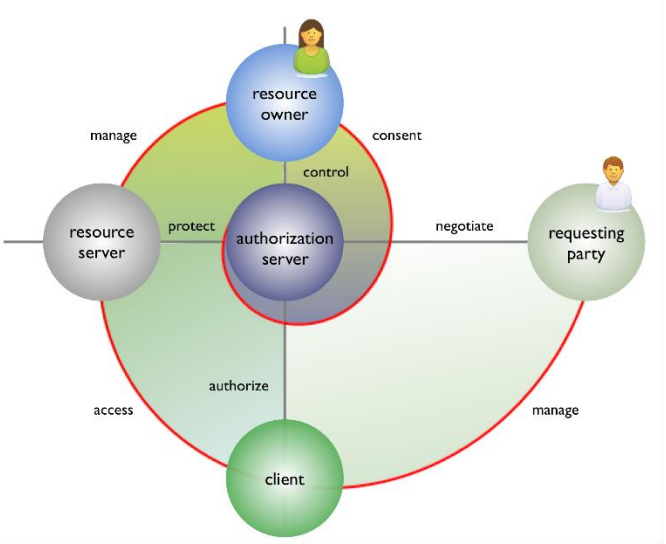

Figura 4 – Arquitetura UMA - Adaptado de [18]

2.2.5. OpenID Connect

OpenID Connect é uma plataforma inspirada no modelo de gerenciamento de identidade centrado no usuário. Criada em 2005 pela comunidade de código aberto, atualmente já alcança a marca de 1 bilhão de contas de usuários [19].

A utilização deste tipo de gerenciamento traz diversos benefícios, tanto ao usuário quanto para o provedor do serviço. No cenário atual da internet existem milhões de aplicativos que pedem a autenticação dos usuários, na maioria das vezes usa atributos do tipo login e senha,

isso faz com que senha fracas e fáceis de serem lembradas sejam adotadas pelos usuários. Por outro lado, os donos de aplicativos precisam gerenciar milhares de perfis, aumentando a complexidade de manter a segurança e a performance dos sistemas. Portanto, uma plataforma como o OpenID auxilia ambos os casos.

A plataforma do OpenID Connect [19] opera em uma camada acima do protocolo OAuth. Ele utiliza a sintaxe e a estrutura do OAuth para autenticar usuários e obter seus perfis. Também estão presentes funções para controlar a privacidade dos dados de autenticação que ficam disponíveis aos usuários, deixando-os com mais liberdade para decidir quais aplicativos podem obter o perfil.

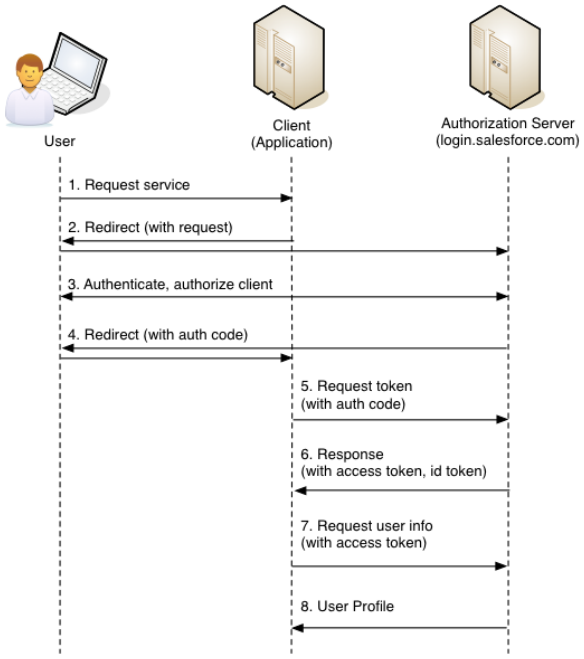

Figura 5 - Fluxo do OpenID - Adaptado de [20]

Na Figura 5 é apresentado o fluxo de trabalho do OpenID. O processo inicia com o cliente fazendo uma requisição no provedor de serviço (1); em seguida, o SP redireciona o cliente para o servidor de autorização para efetuar o login (2); o IdP então autentica e autoriza o cliente caso as credenciais sejam válidas (3); o cliente é redirecionado de volta para a aplicação que ele deseja acessar e um código de autorização é passado pelo IdP (4); a partir desse código o SP faz uma requisição no IdP em busca do token do cliente (5); o servidor IdP responde com os tokens de acesso e o de ID, que funcionam para definir quais recursos o aplicativo pode acessar no IdP, e garantir acesso do cliente na aplicação (6); por fim, o SP faz uma requisição para obter o perfil do cliente final (e-mail, nome, sobrenome e etc.).

O fluxo descrito acima faz parte do protocolo Core do OpenID Connect, que é o mínimo que uma ferramenta deve implementar para ser considerada certificada. Além das funcionalidades deste protocolo Core, a plataforma possui uma coleção de outras definições para casos específicos, podemos citar como exemplos: o Discovery, responsável por definir como os clientes podem descobrir *OpenID Providers* automaticamente; a *Dynamic Registration*, para o registro dinâmico de clientes; e até regras para controlar a sessão do usuário e o *logout* (*Session Management*) [20].

Como a plataforma é composta de um conjunto de normas e especificações, diferentes empresas e comunidades desenvolveram implementações próprias. Portanto, existem diferentes bibliotecas escritas nas mais variadas linguagens, como Python, Java e PHP, que implementam e são certificadas de acordo com o OpenID Connect [21].

2.3. Dispositivos Âncoras

A fim de habilitar a credencial de proximidade proposta pelo modelo, foi proposto o uso de dispositivos que podem ser referenciados geograficamente. Isso é possível pois são aparelhos que atuam como pontos de referência fixados em locais físicos estratégicos (com exceção do *wearable*). Por isso, dependendo da aplicação, é necessário que as âncoras sejam protegidas por controles de acesso físico, pois, se posicionadas em locais inseguros podem resultar no acesso de usuários não permitidos.

Em geral, tais dispositivos devem possuir as seguintes características: ter conexão (mesmo que temporária) com a internet, local e armazenamento e emitir um sinal por algum canal de comunicação (óptico, Wi-Fi, Bluetooth, NFC e etc.). As seções a seguir detalham os dispositivos: Beacons, QRCode e Wearable e como podem ser utilizados como dispositivos âncoras.

2.3.1. Beacons

Os dispositivos beacons são pontos fixos de acesso de redes sem fio (como Bluetooth e o IEEE 802.11), que transmitem dados de forma contínua sem a necessidade de pareamento prévio [22]. Esse tipo de aparelho possui um hardware limitado, contendo apenas um processador, módulo de rádio e baterias.

Devido a essas características, os *beacons* são comumente utilizados para transmitir dados pequenos, como por exemplo um ID, a um curto alcance (cerca de 30 a 40 metros). Um exemplo que ajuda a entender o conceito é o do supermercado. Imagine que um cliente ao entrar na loja um *beacon* se comunique com seu smartphone, em seguida o aplicativo instalado exibe as promoções do dia daquele supermercado, ou ainda, a cada corredor que o cliente entra ele recebe quais produtos estão naquelas prateleiras, tudo isso sem a necessidade de interação com o usuário. A Figura 6 exibe um exemplo de *beacon* (à direita), da marca Estimote, se comunicando com um smartphone via Bluetooth para informar de promoções para venda de produtos.

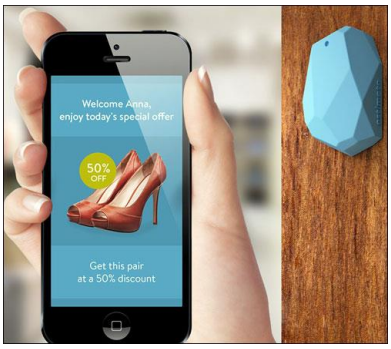

Figura 6 - Exemplo da utilização de um Beacon – Adaptado de [22].

Outra aplicabilidade dos beacons bastante explorada é em sistemas com contexto, onde o ambiente que o usuário está inserido é utilizado com regra de negócio. Os trabalhos [23, 24 e 25] são exemplos de sistemas que utilizam beacons para determinar a localização dos usuários, sendo que o segundo foca em cenários onde o ambiente de contexto é fechado, como em edifícios.

Portanto, é possível determinar a localização de um cliente por meio do sinal recebido de um beacon, pois, devido ao seu curto alcance, se o usuário recebeu o ID ele está próximo ao aparelho, assim, é só manter uma relação, em um banco de dados, entre o ID e a localização daquele dispositivo em particular.

2.3.1.1. Protocolo Eddystone

Devido ao fato de serem implementados com a especificação do Bluetooth, os beacons podem ter algumas variações no modo como transmitem seus dados, dependendo assim da implementação de cada fabricante do aparelho. Existem três padrões que dominam o mercado: o Eddystone (Google), iBeacon (Apple) e AltBeacon (Radius Networks).

O padrão Eddystone é um formato aberto que pode ser detectado tanto por aparelhos Android quanto iOS. Ele possui diversos componentes que podem ser utilizados para transmitir seus dados de diferentes maneiras [26], são eles:

- Eddystone-UID: transmite um identificador estático de 16 *bytes*;
- Eddystone-URL: uma URL comprimida que pode ser acessada pelo cliente;
- Eddystone-TLM: dados sobre as condições do *beacon* úteis para a sua manutenção e funcionamento. Devem sempre ser intercalados com informações de identidade, como o UID ou EID;
- Eddystone-EID: identificador criptografado e com variação temporal, pode ser lido por qualquer dispositivo, porém, apenas usuários autorizados conseguem validar o dado recebido.

Cada componente possui maior relevância em determinadas aplicações, por exemplo, na divulgação de um evento ou de um aplicativo o formato URL é o mais indicado. Nos cenários onde a privacidade e a segurança são requisitos principais, o uso do Eddystone-EID (*ephemeral identifier*) é o mais apropriado. Este componente transmite um identificador criptografado que muda a cada intervalo de tempo, sendo que apenas quem possuir uma chave compartilhada com o *beacon* pode resolver o identificador [27]. Neste cenário é comum que haja uma aplicação online que atue como um "*resolver*", assim, o cliente lê o sinal do *beacon* e envia para a nuvem para resolver aquele ID, ou seja, para que alguma informação de contexto relevante possa ser extraída.

O protocolo obriga que um Beacon deve manter os seguintes dados para a geração do EID [28]:

- **identity-key**: uma chave de 128-bit do tipo AES (*Advanced Envryption Standard*) compartilhada entre o beacon e seu resolver;
- **beacon-counter**: um contador de 32-bit para armazenar os segundos, sendo que não precisa indicar o tempo atual, apenas ser incrementado a cada um segundo; e
- **expoente de rotação (K)**: um inteiro entre 0 e 15 correspondendo a potência de 2 para indicar o tempo que o EID permanece ativo, sendo 2^K segundos.

Para computar o valor do EID duas fases são necessárias, a primeira é para gerar uma *temporary key* e a segunda utiliza essa chave para criptografar os dados que compõem o EID. Os dados necessários para a geração da *temporary key* são criptografados pela *identity key*, e a sua estrutura está definida na Figura

7. Vale ressaltar o uso do contador aqui, assim, cada vez que este procedimento é executada uma nova chave é gerada, pois o valor do contador foi incrementado.

Byte offset	Field	Description
0 - 10	Padding	Value = `0x00`
11	Salt	Value = `0xFF`
12 - 13	Padding	Value = `0x00`
14 - 15	TS[0], TS[1]	Top 16 bits of the beacon-specific time counter in 16-bit big-endian format

Figura 7 - Estrutura de dados para computar *temporary key* do EID – Adaptado de [28].

Em seguida, a chave resultante da primeira fase é utilizada para criptografar a estrutura de dados exposta na Figura 8, sendo que apenas os 8 primeiros bytes do resultado do algoritmo AES são utilizados como o EID. Assim, ao final do período de rotação as duas fases são novamente executadas com os parâmetros modificados, resultando num valor diferente do anterior.

Byte offset	Field	Description
0 - 10	Padding	Value = `0x00`
11	K	Rotation period exponent
12 - 15	TS[0]...TS[3]	Beacon time counter, in 32-bit big-endian format. The K lowest bits are cleared.

Figura 8 - Computando o EID – Adaptado de [28].

Portanto, com a solução alcançada pelo Eddystone-EID os *beacons* podem ser utilizados apenas por quem realmente foi autorizado, ao invés de transmitir um identificador estático em texto puro, onde qualquer cliente *bluetooth* consegue ler e entender. Essa característica torna este protocolo ideal para aplicações onde a localização do usuário é informação de extrema relevância.

2.3.2. QRCode

Apesar de não precisar estar necessariamente em um dispositivo, o QR Code (*quick response code*) também pode ser usado para a verificação da localização do usuário. Este código é um símbolo bidimensional, inventado em 1994 pela Toyota para uso no controle de produção de peças automotivas [29]. A partir disso, o QR Code começou a ser utilizado em diversos cenários, como: controle de passageiros, divulgação de produtos e autenticação [30]. Inclusive o aplicativo de mensagens WhatsApp utiliza esta técnica.

Os símbolos QR Code conseguem transmitir um grande volume de dados (cerca de 7 mil caracteres), compactando-os em uma matriz de pontos, que pode ser lida rapidamente de qualquer ângulo por aplicativos decodificadores. A estrutura da matriz é distribuída na forma quadriculada, sendo que seu tamanho mínimo é de 21x21 células e o tamanho máximo de 177x177 [29].

Pensando no contexto dos dispositivos âncoras, a utilização do QR Code é bastante parecida com a do beacon, pois, ambos podem ser fixados no local onde o usuário pode se autenticar e transmitir o ID criptografado, para que assim o sistema possa determinar a localização do usuário. A diferença está no canal onde o ID é transmitido, sendo que no Beacon esse processo se dá via Bluetooth (na maioria dos casos), e o QR Code precisa ser lido pela câmera de um smartphone, por exemplo.

2.3.3. Wearable

No mundo da informática a busca pela computação pessoal esteve presente nos tempos dos primeiros PCs (*Personal Computer*). Desde então, as pessoas procuram dispositivos que sejam mais próximos das suas necessidades, e, a cada novo avanço das tecnologias, chega-se mais perto desse objetivo. Além do PC, há outros dispositivos criados com esse propósito, como: notebooks, smartphones e o mais recente deles, os wearables.

Existem diversas definições de wearables na literatura, uma delas é a de [31]: são computadores ou dispositivos eletrônicos em miniatura que são vestidos no corpo humano ou anexados em suas roupas. O trabalho [32] definiu as características que um wearable deveria ter:

- **Mobilidade**: estarem acessíveis a qualquer hora e onde o usuário estiver;
- **Usabilidade**: possuir interfaces que habilitem o uso das suas funções mesmo quando os usuários estiverem envolvidos, tanto fisicamente quanto mentalmente, em atividades do mundo real;
- **Aumentar capacidades**: melhorar capacidades dos humanos, como memória, visão, cálculos;
- **Baseado em contexto**: ser capaz de definir o ambiente e as variáveis que rodeiam o usuário.

Para atender os requisitos de mobilidade e usabilidade outras características tiveram que ser sacrificadas, como é o caso do poder de processamento e de armazenamento [33]. Porém, devido à evolução da miniaturização dos componentes alguns wearables estão ganhando processadores melhores, especialmente os relógios inteligentes.

A Figura 9 apresenta um exemplo de dispositivo vestível, o Pebble 2, um relógio campeão de venda na Amazon que possui processador ARM Cortex M7 de 144 MHz, 16MB de memória RAM e sua bateria tem autonomia de até 10 dias.

Figura 9 - Relógio inteligente Pebble 2 – Adaptado de [34].

Portanto, apesar de sua mobilidade, um Wearable pode atuar como âncora em cenários onde o usuário pode estar em diferentes locais para se autenticar. Nesses casos, o importante é garantir que o indivíduo, tentando se autenticar, esteja vestindo um wearable válido e não próximo a algum ponto de referência. Por ser um objeto pessoal, pode-se vincular um Wearable a apenas um usuário, proibindo assim que outra pessoa tente se autenticar no sistema utilizando um dispositivo que não lhe pertence.

2.4. Criptografia de Curva Elíptica

A fim de alcançar o cumprimento dos requisitos básicos de segurança, tais como, confidencialidade e integridade, em dispositivos com processamento limitado, as chaves criptográficas de curva elíptica (ECC) podem ser uma boa opção. Esse tipo de criptografia utiliza do problema do logaritmo discreto e oferece um nível de segurança equivalente a outros modelos, porém, com um tamanho menor de chaves [35]. As definições das funções e conceitos matemáticos e algébricos que sustentam o funcionamento das chaves ECC estão fora do escopo deste trabalho.

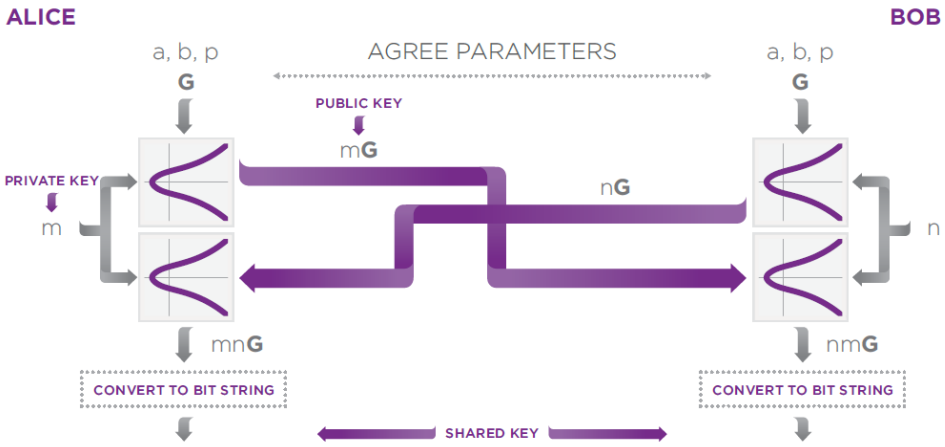

Figura 10 - Protocolo ECDH – Adaptado de [36].

Essas chaves geralmente são usadas juntamente com algum protocolo de *key agreement*, ou seja, procedimentos que duas ou mais partes utilizam para chegarem a um valor em comum, e assim, derivar uma ou mais chaves para criptografia simétrica [35]. O algoritmo ECDH (*Elliptic Curve Diffie-Hellman*), representado na Figura 10, é um exemplo desse tipo de procedimento específico para curvas elípticas.

Supondo que duas partes (Alice e Bob) queiram estabelecer um segredo utilizando o ECDH. Eles precisariam ter alguns parâmetros em comum (a, b, p e G na Figura 10) e então gerar um inteiro aleatório para usar como chave privada (neste caso m para Alice e n para Bob). Em seguida, cada um multiplica o ponto base G com a sua chave privada para formar as chaves públicas a serem trocadas. Com isso, eles multiplicam a chave pública recebida com a sua própria chave privada, gerando um novo ponto que é comum entre as duas partes. Por fim, eles convertem este último resultado (mnG e nmG) em uma *string*, a qual será utilizada como a chave compartilhada.

Neste trabalho, foi utilizado um esquema de chaves baseada na Curve25519, que é uma instância de curva elíptica. Ela utiliza as funções criptográficas X25519, definidas na RFC7748 [37]. A Curve25519 foi projetada para cenários de alta performance, especialmente em casos onde se deseja reutilizar o mesmo par de chaves para trocas futuras, e seu nível de segurança é 128-bit, sendo comparável com a NIST P-256.

2.5. Time-based One-Time Password (TOTP)

É comum na autenticação multifator que um dos parâmetros seja um valor temporal e único, que, uma vez usado deixa de ser válido. Nesta proposta foi utilizado o algoritmo *Time-*

Based One-Time Password (TOTP) como sendo o valor temporal. A RFC 6238 [38] define os detalhes de como se deve utilizar o fator tempo para gerar valores pseudoaleatórios.

O TOTP é uma extensão do *One-Time Password* (OTP), definido na RFC 4226 [39]. Ambos utilizam uma função *hash*, por exemplo o HMAC-SHA-256 ou 512, para gerar o valor pseudoaleatório, a diferença é que o TOTP considera o fator tempo para a geração dos valores. Os parâmetros necessários para isso são: um segredo compartilhado entre as partes interessadas; e um inteiro representando a variação temporal. Para conseguir este segundo, primeiro é definido um contador inicial e depois quando necessário é calculado quanto tempo se passou de acordo com o contador e o tempo atual. Pode-se encontrar essa solução tanto na literatura [40] quanto em cenários do cotidiano, especialmente em bancos. A Figura 11 ilustra um exemplo de um token físico que utiliza o padrão OTP.

Figura 11 - Token OTP físico – Adaptado de [41].

Sendo um padrão amplamente utilizado e com algoritmos já consagrados, o grande desafio para evitar que o código gerado seja imitável é a escolha de uma semente segura. Caso este dado seja comprometido um atacante pode facilmente chegar no valor atual do TOTP, por isso, na seção 4.1.3 é detalhado um esquema de geração de semente que visa mitigar este problema.

Capítulo 3

Trabalhos Relacionados

Este capítulo elucida os principais trabalhos encontrados na literatura, com ênfase na autenticação baseada em localização ou proximidade.

3.1. Autenticação baseada em localização

Esta seção apresenta os trabalhos disponíveis na literatura que são relacionados tanto com a autenticação baseada em proximidade quanto à baseada em localização do usuário. Apesar de existirem uma quantidade significativa de trabalhos que utilizam a localização do usuário para oferecer algum serviço, são poucos os que a usam no processo de autenticação.

O trabalho realizado em [7] utiliza dois PDAs (*personal digital assistant*) para obter a proximidade, um sendo o Beacon e um para o dispositivo móvel do usuário. O primeiro atua como o servidor e utiliza uma infraestrutura de chave pública (PKI) e certificado X.509. O modelo funciona da seguinte maneira: uma fase de setup é necessária para gerar as chaves RSA e certificados, o administrador do sistema salva no beacon o certificado do cliente. Em seguida, na fase de *enrollment*, o dispositivo do cliente faz uma busca no ambiente para identificar e se conectar com o beacon. Um protocolo do tipo desafio-resposta, com bases nos certificados, é executado para garantir que um dispositivo válido seja registrado.

Uma vez que a conexão foi estabelecida com sucesso o PDA entra no estado autenticado, podendo assim habilitar funções que estão relacionadas com aquele beacon em particular. Uma mensagem contínua do tipo *keep alive* garante que os dois PDAs continuem próximos. Apesar de trazer robustez para o modelo, a proposta apresenta algumas limitações:

- O beacon depois de feita a fase de setup pode autenticar apenas o mesmo usuário fazendo com que seja de uso pessoal. Além disso, o fato de o pareamento Bluetooth ser necessário compromete a usabilidade do usuário e sujeita a proposta à ataques do tipo *man-in-the-middle*;

- O uso das chaves RSA e do algoritmo de desafio-resposta traz um grande custo computacional aos dispositivos, especialmente com a utilização de wearables;
- Caso o PDA fixado for comprometido, seja por roubo de certificado ou negação de serviço, todo o sistema é interrompido.

A proposta de [5] visa proporcionar a autenticação baseada em um ambiente mais colaborativo. Este trabalho, chamado de LINK (*Location authentication through Immediate Neighbors Knowledge*), visa determinar a localização de um dispositivo com base nos seus vizinhos na rede Bluetooth. Cada usuário (*claimer*) que deseja utilizar um LBS (*Location Based Service*) precisa emitir um broadcast para todos os seus vizinhos que estão próximos. Em resposta, os vizinhos enviam uma mensagem de certificação ao servidor LCA (*Location Certification Authority*) disponível na internet; além disso, o dispositivo que deseja a autenticação também deve enviar a sua localização. O LCA então, que mantém o registro de todos os *claimers*, no momento da autenticação verifica a pontuação e a localização dos usuários que enviaram as mensagens e relaciona com os dados recebidos do usuário a ser autenticado. Essa pontuação é medida com bases em interações passadas.

O fato de que todas as mensagens trocadas dentro do ambiente LINK são criptografadas, e o uso de dados históricos no LCA, auxiliam na detecção de ataques. A análise de segurança feita pelos autores mostrou sucesso em identificar que *claimers* maliciosos façam requisições de localização não permitida. Porém, há uma limitação em tratar de ataque com um número significativamente grande de *claimers* maliciosos. Apesar de bons resultados apresentados com o protótipo implementado na plataforma Android, o modelo ainda possui alguns desafios, como os ruídos na frequência do Bluetooth, ou mesmo quando não há dados históricos suficientes para uma análise precisa do LCA. Além disso, apenas um canal e um fator foram utilizados no processo de autenticação.

O trabalho [3] apresenta uma abordagem diferente na verificação da proximidade do usuário preservando a privacidade da sua exata localização. O objetivo é garantir uma comunicação segura entre dois clientes (Alice e Bob) de acordo com sua proximidade, ou seja, só é possível se comunicar com clientes dentro de uma mesma posição geográfica. A técnica utilizada para isso é o cálculo da similaridade entre as características físicas do ambiente compartilhado com as que os clientes reportaram no momento da autenticação; e o cálculo é feito utilizando um método probabilístico chamado de *Infinite Gaussian Mixture Modelo* (IGMM).

As características utilizadas no cálculo da similaridade são os dados provenientes basicamente de aparelhos *Access Point* (AP), como por exemplo o *Received Signal Strength*

Indicator (RSSI), o endereço MAC (*Media Access Control*) e também uma sequência de números (SN) representando o tempo de chegada dos pacotes de rede. Esses dados formam duas TAGs de localização, uma privada e uma pública; assim, após detectar os sinais do AP o cliente que deseja se comunicar com a outra parte deve enviar a sua TAG publica. Com isso, após verificar se os dispositivos estão no mesmo ambiente, uma chave de sessão é criada com o uso de índices comuns nos pacotes dos clientes para criptografar as mensagens entre as partes.

Apesar da precisão na detecção de clientes próximos, o sucesso da solução do trabalho [3] depende bastante do ambiente externo, especialmente dos pacotes enviados pelo AP. O que pode gerar vulnerabilidades críticas, como no caso de um AP malicioso ser utilizado como fonte de dados para a geração da TAG. Além disso, a técnica apresentada não possui como foco principal a proteção de algum recurso, mas sim a garantia de uma comunicação segura contra a ataques do tipo *man-in-the middle* ou *spoofing*.

Outro trabalho intimamente relacionado é descrito em [6], onde os autores também propõem um método para identificar a localização do usuário preservando a sua privacidade. Neste trabalho o objetivo é conseguir que um *Service Provider* (SP) apenas entregue o serviço proposto se o usuário estiver em uma área de cobertura válida. Para isso, é utilizado o *Mobile Network Operator* (MNO) presente na maioria das redes de telefonia móvel. A ideia é utilizar uma credencial para identificar o telefone, como o IMEI (*International Mobile Equipment Identifier*) emitida pela operadora e outra para o usuário emitida pelo SP, sendo que a identidade de um usuário é composta por ambas as credenciais.

O funcionamento da proposta [6] inicia com o smartphone requisitando o serviço ao SP, então, um desafio envolvendo a área de referência daquele usuário é retornado. Em seguida, o smartphone solicita ao MNO a credencial de localização com um *timestamp*, com isso o cliente consegue resolver o desafio e se autenticar no SP. Como o desafio solicita apenas saber se a área reportada está contida dentro de uma área de referência maior pré-estabelecida, a posição real do usuário não se faz necessária, preservando assim essa informação de atacantes. Contudo, apesar da preservação da privacidade do usuário o modelo apresenta a limitação de ser dependente exclusivamente do MNO na detecção da localização, inviabilizando a solução em locais sem sinal de cobertura ou em momentos de manutenção.

Outras duas técnicas de autenticação apresentadas no trabalho [4], foram chamadas de STAT I (*Space-Time Authentication Technique*) e STAT II. A primeira técnica utiliza de um terminal de autenticação com módulo GPS (*Global Positioning System*) para a extração da localização do usuário, pois este aparelho o acompanha no deslocamento. No momento da autenticação, o servidor AAA (*Authentication Authorization Accounting*) indaga ao terminal

qual a posição atual enviando uma mensagem criptografada com uma chave simétrica. Ao obter e descriptografar a mensagem, o terminal extrai o dado chamado de *Space-Time* (ST) e o envia ao servidor também usando a criptografia de chave simétrica. Por fim, o servidor verifica se o ST informado está dentro da região física permitida e retorna ao resultado da autenticação. Apesar de os autores não especificaram os detalhes do terminal (tais como o hardware) que acompanham o usuário, eles ainda propõem que para usar o aparelho o seu dono deve identificar-se utilizando biometria.

A técnica STAT II [4] se diferencia apenas pelo processo de determinação da posição, sendo necessária uma entidade âncora para emitir um sinal de baixo alcance. Foi utilizado um aparelho baseado no padrão de comunicação proprietário chamado IQRF, que envia sinais de rádio frequência de baixo alcance. Com isso, para conseguir ler o sinal da âncora deve-se estar próximo dela, possibilitando assim que ao chegar a requisição no servidor ele possa autenticar, ou não, o usuário. No entanto, o trabalho [4] não apresentou um protótipo ou implementação da proposta demostrando sua viabilidade, mas, pode-se destacar o risco de utilizar GPS como fonte da localização, pois é uma abordagem forjável.

3.2. Considerações

Apesar de existirem excelentes trabalhos com propostas viáveis na autenticação baseada em localização, nenhum dos consultados apresenta uma solução integrada com algum IdM ou OpenID. Na Tabela 1 estão representados os recursos de cada trabalho relacionado pesquisado, assim como os que estão presentes na proposta deste estudo.

Tabela 1 - Comparação dos trabalhos relacionados.

Trabalho	**Multicanal**	**Multifator**	**IdM ou OpenID**	**CP forjável**	**Principais Limitações**
Jensen et al. [7]	Não	Não	Não	Não	Ponto único de falha no PDA atuando como beacon; O uso dos beacons é pessoal, necessitando de um para cada usuário.

Trabalho	**Multicanal**	**Multifator**	**IdM ou OpenID**	**CP forjável**	**Principais Limitações**
Talasila et al. [5]	Não	Não	Não	Não	Ruídos na frequência do Bluetooth dificultam a autenticação; Dificuldade na identificação de ataques com número elevado de *claimers* maliciosos;
Xiao et al. [3]	Não	Não	Não	Sim	Procedimento com AP podem ser imitáveis, especialmente se o aparelho for malicioso;
Carmenisch et al. [6]	Não	Sim	Não	Não	Depende da rede de telefonia móvel, fator externo que pode causar gargalo;
Jaros et al. [4]	Sim	Sim	Não	Sim	Técnica de determinação da localização com GPS, que pode ser imitada;
Proposta deste trabalho	Sim	Sim	Sim	Não	Será abordada no próximo capítulo.

Capítulo 4

Proposta

Neste capítulo é apresentado a proposta de um modelo de autenticação multicanal baseada na proximidade do usuário. A princípio, apenas as especificações de cada componente do modelo são descritas, posteriormente, serão descritas as tecnologias concretas que implementam as suas respectivas especificações.

4.1. Modelo

No modelo proposto, a autenticação é baseada na proximidade do usuário com um dispositivo âncora, que deve ser fixado em um determinado local (sala de operações, por exemplo). A comunicação entre usuário e dispositivo âncora é multicanal para evitar que, caso um componente seja comprometido, todo o sistema de autenticação seja comprometido. Dessa maneira, foram propostos três métodos de autenticação baseado em proximidade que podem ser utilizados individualmente ou combinados.

A Figura 12 apresenta os componentes do modelo, que é composto pelos dispositivos âncoras (com seus respectivos canais), o sistema de gestão de identidades (IdM) e dos recursos que devem ser protegidos. As seções a seguir detalham cada um destes componentes.

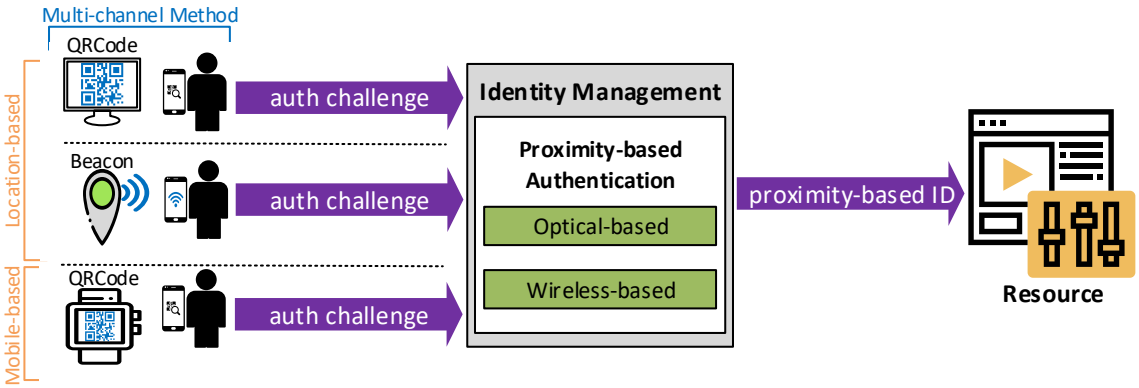

Figura 12 - Modelo de autenticação baseado em proximidade.

4.1.1. *Identity Management* (IdM)

O IdM é um serviço responsável por gerenciar os atributos de identidades do usuário, como login, senha e biometria e autenticá-las quando requisitado. Além disso, também são gerenciadas por ele as credenciais de proximidade dos usuários. Para isso, o serviço possui um módulo responsável por validar as credenciais de proximidade emitidas por dispositivos âncoras previamente cadastrados.

A Figura 13 ilustra a visão geral de como é feita a fase de registro.

Figura 13 - Módulo de controle de dispositivos âncoras.

O registro dos dispositivos âncoras é realizado apenas pelo administrador do sistema, que deve informar ao IdM os seus próprios atributos de identidade para obter um token de acesso com permissão de registrar dispositivos. Essa fase inicial é executada apenas uma vez e garante que o algoritmo utilizado para gerar a credencial de proximidade seja sincronizado entre IdM e âncoras. Com isso, o esquema ganha robustez, uma vez que o desafio da autenticação não circula pela rede.

Outra responsabilidade do IdM é manter na base de dados uma relação entre locais físicos cadastrados e quais usuários são permitidos de se autenticarem a partir de cada local. Esta incumbência é atribuída ao IdM, pois a verificação se o usuário pode, ou não, acessar o sistema a partir daquele local é feita no momento da autenticação.

O processo de obtenção da credencial de proximidade é um ponto crítico para o funcionamento da proposta, por isso foram propostos três métodos diferentes de ser realizado. Nas subseções abaixo serão abordados os três métodos previstos no modelo.

4.1.2. Método baseado em rede Wireless

Este método utiliza uma rede de baixo alcance, como Bluetooth, para emitir credenciais de proximidade. Essas credenciais são emitidas por um dispositivo fixo, o Beacon, que apesar de utilizar o Bluetooth não exige que os clientes executem um pareamento prévio, como em

outras versões do Bluetooth. Isso ocorre, pois, a transmissão dos dados é feita de forma contínua utilizando o módulo *Bluetooth Low Energy* (BLE), disponível a partir da versão 4 do protocolo.

Assim, o Beacon é fixado em um determinado local, sendo responsável por emitir, de forma contínua, a credencial de proximidade através do sinal de Bluetooth, que comumente tem seu alcance entre 30 a 40 metros. Entretanto, a maioria dos protocolos para Beacons existentes, tem a limitação de sempre emitir o mesmo valor (uma URL ou uma frase qualquer), sendo facilmente forjável.

Para diminuir a possibilidade de um invasor da internet forjar esse valor, adotamos o protocolo Eddystone, que possui o modo de comunicação chamado de Eddystone-EID (Ephemeral Identifier), que transmite um identificador criptografado que muda depois de um intervalo de tempo definido.

Para que a criptografia seja possível, na fase de registro de dispositivo é necessário adotar um protocolo de *key agreement*, que define uma chave simétrica entre o Beacon e o IdM, para que possam gerar o mesmo EID. A chave pode ser tanto fixa durante a vida útil da âncora como também ter uma data de validade, necessitando assim de outra fase de setup ao término da data.

A criptografia é possível graças a uma fase de registro onde ocorre o *key agreement*. Nesse processo, cada parte envolvida possui seu par de chaves Curve25519, sendo que no momento do registro cada um disponibiliza a sua chave pública. Em seguida, por meio do protocolo ECDH, cada host calcula uma chave compartilhada através da multiplicação da sua chave privada com a pública do outro. Assim, por compartilharem da mesma chave, tanto o Beacon quanto o seu Resolver (serviço web) podem gerar o mesmo EID.

Além da chave pública, no momento do registro de um Beacon, devem ser informados os parâmetros: contador, o período de rotação e o ID do local onde está inserido o dispositivo, como por exemplo uma sala de controle ou enfermaria. A Figura 14 mostra a troca de mensagens entre o administrador, o Beacon e o servidor no momento de registro.

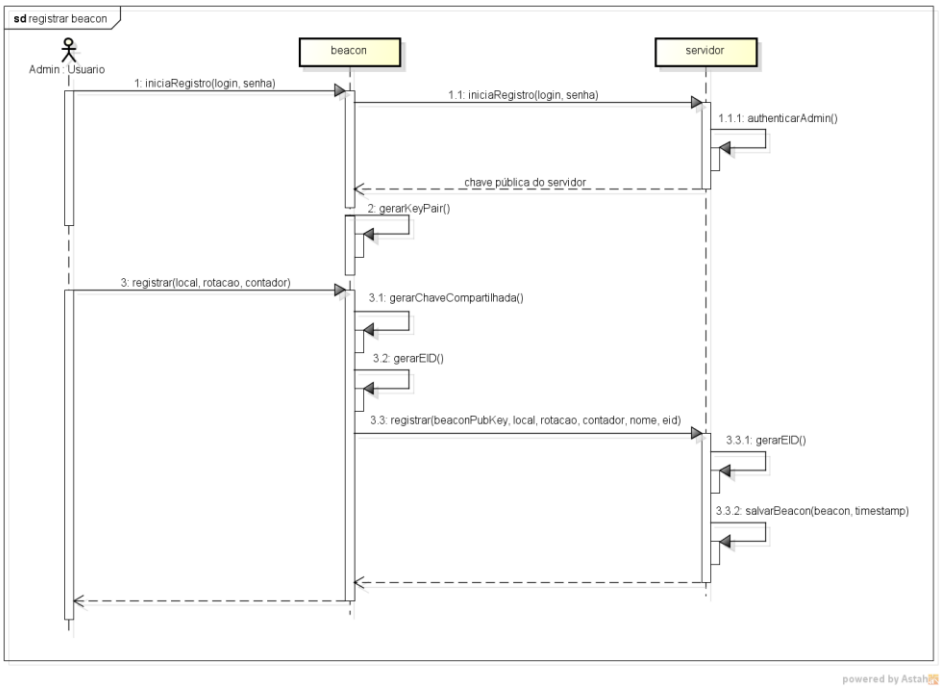

Figura 14 - Registro de dispositivos Beacon

A adoção do protocolo Eddystone com a criptografia de curva elíptica agrega robustez e velocidade na solução, pois o valor do EID é modificado em um intervalo de tempo personalizável. Além disso, a chave compartilhada utilizada na geração do valor nunca foi trafegada na rede.

Assim, o Beacon é responsável por emitir, via broadcast, o EID, que é a credencial de proximidade. O usuário que estiver fisicamente próximo ao Beacon pode capturar, através de seu smartphone, essa credencial e informar ao IdM. Dessa maneira, no momento da autenticação o IdM deve validar o EID recebido. A validação exige que o IdM gere um EID para cada Beacon registrado, até que ele encontre um igual ao valor recebido.

Além das vantagens de segurança providas por esse método, a usabilidade é interessante. Pois o usuário não precisa realizar nenhuma ação ativa para obter a credencial emitida pelo Beacon. Entretanto, a principal desvantagem desse método apresenta-se no caso de o usuário estar em um ambiente com ruídos na frequência do Bluetooth, ele pode ficar impossibilitado de se autenticar. Uma alternativa a esse método de autenticação será discutida na próxima subseção.

4.1.3. Método baseado em leitura óptica

Este método utiliza um canal menos suscetível a ruídos para emitir a credencial de proximidade. Essa credencial é disponibilizada graficamente através de um QRCode apresentado em um display fixado. Este dispositivo que apresenta o QRCode pode ser qualquer dispositivo com conexão à internet que contenha um visor, por exemplo: PC, smartphone, tablet, entre outros. Sendo que a conexão com a internet só é obrigatória no momento do registro do dispositivo no IdM.

A credencial disponibilizada pelo QRCode é gerada por um algoritmo de *Time-based One Time Password* (TOTP). O TOTP é um método comumente encontrado em mecanismos de autenticação multifator que desejam uma credencial temporária e dinâmica. O TOTP utiliza uma função, por exemplo o HMAC-SHA-512, para gerar um valor pseudoaleatório. Os parâmetros necessários para essa geração são: uma semente (usada pela função *hash* para gerar o valor) e um inteiro representando o tempo que a credencial permanecerá ativa. Diante do fato de que estes valores são cruciais para o não forjamento da credencial de proximidade, neste caso o TOTP, a fase de registro é responsável por estabelecer uma semente segura entre as partes. A Figura 15 apresenta a troca de mensagens necessárias para o registro.

O registro inicia com o administrador informando o seu login e senha (1) para o dispositivo QRCode, que verifica se este é um usuário válido (1.1). O IdM então valida as credenciais e emite um *token* de acesso (1.1.1) com duração de poucos segundos. Em seguida, o administrador informa qual o local físico onde o QRCode será instalado (2), um apelido para identificação do dispositivo também pode ser utilizado para ambientes com muitos QRCodes. No passo 2.1 o dispositivo gera o seu par de chaves Curve25519 e o armazena localmente. Com isso, o QRCode é capaz de enviar os dados necessários para o registro: a sua chave pública, o local físico, o apelido e o token de acesso (2.2). Quando o IdM recebe esta requisição ele valida o *token*, e caso seja válido, ele inicia o processo de registro gerando a semente do TOTP (2.2.1). A semente é a chave compartilhada resultante do algoritmo ECDH entre a chave pública do QRCode e a privada do IdM (e vice-versa), portanto, o mesmo procedimento é feito no passo 2.3. Ainda antes de finalizar, o IdM salva os dados do registro na sua base (2.2.2) e retorna ao QRCode três parâmetros: o ID deste dispositivo, um inteiro aleatório entre 90 a 120 representando o intervalo de tempo da credencial; e a data que o registro irá expirar, obrigando o administrador a realizar este procedimento novamente. Finalmente, o fluxo termina quando o QRCode salva a semente, intervalo e data de expiração na sua memória local (2.4). A partir disso o dispositivo está pronto para emitir o valor TOTP.

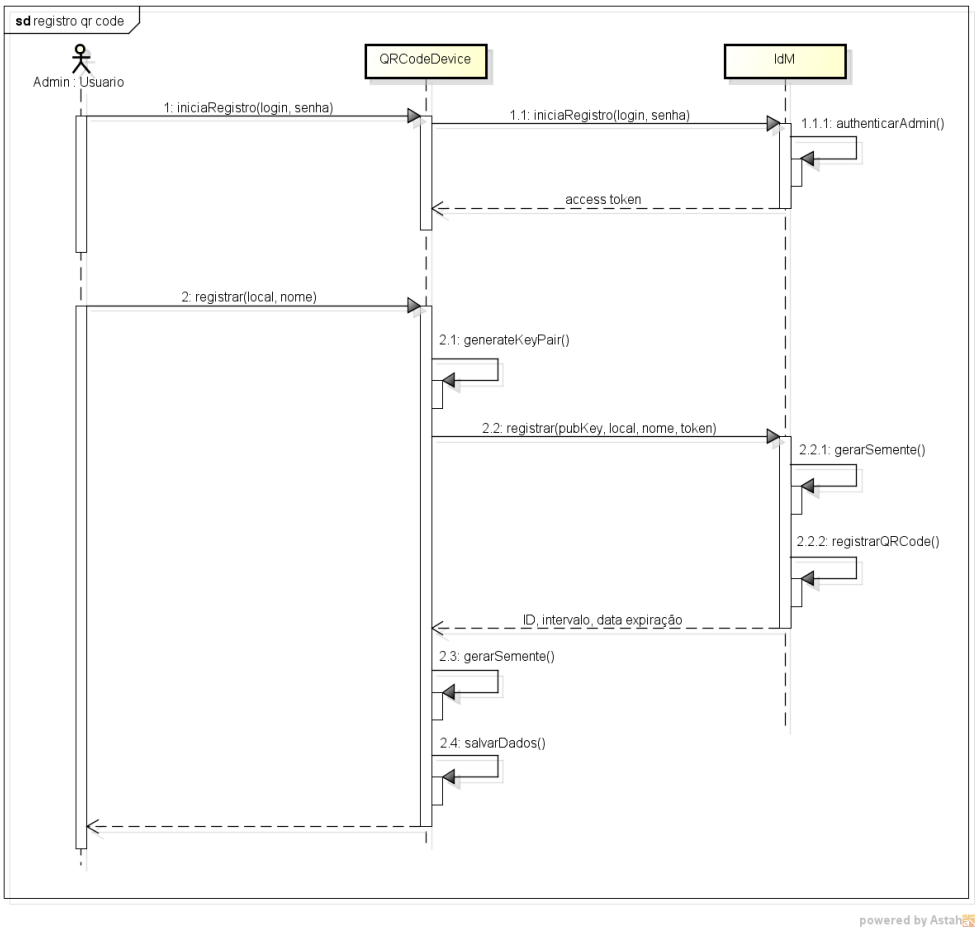

Figura 15 - Registro de dispositivos QRCode.

Dessa maneira, o usuário deve ler o QRCode apresentado pelo dispositivo com uma câmera para obter o valor gerado. Posteriormente, o usuário informa ao IdM este valor para ser validado. Para realizar a validação, o IdM deve buscar os dados no banco pelo ID e depois executar o algoritmo do TOTP. A proximidade do usuário é válida caso o valor gerado pelo IdM seja igual ao valor informado pelo usuário e esteja dentro do intervalo aceitável.

É importante destacar que ambos os métodos destacados (Óptico e Wireless) necessitam que exista um sistema de proteção física que garanta que o dispositivo âncora esteja realmente fixado em uma determinada sala de operações.

4.1.4. Método baseado em Wearable

Este método é similar ao Óptico, utilizando de QRCode e TOTP para transmitir a credencial de proximidade; a diferença é a mobilidade. Neste cenário o dispositivo âncora é um *Wearable* (um smartwatch por exemplo) que o usuário pode vestir e carregar consigo. Assim,

no momento da autenticação o código deve ser lido do *wearable* através do smartphone para ser enviado ao IdM.

A fase de registro neste método também é diferente, pois, o wearable deve ser vinculado a um usuário e não um local físico, isso também garante que a credencial de proximidade emitida por ele não possa ser utilizada por outro usuário. Apesar dessa peculiaridade, o restante do fluxo de autenticação é o mesmo que no cenário anterior, trocando apenas o ID do local físico pelo ID do usuário.

Com este método, o esquema ganha uma flexibilidade para atender casos em que o usuário tem que acessar o recurso, mas não pode estar fisicamente no local. Por exemplo, um médico em período de férias deve acessar o prontuário de um paciente em caso de emergência. Além disso, o modelo continua robusto, porque o usuário deve possuir dois aparelhos no momento da autenticação por proximidade.

Este canal, via dispositivo âncora vestível, pode ser também utilizado em conjunto com algum outro método. O IdM pode ser ajustado para que, ao invés de uma credencial de proximidade o usuário precise fornecer duas, assim, na requisição de autenticação deve ser o valor do Beacon/QRCode e o do Wearable, diminuindo o risco de forjamento das credenciais.

Capítulo 5

Protótipo

Este capítulo detalha a implementação e as tecnologias utilizadas na construção de um protótipo com base nas especificações do modelo apresentado anteriormente. Além da construção do IdM, também foram criados simuladores para os dispositivos âncoras e uma aplicação mobile para o usuário final. Em seguida, avaliações de segurança e de desempenho realizadas com o protótipo são apresentadas.

5.1. Implementação do Protótipo

O protótipo e simuladores foram desenvolvidos com a utilização de padrões, tecnologias de mercado consolidadas e bibliotecas de código aberto. A linguagem Java foi amplamente utilizada devido a sua característica multiplataforma, pois, alguns dos componentes implementados foram feitos para o sistema operacional Android.

A Figura 16 apresenta os componentes, e as tecnologias utilizadas em sua construção, presentes no protótipo desenvolvido. A construção do IdM foi dividida em duas partes: o Autenticador e o Resolver, que serão detalhados na próxima subseção. Os detalhes dos simuladores de dispositivos âncoras, assim como os *screenshots* da sua utilização também são apresentados a seguir.

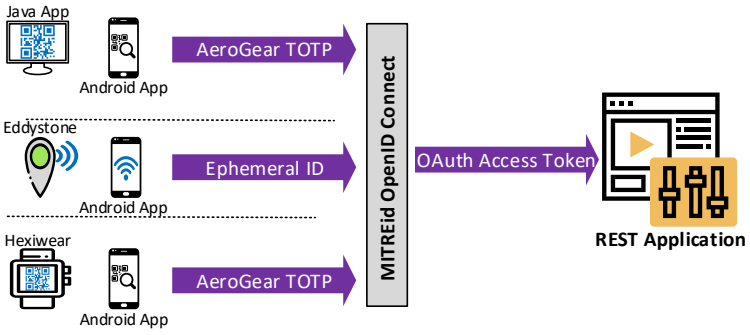

Figura 16 - Componentes do protótipo.

5.1.1. Implementação do IdM

Para o servidor IdM, mais especificamente para o módulo Autenticador, foi utilizado o framework Spring Boot, com os pacotes: o Security para proteger o acesso aos recursos do servidor e realizar a autenticação; o OAuth2 para controle da autorização e dos *tokens* de acesso; o Web para disponibilizar as APIs REST; o Cloud para configurações de escalabilidade e balanceamento de carga e, por fim, o módulo Data JPA utilizado para acesso e manipulação dos dados com o banco PostgreSQL. O uso do Spring Boot deve-se ao fato da grande comunidade de usuários, pacotes disponíveis e atualizações frequentes.

Para o módulo de Resolver, responsável por manter os dispositivos âncoras e validar a credencial de proximidade, foi utilizado apenas os pacotes com as APIs REST do Spring e as bibliotecas Curve25519 da SignalApp [42]; para manipulação das chaves criptográficas, a Bouncy Castle Provider [43]; para a leitura das chaves e a AeroGear OTP [44], uma implementação das especificações *One-Time Password* (OTP) e TOTP, utilizada no método óptico e com *wearable*. Foi utilizado também o gerenciador de bibliotecas e compilador Maven, além do container de aplicação Tomcat para a implantação do artefato final.

A fim de registrar um dispositivo âncora, são necessárias duas requisições POST do protocolo HTTP, uma para autenticar o administrador e obter o *token* com o escopo de "admin", e outra para enviar os dados necessários para o registro. A Figura 17 exemplifica os dois cenários; a primeira é a autenticação e a segunda o registro de um Beacon. As figuras são capturas de tela do programa Insomnia, software que permite fazer chamadas HTTP.

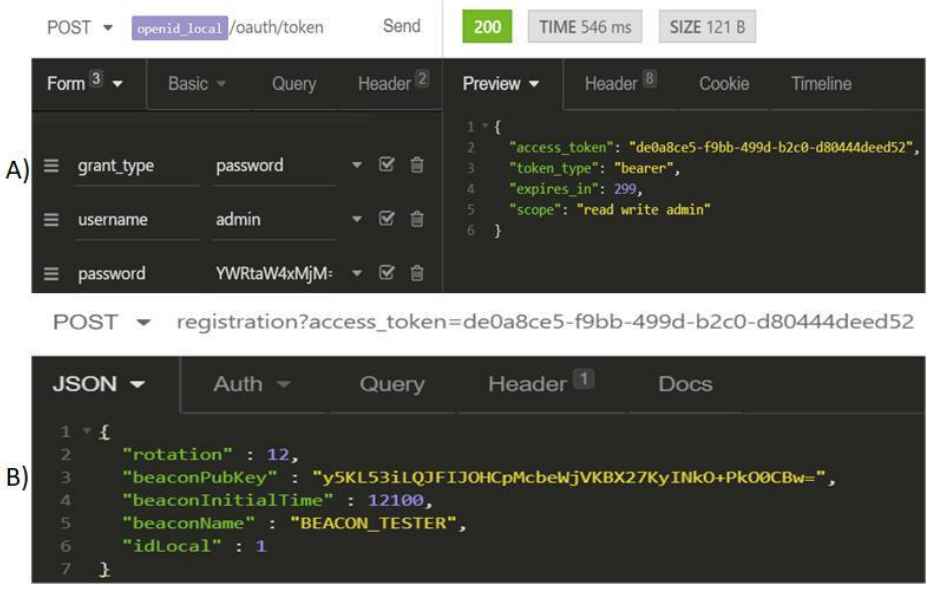

Figura 17 - Servidor IdM: A) requisição de login do administrador e B) registro de um Beacon.

Na Figura 17a) do lado esquerdo estão os parâmetros a serem enviados ao servidor, o *grant_type* é do protocolo OAuth que identifica o tipo de autenticação. Do lado direito estão os dados retornados, em caso de sucesso, no formato JSON. São eles:

- **access_token**: token de acesso utilizado para autorização nas próximas requisições;
- **token_type**: o tipo padrão de token do OAuth é o *Bearer*, que basicamente significa que o portador está autorizado a realizar algum tipo de operação;
- **expires_in**: o tempo de duração do token, após isso deve-se gerar outro *token*;
- **scope**: permissões que o portador possui dentro do sistema, no exemplo, o usuário pode ler, escrever e acessar funções de administrador (registro).

A Figura 17b) mostra os dados necessários para o registro de um Beacon também no formato JSON. Nesta requisição, o *token* de acesso adquirido no passo anterior foi utilizado na própria URL, porém, em ambientes de produção deve-se enviá-lo dentro do cabeçalho HTTP.

A funcionalidade autenticação, validando as credenciais de identidade e de proximidade, será apresentada nas seções seguintes ao discutirmos sobre a implementação dos dispositivos âncoras.

A implementação do IdM proporciona uma solução completa no gerenciamento de identidade e dos dispositivos âncoras. Porém, para ambientes onde já existem um servidor IdM o módulo de Resolver pode ser utilizado separadamente e implantado como uma aplicação Java independente.

5.1.2. Beacon

Apesar de existirem Beacons disponíveis no mercado poucos implementam o protocolo Eddystone-EID. Além disso, a grande maioria depende de um smartphone para acessar a internet. Por isso, para atuar como um Beacon no protótipo foi desenvolvido um simulador para o sistema operacional Android, pois, a grande maioria dos smartphones, além de conexão com a internet, possuem Bluetooth 4.0+, com a versão BLE necessária para o broadcast do EID.

Para o desenvolvimento do simulador, além das bibliotecas Curve25519 e *Bouncy Castle Provider* já mencionadas, foram utilizadas as bibliotecas: Android Beacon da AltBeacon [45], responsável por transmitir os pacotes de dados via BLE; e a Retrofit [46] para as requisições HTTP. Foi utilizado o Gradle como gerenciador de pacotes e compilador, e a versão mínima do Android necessária é a 6.0 (API *level* 23).

As funcionalidades realizadas pelo simulador são: gerar chaves Curve25519, comunicação com o IdM via REST, armazenamento dos parâmetros necessários (período de rotação, contador, chave compartilhada e etc.) e broadcast do EID.

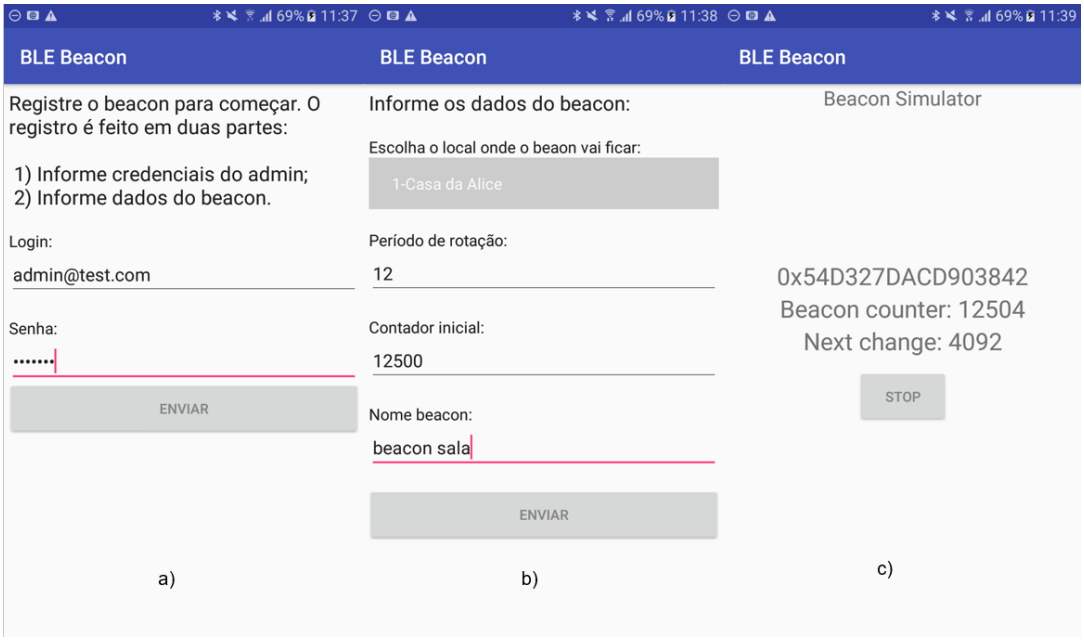

Figura 18 - Telas do simulador Beacon.

A Figura 18 exibe as telas de funcionamento do simulador BLE Beacon rodando em um smartphone Motorola G4 Plus, que possui um processador Qualcomm Snapdragon 617 de 1.5 GHz, 2 GB de memória RAM e Android 6.0. A primeira tela é a primeira etapa do registro do Beacon, onde o administrador deve informar seu e-mail e sua senha. Em seguida, caso o usuário seja autenticado com sucesso, a tela de configuração do Beacon (b) é exibida, e o administrador pode então escolher, a partir de uma lista já cadastrada, qual o local físico que será instalado o Beacon. Além disso, os parâmetros período de rotação, contador inicial e nome do aparelho também podem ser configuráveis.

Por fim, a última tela (c) mostra o Beacon fazendo o broadcast do EID ativo (também exibido na tela); nessa etapa o registro já realizado com sucesso. Nessa tela também foi colocado um botão de “STOP” que o usuário poderá pausar a transmissão da credencial no momento que desejar. Com isso, qualquer cliente ouvindo na frequência do BLE, dentro do range permitido, pode ler a credencial e utilizar no momento da autenticação.

5.1.3. QRCode

Para implementar o método baseado em leitura óptica, foi criado um sistema Java gerador de QRCode, sendo possível ser instalado em diversos dispositivos, inclusive nos que rodam Android. Também foi utilizado a biblioteca da SignalApp para as chaves, e a AeroGear como implementação da especificação TOTP. Além dessas, a biblioteca QRCodeGen [47] foi usada para gerar a matriz de pontos encapsulando o valor da credencial. Uma das vantagens de se utilizar a linguagem Java é a possibilidade de utilizar as mesmas bibliotecas para diferentes plataformas. É importante destacar também que com o aplicativo QRCode para Android habilita-se uma vasta gama de *smartwatches* possíveis para o cenário com *Wearable*.

O registro do dispositivo gerador de QRCode é parecido com o beacon, o administrador deve enviar os parâmetros: a) ID do local; b) chave pública; e c) sua credencial. O IdM então salva os dados no banco, gera um número aleatório de 90 a 120 segundos (tempo que o TOTP ficará ativo) e retorna este valor juntamente com a chave pública do servidor no formato Base64 (conforme é mostrado na Figura 19). Com isso, o dispositivo também pode calcular também a chave compartilhada.

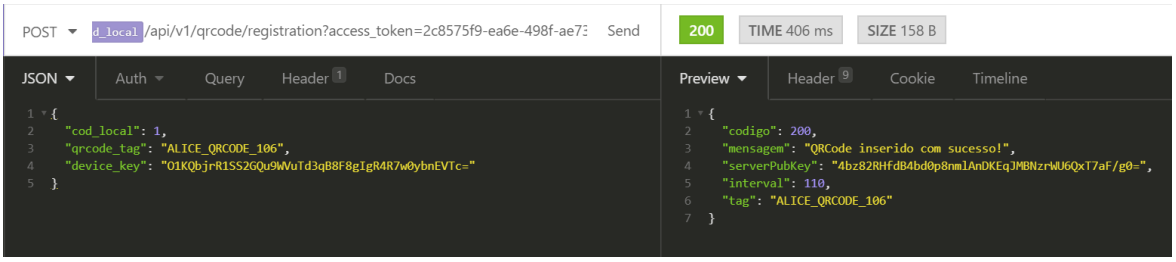

Figura 19 - Requisição de registro de um QRCode.

Como foi dito na definição do modelo do método baseado em leitura óptica, a semente utilizada no algoritmo gerador do TOTP nunca é trafegada pela rede. Ao invés disso, a biblioteca da SignalApp possui um método de key agreement baseado no ECDH que deriva a mesma chave compartilhada, tanto utilizando a chave privada do IdM com a pública do QRCode, quanto com a privada do QRCode e a pública do IdM, gerando uma chave simétrica. Por esse motivo a biblioteca deve estar presente nas duas partes que desejam se comunicar, e, a partir desse procedimento a semente é estabelecida, e salva, em locais seguros, como por exemplo a KeyStore [48] do Android.

Na Figura 20 estão representadas as telas durante a execução aplicativo gerador de QRCode instalado no mesmo smartphone mencionado na seção do Beacon. Na primeira tela

(a) está sendo solicitado que o administrador inicie o registro. Em seguida, local a ser inserido o dispositivo e um apelido para fácil identificação são requeridos (b). Perceba que o envio do token de acesso, a geração do par de chaves e o recebimento da chave pública do IdM são transparentes ao usuário, porém, ocorrem após o envio dos dados da tela b). Por fim, a terceira e última tela (c) já mostra o QRCode sendo exibido com a credencial de proximidade; no canto inferior um contador indica o tempo restante para o valor do código mudar, quando esse contador chega a zero o método da biblioteca AeroGear é invocado novamente e um novo hash, baseado na semente e na variação temporal, é gerado.

Figura 20 - Telas do dispositivo QRCode.

O formato dos dados expostos dentro do QRCode segue o padrão: “ID do local: ID do QRCode: TOTP”. Foi definido dessa maneira para facilitar a busca de qual dispositivo está sendo usado no momento da autenticação, com isso, o IdM pode localizar rapidamente qual a semente e o intervalo registrado para aquele QRCode.

Apesar de ser um método que demanda um maior esforço do usuário (ter que ler o código com uma câmera), o método óptico é um canal livre de ruídos e detectável apenas naquele local. Outro ponto positivo é que os detalhes de sua implementação não limitam as opções de aparelhos que podem atuar como dispositivo âncora óptico.

5.1.4. Wearable

Como protótipo para o cenário do *wearable* foi utilizado um smartwatch de hardware livre, comumente usado para prototipação, chamado de Hexiwear [49]. O relógio possui display com *touch*, BLE, processador ARM Cortex-M4 de 120 MHz, 256K de SRAM além de acelerômetro, giroscópio e sensor de pressão. Quando acoplado a uma estação de trabalho (*Docking Station*) o aparelho ganha funcionalidades extras, como por exemplo conexão com a

internet via WiFi, possibilitando que ele se comunique com o IdM para a fase de registro. A Figura 21 mostra o relógio acoplado na estação de trabalho juntamente com uma placa com módulo de comunicação wireless.

Para a programação da plataforma Hexiwear, foi utilizada a linguagem Python com a IDE Zerynth Studio [50], sendo este utilizado para a programação de diversos microcontroladores. As bibliotecas utilizadas foram: a Pil Image [51] para o redimensionamento da imagem, pois a resolução do display é de apenas 90px x 90px; a PyQRCode [52] para criação da matriz de pontos e a PyOTP [53] para gerar o valor pseudoaleatório a ser transmitido. Além disso, uma implementação em Python do Google para gerar EID [54] foi utilizada, pois esse código possui uma implementação das chaves Curve25519. A implantação do código é feita conectando o relógio na estação de trabalho e a estação no computador via USB.

Figura 21 - Smartwatch Hexiwear e Docking Station.

O objetivo desta implementação é fazer com que um hardware real de wearable implementasse a mesma funcionalidade do protótipo com o QRCode, descrito na seção 5.1.3.

Porém, devido a algumas limitações da plataforma Hexiwear não foi possível fazer com que o valor do QRCode mudasse ao fim do intervalo estabelecido. Isso porque as bibliotecas utilizadas para o procedimento de geração da imagem com o QRCode só funcionam como esperado dentro da IDE do Zerynth. Assim, a cada vez que o intervalo da credencial chegasse ao fim deveria ser gerado o código na IDE, e depois transmitir os bytes da imagem via USB para o relógio.

Contudo, apesar de a implementação com a plataforma Hexiwear não atender os requisitos do modelo para uma âncora válida, uma outra opção de relógio poderia ser utilizada com poucas alterações sendo necessária. Por exemplo, o relógio Moto 360 [55] da Motorola possui o sistema operacional Android Wear, uma variação do sistema presente nos smartphones, e ainda conexão WiFi 802.11 b/g. Com isso, seria possível adaptar o protótipo feito em Java para o S.O deste relógio.

5.1.5. App *Authenticator*

Por fim, um último aplicativo desenvolvido também para Android foi o *Auhtenticator*, sendo este o meio de interação do usuário final com os outros componentes, e também com os recursos que ele desejaria acessar. Devido ao fato que dentro do protótipo o papel deste app é testar os diferentes canais de leitura da credencial de proximidade, foi concedido ao usuário a liberdade de escolher qual âncora escolher (Beacon, QRCode ou Wearable) no momento da autenticação.

Além das bibliotecas Retrofit e Okhttp, utilizadas na comunicação via REST com o IdM, também foi utilizada uma biblioteca para o escaneamento do QRCode a ZXing Android Barcode Scanner [56]. O aplicativo deve manter os dados criptografados de identificação do usuário localmente, por isso, foi utilizado de um certificado X509 (já incluso no executável) para manter a chave privada RSA responsável pela criptografia. Como a credencial de proximidade está em constante mudança não é necessário o seu registro. O token de acesso recebido do IdM pode, ou não ser salvo, dependendo do tempo ajustado que irá durar a sua validade. Caso seja um tempo curto não há a necessidade de se salvar (como é o caso do protótipo).

A Figura 22 exibe algumas das telas do *Authenticator*. Na primeira A) o usuário digita seu login e senha, na segunda B) ele escolhe qual âncora utilizar: Beacon ou QRCode sendo que este último pode ser, tanto o display fixado em algum local, quanto o código proveniente

do *wearable*. Caso o usuário clique no botão Cenário 1 o aplicativo exibe uma mensagem de procurando por Beacons nas proximidades, e quando encontrado uma requisição com o login e o EID é enviado ao IdM. Por outro lado, caso seja o Cenário 2, a câmera do smartphone é aberta e o usuário deve posicionar para ler o QRCode e então disparar a requisição. Por fim, após a autenticação ser realizada com sucesso e o token recebido, o usuário é direcionado para a tela C) onde os dados críticos ou funções administrativas são ofertadas. Um texto informando o tempo de duração do token é exibido no rodapé da tela, sendo este configurável pelo administrador do sistema.

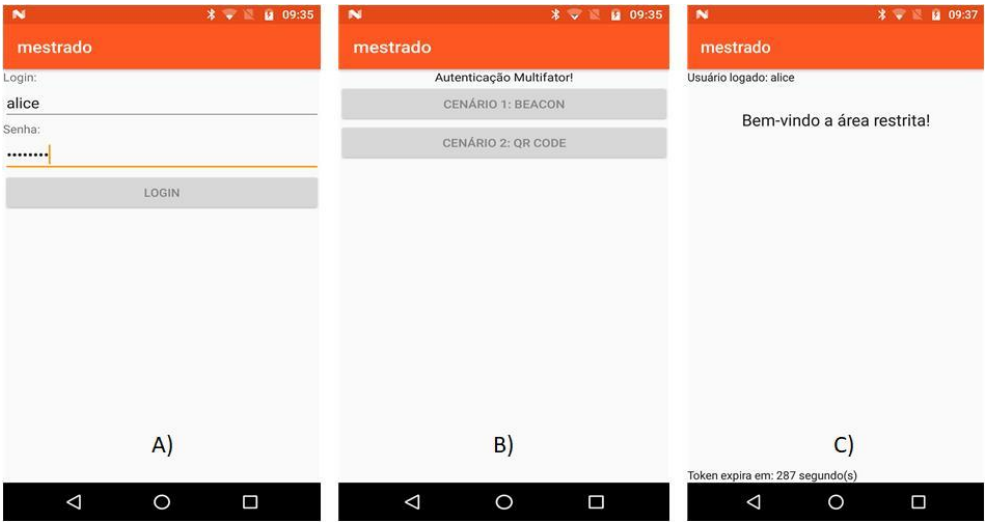

Figura 22 - Telas do aplicativo Authenticator.

5.2. Avaliação de Segurança

A fim de apresentar os possíveis riscos dentro do modelo proposto, a Tabela 2 apresenta os impactos para cada componente da arquitetura que for comprometido por uma parte maliciosa, assim como possíveis contramedidas que poderiam ser exploradas em trabalhos futuros.

Outros componentes tradicionais dentro do IdM, tais como os módulos de controle de acesso e de autorização, devem ter os mesmos cuidados em qualquer outra implementação. Por exemplo, utilizar de sistemas de detecção e decisão em tempo real, modelos preditivos e políticas de respostas a incidentes [57].

Além da avaliação dos riscos do comprometimento de cada componente, também foram executadas análises estáticas dos códigos fontes do protótipo. Para isso, foi utilizado o software SonarQube [58], uma plataforma utilizada para análise de códigos em diferentes linguagens

seguindo um conjunto padrão de regras e métricas. Além de identificar bugs, o SonarQube também traz relatórios de vulnerabilidades, boas práticas, pontos críticos de segurança e testes automatizados [58].

Tabela 2 - Avaliação de Segurança

Componente	Impacto	Contramedidas
Beacon	Caso o atacante tenha acesso aos arquivos do dispositivo, ele conheceria os parâmetros utilizados para gerar um EID válido. O impacto de um atacante escutar a rede Bluetooth é mínimo, pois além do baixo alcance limitar a sua posição física, ele deve enviar o EID de uma aplicação válida ao IdM. Além disso, ele ainda teria de conhecer o login e senha do usuário.	• Fechar a conexão com a internet após a fase de registro; • Controle de acesso físico; • Adicionar data de validade para o registro da âncora; • Posicionar o Beacon em local fechado limitando o alcance; • Invalidar o registro assim que houver um número de usuários errando suas senhas a partir de um determinado âncora.
QRCode	Um código malicioso na câmera do cliente poderia ler o valor e enviar a um atacante remoto, que em posse da senha poderia se autenticar remotamente. O controle do dispositivo também iria dar poder ao atacante reproduzir o valor do TOTP.	• As mesmas medidas do Beacon; • O valor do TOTP só permanece ativo durante no máximo 120 segundos; • Deixar o display com o código desligado quando não utilizado.
Resolver	O atacante poderia registrar novas âncoras fantasmas em regiões perigosas. Porém, seria necessário conhecer as credenciais de identidade de algum usuário válido com privilégios de administrador.	• Desabilitar imediatamente o segundo fator da autenticação e notificar o administrador.
Aplicação	O atacante poderia tomar conhecimento da credencial de localização daquele usuário específico. No entanto, ele só seria capaz de se autenticar caso estivesse próximo a alguma âncora válida.	• Notificar o usuário por outros canais de tentativas de login com a sua credencial, assim ele teria ciência para trocar seus dados de acesso.

A versão utilizada foi a 7.1, assim como plugins para a Java e Android contendo cerca de 498 regras. No que diz respeito às regras segurança, elas são originadas de três organizações de grande mérito no assunto [59]:

- **Common Weakness Enumeration (CWE)**: é uma lista formal de fraquezas relacionadas ao código de um software. O seu principal objetivo é definir padrões para a identificação, prevenção e mitigação dessas fraquezas. Atualmente existem um total de 995 CWEs, divididos em diferentes categorias e áreas de atuação (como acadêmico ou desenvolvedores). Para softwares escritos em Java, há um total de 73 fraquezas listadas atualmente [60];
- **SANS Top 25**: é uma coleção de 25 CWEs com os erros e vulnerabilidades considerados mais perigosos. Essa lista é mantida pela organização SANS², uma instituição referência em treinamentos de segurança, e possui três categorias: Interação insegura entre componentes (os meios em que os dados são transmitidos entre os componentes), gestão de recursos arriscada (políticas de tratamento de recursos) e defesas pífias (mecanismos que são usados incorretamente ou ignorados) [61];
- **OWASP Top 10**: o Open Web Application Security Project é uma organização sem fins lucrativos com o objetivo de auxiliar nas decisões sobre os riscos de segurança presentes no software. O OWASP mantém uma lista de 10 categorias gerais com diversas fraquezas, ataques e impactos mais específicos [62].

A fim de exemplificação, algumas das regras de vulnerabilidades verificadas pelo SonarQube estão descritas na Tabela 3. A origem, sua categoria e uma breve descrição também são mostradas. É importante ressaltar que alguns índices reportados pela verificação da segurança do código podem ser falsos positivos, diferente das métricas de bugs, que quando identificadas realmente são um bug. Isso ocorre porque algumas regras de segurança necessitam do julgamento humano no quesito do quão sensível é um dado, ou não.

Como podemos observar na tabela, são consideradas desde regras gerais já conhecidas e independentes da linguagem (exemplo a injeção de SQL), como também são verificadas regras específicas de códigos e arquivos de configuração do ambiente Java, como é o caso dos filtros e dos interceptors.

Dentro do SonarQube é possível definir os “Quality Profiles” e “Quality Gates”, que são, respectivamente, os perfis de quais regras vão ser usadas (sejam de bugs, code smells ou segurança); e quais os limites aceitáveis para cada categoria para o que teste seja aceito. Como o foco da análise são as métricas de segurança, foi instalado no SonarQube o plugin Find Security Bug [63], que possui um *Quality Profile* já definido para auditoria de segurança em

códigos Java, possuindo cerca de 176 regras, todas de segurança, provenientes das origens comentadas acima.

Tabela 3 - Exemplos de vulnerabilidades SonarQube.

Origem	Vulnerabilidade	Categoria	Descrição
SANS Top 25	CWE-798	Defesa Pífia	Uso de credenciais hard-coded.
SANS Top 25	CWE-22	Gestão de recursos arriscada	Chamada de funções de E/S vulnerável a ataques de injeção de pacotes
SANS Top 25	CWE-89	Interação insegura entre componentes	Vulnerabilidade a ataques de injeção SQL
OWASP	ReDoS	A1 - Injection	Negação de serviço (DS, *Denial of Service*) de expressão regular.
OWASP	*Defined Filters*	A6 – *Security Misconfiguration*	Os filtros declarados no arquivo "web.xml" devem ter definido o atributo <filter-mapping>.
OWASP	EJB Interceptors	A6 – *Security Misconfiguration*	Os *interceptors* EJB padrões devem ser listados no arquivo "ejb-jar.xml".

O *Quality Gate* utilizado nas análise foi o padrão do Sonar, e as condições que ele define para um teste de sucesso são definidos na Figura 23. Como pode-se observar, o projeto deve ter nota A nos quesitos confiabilidade, manutenção e segurança, além de ter menos que 3% de código duplicado e menos que 80% de testes de cobertura. Para as análises do protótipo, os relatórios de testes de cobertura, e os "*code smells*" de menor relevância, não foram considerados.

Foram inspecionados os códigos do servidor IdM, os simuladores Beacon (chamado de BLEBeacon) e QRCode (QRCodeGenerator) e o aplicativo que o cliente utiliza chamado de AuthenticatirApp. O comando para rodar as aplicações baseadas em Maven foi o "mvn sonar:sonar -Dsonar.host.url=http://localhost:9000", enquanto que para aquelas baseadas em Gradle foi utilizado o "gradlew sonarqube -Dsonar.host.url=http://localhost:9000", ambas informando a URL do host onde a ferramenta foi instalada.

Metric	Over Leak Period	Operator	Warning	Error
Coverage on New Code	Always	is less than		80.0%
Duplicated Lines on New Code (%)	Always	is greater than		3.0%
Maintainability Rating on New Code	Always	is worse than		A
Reliability Rating on New Code	Always	is worse than		A
Security Rating on New Code	Always	is worse than		A

Figura 23 - *Quality Gate* padrão do Sonar utilizada como limites – Adaptado de [58].

Diversas análises foram executadas no decorrer do desenvolvimento, a fim de identificar e solucionar as vulnerabilidades, inclusive correções na própria ferramenta. A visão macro dos relatórios de todos os componentes do protótipo está representada na Figura 24. Como é possível observar, a versão final de cada componente passou nos requisitos de segurança estabelecidos pelo perfil FindBugs Security Audit.

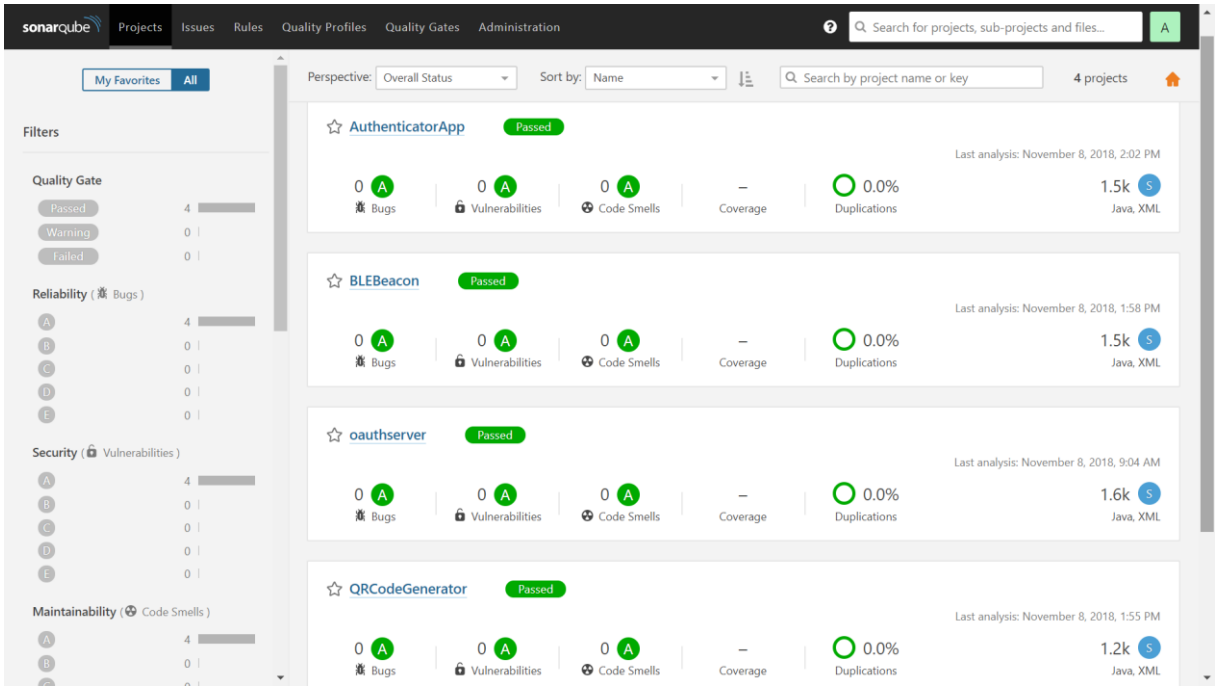

Figura 24 - Visão macro dos relatórios da análise estática dos códigos

5.3. Avaliação de Desempenho

O protótipo também foi submetido a uma avaliação de desempenho, a qual teve por principal objetivo medir a aplicabilidade dos componentes. Além disso, considerando que o servidor IdM comporta as funções principais do modelo (tais como registro e autenticação), foi analisado a sua escalabilidade, medindo os tempos de resposta de diversos cenários com número de clientes simultâneos diferentes.

Os testes realizados não avaliam o tempo dos processos de obtenção da credencial de proximidade, pois, além de serem diferentes para cada método (wireless e óptico) eles envolvem a interação humana. Com isso, foi desenvolvido um programa em Java (chamado de *tester*) para atuar como um cliente simulado, com o objetivo de ler um registro aleatório de um dispositivo âncora na base do IdM, gerar a credencial de proximidade (TOTP ou EID) com os parâmetros lidos e então enviar uma requisição de autenticação ao IdM.

A simulação reproduz a situação após o usuário ler a credencial com o App, por isso, o programa *tester* registra o tempo necessário para: a) enviar a requisição HTTP ao servidor com as credenciais de identidade e proximidade; b) o IdM autenticar as credenciais; e c) a resposta de retorno com o resultado da autenticação. Foram executados testes com 1, 5, 10, 15 e 20 clientes, sendo cada cliente uma *thread* executada em paralelo. Nos dois últimos casos foram utilizados dois hosts, um com 10 e outro com 5 e depois os dois com 10 *threads* simultâneas; nesses casos foram criados scripts para rodar o *tester* e sincronizados em cada máquina.

Foram utilizadas três máquinas no total todas com o sistema operacional Windows 10, uma para o servidor IdM com processador Intel i7-3770 com 3.4 GHz de *clock* e 8GB de memória RAM, e duas como clientes rodando o *tester* possuindo 16GB de RAM, processador i7-7500, 2.9 GHz de *clock* e com a funcionalidade de Hyper-Threading da Intel habilitada. Os testes foram separados em 4 baterias, cada uma com quantidade diferente de dispositivos âncoras registrados e foram utilizadas bases de dados com 1, 10, 100 e 1000 Beacons e QRCodes. Para cada bateria foram executados 5 vezes o teste para cada número de clientes e calculado a média do tempo de execução; portanto, um total de 25 execuções por bateria foram realizadas.

A Figura 25 apresenta os resultados das execuções dos testes no cenário do Beacon. O eixo y apresenta o tempo de execução em segundos, enquanto o x apresenta o número de requisições. As séries de dados mostram a quantidade de Beacons na base durante os testes. Como pode-se observar o cenário com 1000 dispositivos cadastrados foi o que levou mais tempo para a autenticação, isso porque o IdM, ao receber uma requisição para autenticar a credencial do Beacon, deve gerar o EID de cada um registrado na sua base e ir comparando com o valor recebido. Portanto, os resultados gerados podem não ser proporcionais, pois, o beacon certo pode ser, tanto o primeiro quanto o último, consultado pelo servidor.

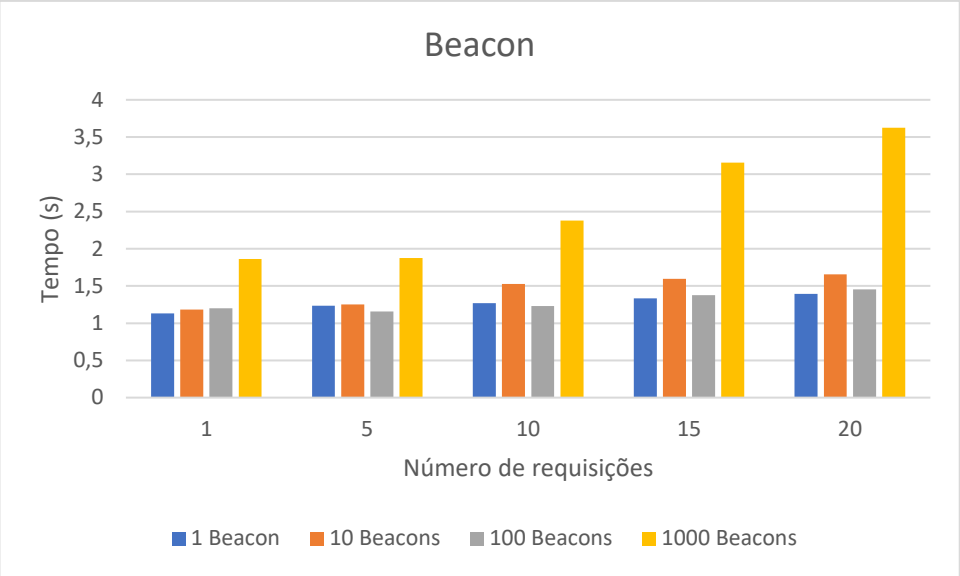

Figura 25 - Resultados dos testes com Beacon.

Por outro lado, os resultados do cenário com QRCode, exibidos na Figura 26, são um pouco mais uniformes. Como pode-se observar no gráfico, as variações entre os números de requisições simultâneas são pequenas; e em alguns casos para a mesma quantidade de clientes com bases diferentes não houve alteração no tempo. Inclusive a bateria com o maior número de dispositivos e maior número de threads teve acréscimo de apenas 12,21% em relação à base com 1 QRCode e 20 clientes. Isso é atribuído ao fato de que o IdM não precisa percorrer a sua lista de registros, pois, na requisição de autenticação deste cenário já estão os Ids do local e do dispositivo utilizado, facilitando a busca na base de dados.

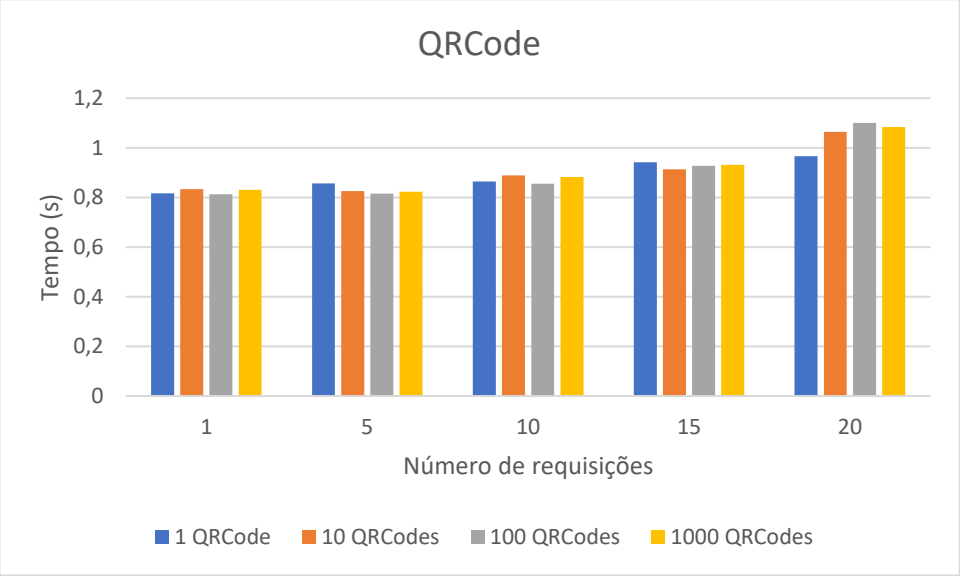

Figura 26- Resultados dos testes com QRCode.

Por fim, considerando as características de cada cenário e os resultados percebidos, pode-se afirmar que para ambientes com um maior número de usuários é recomendável o uso do QRCode. Pois, como mostra a comparação na Figura 27 há uma enorme diferença do tempo entre o Beacon e QRCode no cenário com 1000.

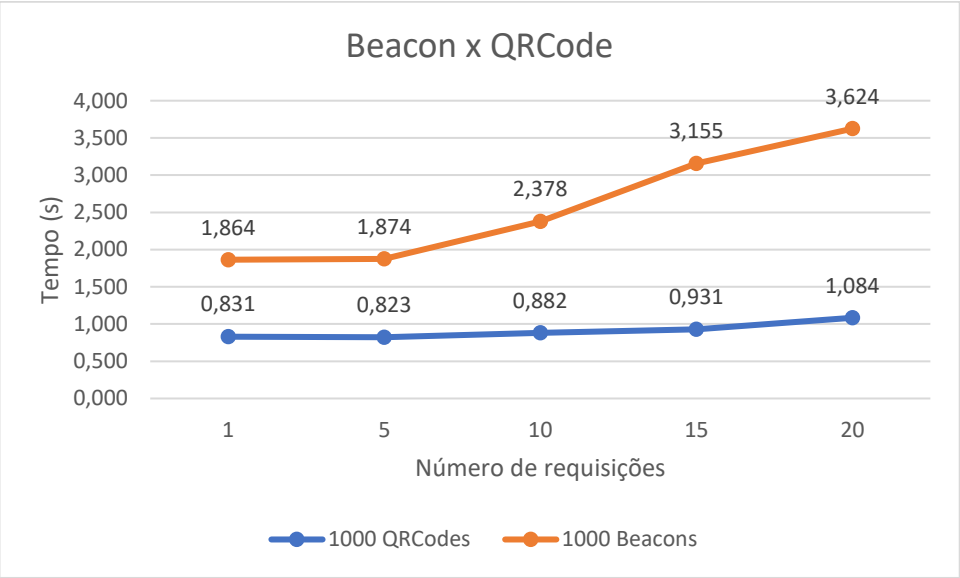

Figura 27 - Comparação dos cenários Beacon x QRCode.

Apesar de não mencionados nos testes o cenário com o wearable é representado pelos resultados do QRCode, pois a função do smartwatch é prover fonte de origem da credencial utilizando o TOTP, o que também foi feito com o programa de teste. Outro ponto a ser considerado na avaliação de performance é a fase de registro que não está presente nos testes por não ser uma operação frequente, visto que após todas as âncoras estiverem registradas não será executado o procedimento por um longo período.

Contudo, apesar de os testes realizados fornecerem resultados da viabilidade do acréscimo da credencial de proximidade em um servidor IdM, e que é possível escalar para granes usuários, ainda se faz necessária um teste com dispositivos âncoras reais distribuídos em posições geográficas diferentes. Com isso, seria possível identificar possíveis interferências, alcance do sinal wireless e a usabilidade dos usuários.

Capítulo 6

Conclusão

Este trabalho desenvolveu um modelo de autenticação multicanal e multifator baseado na proximidade do usuário, implementado diferentes técnicas para garantir que o procedimento de detecção da localização não seja facilmente imitável. Foram utilizados padrões e tecnologias de mercado, inclusive um servidor com OAuth e OpenID, para fornecer um modelo aplicável em diferentes aplicações e domínios.

A proposta apresentada utiliza dos canais ópticos, wireless e wearable com o intuito de oferecer mais segurança no caso do comprometimento de algum componente específico. O administrador do sistema pode optar por adotar um ou mais dos canais propostos. Também foram apresentadas as propostas para os dispositivos âncoras, uma vez que os usuários próximos a eles podem ser autenticados, seja lendo dados da rede Bluetooth ou um código com uma câmera. Além de forçar o usuário a estar fisicamente em alguma região, a proposta também pode aumentar a segurança nos casos onde o usuário precise mover-se constantemente. Os usuários devem informar uma credencial de identidade e uma de proximidade para obter um token de acesso com prazo de validade, e com o token requisitar os recursos protegidos. Como o servidor de autenticação segue os modelos e princípios de um IdM, é possível que diversas aplicações de domínios diferentes façam uso do token de acesso para conceder privilégios aos seus usuários.

Portanto, para validar os conceitos da proposta foi implementado um protótipo contendo componentes que representam as âncoras, servidor e aplicação final. A partir disso, foi possível executar testes no protótipo e identificar a viabilidade da proposta. Também foi possível concluir, graças a uma avaliação estática dos códigos fontes, que a implementação do modelo não é passível de nenhuma das vulnerabilidades mais conhecidas para aplicações web. Dessa forma, considerando o bom desempenho de performance nos testes realizados, conclui-se que a proposta deste trabalho é viável e promissora.

6.1. Limitações e Trabalhos Futuros

Esta seção tem como objetivo apresentar algumas limitações da proposta apresentada e sugestão de trabalhos futuros. O modelo proposto estende a capacidade de um IdM centrado no usuário na autenticação com base na proximidade. O uso de múltiplos canais na obtenção da credencial, apesar de aumentar robustez do modelo, implica em técnicas de controle de acesso físico para proteger os dispositivos âncoras, pois, caso algum atacante controle algum desses pontos ele poderá autenticar-se mesmo estando em locais não permitidos.

Uma estratégia para prevenir que clientes nos locais próximos aos dispositivos âncoras leiam e repassem a credencial de proximidade para atacantes remotos também se faz necessário. No entanto, o próprio uso de vários fatores na autenticação pode auxiliar contra esse tipo de ataque. Outra abordagem que pode ser utilizada é a de mensagens *keep alive*, onde o dispositivo cliente deve a todo momento ler a credencial de proximidade para se manter autenticado.

Como proposta de trabalhos futuros, deseja-se aplicar o protótipo desenvolvido em um cenário real, com dispositivos âncoras fixados em pontos estratégicos e o uso de uma aplicação com recursos a serem protegidos pelo modelo de autenticação. Um exemplo disso pode ser dentro de universidades, onde os alunos para utilizarem um laboratório, ou máquina específica, precisem se autenticar com sua proximidade. Outro exemplo seria para que os alunos possam utilizar a rede sem fio da instituição, nesse caso, o próprio AP poderia retornar um TOTP quando um dispositivo se conectasse a ele.

Referências

[1] The Hacker News, “Bank Servers Hacked to Trick ATMs into Spitting Out Millions in Cash”. [Online]. Available: https://thehackernews.com/2018/10/bank-atm-hacking.html. [Accessed Dec 6, 2018].

[2] S. K. S. Gupta, T. Mukherjee, K. Venkatasubramanian, and T. B. Taylor, “Proximity based access control in smart-emergency departments,” Proc. - Fourth Annu. IEEE Int. Conf. Pervasive Comput. Commun. Work. PerCom Work. 2006, vol. 2006, pp. 512–516, 2006.

[3] L. Xiao, Q. Yan, W. Lou, G. Chen, and Y. T. Hou, “Proximity-based security techniques for mobile users in wireless networks,” IEEE Trans. Inf. Forensics Secur., vol. 8, no. 12, pp. 2089–2100, 2013.

[4] D. Jaros and R. Kuchta, “New location-based authentication techniques in the access management,” Proc. - 6th Int. Conf. Wirel. Mob. Commun. ICWMC 2010, no. 1, pp. 426–430, 2010.

[5] M. Talasila, R. Curtmola, and C. Borcea, “Collaborative Bluetooth-based location authentication on smart phones,” Pervasive Mob. Comput., vol. 17, no. PA, pp. 43–62, 2015.

[6] J. Camenisch, D. A. Ortiz-yepes, and F. Preiss, “Strengthening Authentication with Privacy-Preserving Location Verification of Mobile Phones,” Wpes, pp. 37–48, 2015.

[7] W. Jansen, V. Korolev, and B. Hamilton, “Proximity-based Authentication for Mobile Devices.”

[8] D. Dasgupta, A. Roy, and A. Nag, “Toward the design of adaptive selection strategies for multi-factor authentication,” Comput. Secur., vol. 63, pp. 85–116, 2016.

[9] M. Al-fairuz and K. Renaud, “Multi-channel , Multi-level Authentication for More Secure eBanking,” Issa, 2010.

[10] M. Abadi and M. R. Tuttle, “A semantics for a logic of authentication”. ACM symposium on Principles of distributed computing, 1991.

[11] Smith, R. E, “Authentication: from passwords to public keys”, Addison-Wesley Longman Publishing Co., Inc., 2001.

[12] X. Huang, Y. Xiang, A. Chonka, J. Zhou, and R. H. Deng, “A Generic Framework for Three-FactorAuthentication: Preserving Securityand Privacy in Distributed Systems”. IEEE Transactions On Parallel And Distributed System, 2011.

[13] J. Constine, “Instagram Finally Adds Two-Factor Authentication To Fight Hackers”, TechCrunch, 2016. [Online]. Available: https://techcrunch.com/2016/02/16/instagram-two-factor/. [Accessed Jul 1, 2017].

[14] ISO, “INTERNATIONAL STANDARD ISO / IEC Information technology — Security techniques — A framework for identity,” vol. 1, 2011.

[15] Y. C. Cao and L. Y. Yang, “A survey of Identity Management technology”, International Conference on Information Theory and Information Security, 2010.

[16] UFRJ, “Gerenciamento de Identidades”, Universidade Federal do Rio de Janeiro - Escola Politécnica, Redes de Computadores, 2012. [Online]. Available: https://www.gta.ufrj.br/grad/12_1/gerenc_identidades/index.php?file=sgi. [Accessed May 21, 2017].

[17] A. Jøsang and S. Pope, “User centric identity management”, AusCERT Asia Pacific Information Technology Security Conference, 2005.

[18] Kantara Initiative, “What is UMA?”. [Online]. Available: https://kantarainitiative.org/confluence/display/uma/UMA+FAQ. [Accessed Jan 14, 2019].

[19] “What is OpenID?”. [Online]. Available: http://openid.net/what-is-openid/. [Accessed Jun 20, 2017].

[20] “Inside OpenID Connect on Force.com”. [Online]. Available: https://developer.salesforce.com/page/Inside_OpenID_Connect_on_Force.com. [Accessed May 12, 2017].

[21] OpenID, “Certified OpenID Connect Implementations”. [Online]. Available: https://openid.net/developers/certified/. [Accessed Jan 17, 2019].

[22] S. Ranger, “What is Apple iBeacon? Here’s what you need to know”, 2014. [Online]. Available: https://www.zdnet.com/article/what-is-apple-ibeacon-heres-what-you-need-to-know/. [Accessed May 15, 2017].

[23] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello and B. Schilit, “Place Lab: Device Positioning Using Radio Beacons in the Wild”, International Conference on Pervasive Computing, 2005.

[24] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, “The Cricket location-support system”, International conference on Mobile computing and networking, 2000.

[25] M. Choi, J. Lee, S. Kim, Y. S. Jeong and J. H. Park, “Location based authentication scheme using BLE for high performance digital content management system”. Neurocomputing, Volume 209, 2016.

[26] Google, “Eddystone format”. [Online]. Available: https://developers.google.com/beacons/eddystone. [Accessed Feb 6, 2018].

[27] Google, “Eddystone Ephemeral Identifier”. [Online]. Available: https://developers.google.com/beacons/eddystone-eid. [Accessed Feb 6, 2018].

[28] Google, “EID Computation”. [Online]. Available: https://github.com/google/eddystone/blob/master/eddystone-eid/eid-computation.md. [Accessed Feb 6, 2017].

[29] T. J. Soon, "QR code." Synthesis Journal 2008, 2008.

[30] K. C. Liao and W. H. Lee, “A Novel User Authentication Scheme Based on QR-Code”, Journal of Networks, 2010.

[31] S. Wang, R. Bie, F. Zhao, N. Zhang, X. Cheng and H. A. Choi, “Security in wearable communications”, IEEE Network, 2016.

[32] O. Amft and P. Lukowicz, “From Backpacks to Smartphones: Past, Present, and Future of Wearable Computers”, IEEE Pervasive Computing, 2009.

[33] M. Lee, K. Lee, J. Shim, S. J. Cho and J. Choi, “Security Threat on Wearable Services: Empirical Study using a Commercial Smartband”, International Conference on Consumer Electronics-Asia (ICCE-Asia), 2016.

[34] “Pebble Time 2”. [Online]. Available: https://www.smartwatchspecifications.com/Device/pebble-time-2/. [Accessed May 4, 2018].

[35] A. Menezes, “Elliptic Curve Public Key Cryptosystems”, Springer Science & Business Media, 2012.

[36] Entrust, “Zero To ECDH in 30 Minutes”. [Online]. Available: https://www.entrust.com/wp-content/uploads/2014/07/Zero-to-ECDH-WEB-Dec15.pdf. [Accessed Jun 4, 2018].

[37] A. Langley, M. Hamburg and S. Turner. “Elliptic curves for security,” No. RFC 7748. 2016.

[38] D. M'Raihi, S. Machani, M. Pei and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238.

[39] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226.

[40] M. L. T. Uymatiao and W. E. S. Yu, “Time-based OTP authentication via secure tunnel (TOAST): A mobile TOTP scheme using TLS seed exchange and encrypted offline keystore,” ICIST 2014 - Proc. 2014 4th IEEE Int. Conf. Inf. Sci. Technol., pp. 225–229, 2014.

[41] “OTP Token C200”. [Online]. Available: http://www.usbtoken.ro/shop_en/otp-tokens-server-authentication/c200-otptoken. [Accessed Jun 20, 2018].

[42] SignalApp, “Curve25519-Java”. [Online]. Available: https://github.com/signalapp/curve25519-java. [Accessed Jun 10, 2018].

[43] “Bouncy Castle”. [Online]. Available: https://www.bouncycastle.org/. [Accessed Jun 10, 2018].

[44] “AeroGear Security OTP Specification”. [Online]. Available: https://aerogear.org/docs/specs/aerogear-security-otp/. [Accessed Jun 20, 2018].

[45] “Android Beacon Library”. [Online]. Available: https://altbeacon.github.io/android-beacon-library/index.html. [Accessed Jul 15, 2018].

[46] “Retrofit”. [Online]. Available: https://square.github.io/retrofit/. [Accessed Feb 10, 2018].

[47] “QRCode Gen: a simple QRCode generation api for java built on top ZXING”. [Online]. Available: https://github.com/kenglxn/QRGen. [Accessed Jul 16, 2018].

[48] “Android KeyStore”, [Online]. Available: https://developer.android.com/training/articles/keystore?hl=pt-br. [Accessed Jan 20, 2018].

[49] “Hexiwear”. [Online]. Available: https://www.hexiwear.com/. [Accessed Ago 18, 2018].

[50] “Zerynth”. [Online]. Available: https://www.zerynth.com/. [Accessed Jul 21, 2018].

[51] “Pil Image”. [Online]. Available: https://pillow.readthedocs.io/en/3.1.x/reference/Image.html. [Accessed Ago 20, 2018].

[52] “PyQRCode”. [Online]. Available: https://pypi.org/project/PyQRCode/. [Accessed Ago 5, 2018].

[53] “PyOTP”. [Online]. Available: https://github.com/pyauth/pyotp. [Accessed Ago 6, 2018].

[54] “EID Tools”. [Online]. Available: https://github.com/google/eddystone/blob/master/eddystone-eid/tools/eidtools.py. [Accessed Apr 15, 2018].

[55] Motorola, “Moto 360”. [Online]. Available: https://www.motorola.com.au/products/moto-360-sport. [Accessed Out 2, 2018].

[56] “ZXing Android Embedded”. [Online]. Available: https://github.com/journeyapps/zxing-android-embedded. [Accessed Jul 18, 2018].

[57] E. J. Steven and G. Peterson, “Introduction to Identity Management Risk Metrics.”

[58] “SonarQube”. [Online]. Available: https://www.sonarqube.org/. [Accessed Sep 21, 2018].

[59] SonarQube, “Security-related rules”. [Online]. Available: https://docs.sonarqube.org/display/SONARqube71/Security-related+rules. [Accessed Sep 21, 2018].

[60] CWE, “CWE VIEW: Weaknesses in Software Written in Java”. [Online]. Available: http://cwe.mitre.org/data/definitions/660.html. [Accessed Sep 28, 2018].

[61] SANS, “What Errors Are Included in the Top 25 Software Errors?”. [Online]. Available: https://www.sans.org/top25-software-errors. [Accessed Sep 28, 2018].

[62] OWASP, “OWASP Top 10 Application Security Risks - 2017”. [Online]. Available: https://www.owasp.org/index.php/Top_10-2017_Top_10. [Accessed Sep 28, 2018].

[63] “Find Security Bugs”. [Online]. Available: http://find-sec-bugs.github.io/. [Accessed Sep 30, 2018].