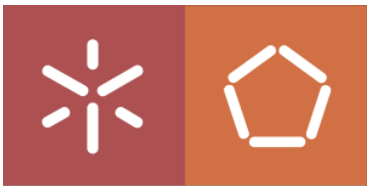

Universidade do Minho
Escola de Engenharia

Universidade do Minho
Mestrado Integrado em Engenharia Informática

SSI

Segurança de Sistemas Informáticos
Trabalho Prático 1

Outubro 2020

Tânia Filipa Amorim Rocha
A85176

Maria Miguel Albuquerque Regueiras
A85242

Conteúdo

Lista de Figuras

1 Introdução

No âmbito da cadeira de Segurança de Sistemas Informáticos, foi proposta a realização de um relatório que identificasse e detalhasse a que riscos um sistema pode estar exposto. Assim, será abordado primeiro o funcionamento do sistema assim como uma descrição dos seus componentes e consequentes interações.

De seguida, são apresentados os objetivos que se querem atingir a nível de segurança da informação e infraestrutura. São apresentados, por fim, as vulnerabilidades e ameaças que se encontraram com respetivas soluções (nos casos em que tal era possível).

Ultimamente, concluí-se com uma análise geral e recomendações por parte do grupo.

2 O Sistema

O sistema *mID* tem como objetivo a possível substituição dos documentos pessoais de identificação dos cidadãos por uma versão digital equivalente num *smartphone* com várias funcionalidades, como, por exemplo, a limitação (por parte do utilizador) de que informações pretende ceder a entidades verificadoras que necessitem de informação sobre o mesmo. Para além disso, existe uma entidade emissora que serve de fonte de autenticação dos dados.

Para isso, nesta secção aborda-se pontos essenciais do sistema para posterior análise de vulnerabilidades e ameaças. É ainda apresentado um diagrama visual do fluxo de dados para melhor compreensão de possíveis pontos fragilizados do sistema.

2.1 Aplicação *mID*

A aplicação *mID* é a componente que é diretamente utilizada pelo cidadão que pretende usufruir do serviço, suportada em Android e IOS. A sua função principal é armazenar os dados de um documento de identificação e os elementos usados pela aplicação *leitor*. O seu funcionamento pode ser resumido da seguinte forma:

- A primeira vez que é utilizada, a aplicação conecta-se com a entidade emissora dos documentos através de uma **conexão TCP/IP**;
- O cidadão **autentica-se**;
- É iniciado o **download** dos dados relativos aos documentos no formato **JSON**;
- Esta transferência é **realizada periodicamente** para atualização dos dados (sem recorrer a uma autenticação prévia).

Existem ainda interações entre esta entidade e os dispositivos com a aplicação Leitora, ambas iniciadas pelo portador e desencadeadas pela leitura de um **QR Code**. Estas podem ser:

- **Offline**: o dispositivo do portador transfere os atributos de identificação diretamente para o dispositivo leitor, assim como os dados necessários para a sua verificação;
- **Online**: o dispositivo leitor transfere um *token* de autorização para que o verificador consulte diretamente a entidade emissora do documento, obtendo dados mais recentes sobre o cidadão.

Adicionalmente, estas realizam-se através do estabelecimento de um canal entre os dispositivos e o envio de um pedido, por parte do verificador, solicitando os atributos necessários através dos seus identificadores. O portador pode decidir se quer transferir os dados na totalidade ou apenas um subconjunto deles. As mensagens entre eles são codificadas no formato **CBOR** - Concise Binary Object Representation. Estas interações podem utilizar uma das seguintes tecnologias:

- **BLE** - Bluetooth Low Energy;
- **NFC** - Near Field Communication;
- **WiFi-Aware**.

2.2 Aplicação Leitora

A aplicação Leitora é utilizada pelos verificadores, pessoas que pretendem verificar informações sobre cidadãos. É também uma aplicação móvel para Android e IOS. Como mencionado anteriormente, este interage com a aplicação *mID* para obter dados de um cidadão que sejam precisos para a sua identificação.

Esta necessita de **suportar os protocolos** indicados anteriormente. É também o verificador que decide se a interação será em modo offline ou online. Também possui mecanismos de **autenticação** do verificador ou da própria aplicação.

Adicionalmente, é preciso que esta aplicação consiga fornecer os dados necessários para permitir a auditoria das interações e operações por parte do portador.

2.3 Entidade Emissora

A entidade emissora é a entidade responsável por:

- Emitir documentos de identificação;
- Conferir-lhes autenticidade;
- Fornecer mecanismos para garantir autenticidade e integridade aos documentos transmitidos tanto no modo online como offline.

Esta entidade, tal como as outras, interage com ambas via **rede pública** recorrendo, mais uma vez, a conexões **TCP/IP**. O estado atual ds infra-estrutura desta entidade tem os seguintes componentes/tecnologias:

- Sistema operativo do servidor web: CentOS 7.8.2003 ;
- Backend principal: Django v3.0 ;
- Servidor web: UWSGI ;
- Base de dados principal: PostgreSQL 12.4 ;
- Sistema operativo do serviço de gestão do sistema: Ubuntu 20.04 ;
- Backend de gestão: Flask 1.0 ;
- Servidor web: Gunicorn ;
- Base de dados de gestão: PostgreSQL 12.1 (a correr num container Docker versão 19.03.6).

2.4 Diagrama de Fluxo de Dados

Para auxiliar a visão das interacções e comunicações que existem entre as entidades do sistema, foi realizado um Diagrama de Fluxo de Dados (DFD), apresentado na figura 1, que as representa de forma objetiva assim como o tipo de dados que circulam no sistema. Este é de grande importância para analisar potenciais ameaças a nível das comunicações (referidas mais à frente).

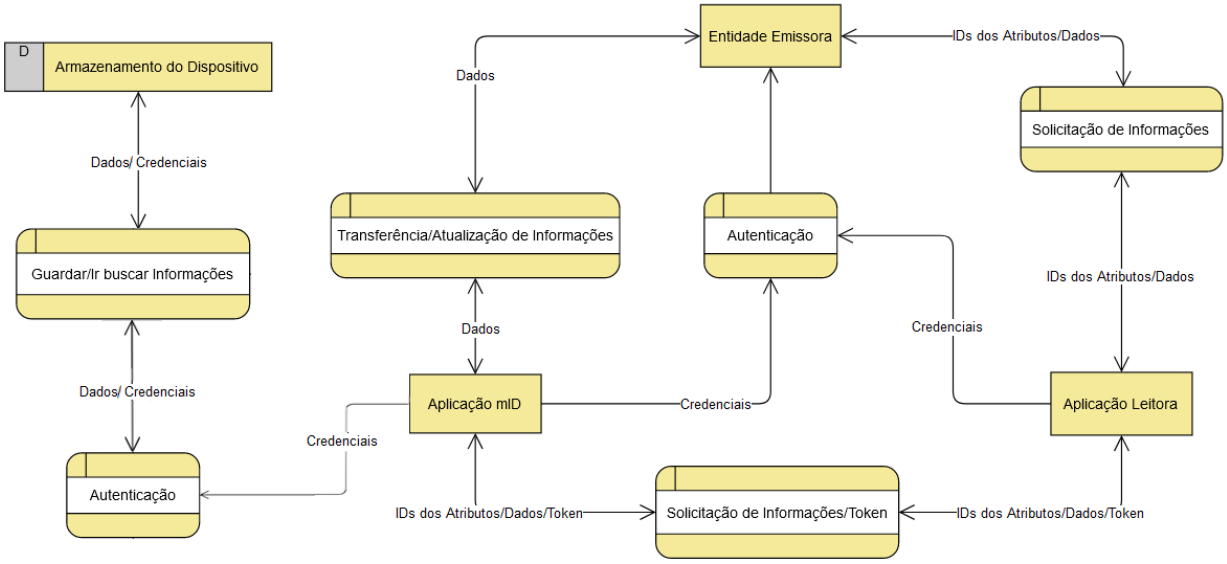

Figura 1: Diagrama de Fluxo de Dados do Sistema

3 Objectivos

Sendo o objectivo principal deste relatório apresentar de forma clara que ameaças e vulnerabilidades o sistema apresenta, é preciso uma análise que parte da necessidade de garantir segurança não só ao sistema, mas também ao funcionamento deste.

De seguida apresentam-se as características que se pretendem garantir:

- **Confidencialidade**: garantir que apenas os utilizadores do sistema (tanto cidadãos como entidades verificadoras) com os privilégios adequados têm acesso aos dados ou os alteram.
- **Integridade**: garante a autenticidade dos dados mantendo a consistência, precisão e fidelidade dos dados.
- **Disponibilidade**: garante que os serviços oferecidos pelo sistema de mantém disponíveis sem qualquer obstáculo ou interferência.

Neste caso de estudo, é necessário garantir que:

- Os mecanismos de autenticação das aplicações possuem confidencialidade e integridade dos dados transferidos entre dispositivos e entidades;
- Os próprios dados armazenados no dispositivo portador apresentam integridade e autenticidade;
- As interacções entre o dispositivo portador , dispositivo leitor e a entidade emissora são confidenciais;
- A entidade emissora, apesar de não intervir nas operações em modo offline, está sempre disponível assim como os serviços que esta oferece.

Com estes objetivos em mente e uma vez com o funcionamento do sistema compreendido, é possível agora passar à descrição do que pode estar exposto a ataques, assim como discernir qual o impacto que cada possível ameaça terá e o risco que isso constitui (qual a probabilidade de tal acontecer). Podem ainda ser apresentadas algumas sugestões de soluções.

4 Catálogo de Vulnerabilidades

A pesquisa de vulnerabilidade é baseada nos vários componentes do sistema descritos no enunciado. As vulnerabilidades apresentadas de seguida têm na sua generalidade um nível de severidade entre alto e critico.

- **Sistema operativo do servidor web: CentOS 7.8.2003**
  - CVE-2020-0543 - A limpeza incompleta de operações específicas de leitura de registo especial em alguns processadores Intel (R) pode permitir que um usuário autenticado habilite a divulgação de informações via acesso local.
- **Backend principal: Django v3.0**
  - CVE-2020-9402 - Django 1.11 até 1.11.29, 2.2 até 2.2.11 e 3.0 até 3.0.4 permite injecções SQL se dados não confiáveis forem usados como um parâmetro de tolerância em funções GIS e agregados no Oracle. Ao passar uma tolerância adequadamente elaborada para funções GIS e agregados no Oracle, é possível interromper o escape e injectar SQL malicioso.
  - CVE-2020-13596 - Um problema foi descoberto no Django 2.2 até 2.2.13 e 3.0 até 3.0.7. Os parâmetros de queries gerados pelo administrador do Django, ForeignKeyRawIdWidget não são codificados pela URL correctamente, levando à possibilidade de um ataque XSS.
  - CVE-2020-13254 - Um problema foi descoberto no Django 2.2 até 2.2.13 e 3.0 até 3.0.7. Nos casos em que o back-end do cacheMemory não executa a validação da chave, a passagem de chaves de cache mal formadas pode resultar em uma colisão de chaves e potencial exposição de dados.
- **Servidor web: UWSGI**

- CVE-2018-7490 - uWSGI até 2.0.17 manipula e verifica erradamente o DOCUMENT_ROOT durante as opções –php-docroot, permitindo travessias entre directorias.
- CVE-2020-11984 - Cópia do buffer sem verificar o tamanho do input ('Classic Buffer Overflow')
- CVE-2018-6758 - Out-of-bounds Write - o uwsgi_expand_path function no core/utils.c em Unbit uWSGI até 2.0.15 sofre de stack-based buffer overflow devido a grandes dimensões de directorias.

- **Base de dados principal: PostgreSQL 12.4 e Base de dados de gestão: PostgreSQL 12.1**
  - CVE-2020-14349 - Verificou-se que as versões do PostgreSQL anteriores a 12.4, antes de 11.9 e antes de 10.14 não limpam adequadamente o search_path durante a replicação lógica. Um atacante autenticado pode usar essa falha em um ataque semelhante ao CVE-2018-1058 (a seguir explicada), a fim de executar um comando SQL arbitrário no contexto do usuário usado para replicação.
  - CVE-2018-1058 - O Postgresql permitia que um utilizador modificasse o comportamento de uma consulta para outros utilizadores. Um atacante com uma conta de utilizador pode usar essa falha para executar código com as permissões de super-utilizador na base de dados. As versões 9.3 a 10 são afectadas.
  - CVE-2020-14350 - Algumas extensões do PostgreSQL não usam o search_path com segurança no script de instalação. Um atacante com privilégios suficientes pode usar esta falha para induzir um administrador a executar um script especialmente criado durante a instalação ou actualização de tal extensão. Vulnerabilidade presente nas versões do PostgreSQL antes de 12.4, antes de 11.9, antes de 10.14, antes de 9.6.19 e antes de 9.5.23.
- **Sistema operativo do serviço de gestão do sistema: Ubuntu 20.04**
  - CVE-2020-15708 - Package desta versão do Ubuntu cria sockets de controlo com permissões globais de escrita e leitura. Um atacante pode usar esta situação para rescrever ficheiros ou executar código arbitrário.
- **Backend de gestão: Flask 1.0**
  - CVE-2019-1010083 - O Pallets Project Flask anterior a 1.0 é afetado por: uso inesperado de memória. O impacto é: denial of service. O vetor de ataque é: dados JSON codificados criados.
- **Servidor web: Gunicorn**
  - CVE-2018-1000164 - Gunicorn versão 19.4.5 contém uma vulnerabilidade CWE-113: Neutralização imprópria de sequências CRLF em cabeçalhos HTTP na função "process_headers"em "gunicorn / http / wsgi.py"que pode resultar num atacante a fazer com que o servidor retorne cabeçalhos HTTP arbitrários. Esta vulnerabilidade parece ter sido corrigida em 19.5.0.
- **Docker 19.03.6**
  - CVE-2020-13401 - Um problema foi descoberto no Docker Engine versões até 19.03.11. Um atacante em um container, com o recurso CAP_NET_RAW, pode criar anúncios de router IPv6 e, consequentemente, falsificar(spoofing) hosts IPv6 externos, obter informações confidenciais ou causar uma negação de serviço.

5 Modelo de Ameaças

O modelo de ameaças ou mais conhecido por *Threat Model* é um processo que reconhece ameaças potencias tais como vulnerabilidades de estrutura ou de segurança e enumera-as. O objectivo desta modelação é fornecer aos defensores os possíveis atacantes e ataques ideais ao sistema através da análise às vulnerabilidades e fraquezas do mesmo. Este tipo de modelação visa encontrar os problemas de segurança antecipadamente para os analisar, corrigir e posteriormente elevar a qualidade e segurança do produto.

Existem várias estratégias para a modelação de ameaças mas a que será implementada neste trabalho será uma análise estruturada que se foca nos ataques e atacantes possíveis do sistema. Este tipo estratégia adopta uma visão no sentido de "o que é que um atacante quererá do meu sistema?", o que nos permite enumerar vários tópicos com dados a manter seguros.

Através da abstracção das ameaças é possível organiza-las através de um sistema de classes nomeado de *STRIDE* o que permite uma melhor análise de riscos e *exploits* comuns usados por atacantes.

STRIDE é um acrónimo de várias situações:

- *Spoofing*: utilizar a identidade de algo ou alguém, ou seja, personificar um sistema ou uma pessoa, violando a autenticação correta.
- *Tampering*: modificar dados num disco, rede ou memoria violando a integridade de dados.
- *Repudiation*: o acto de recusar a confirmação de que algo ocorre, por exemplo, eliminar *logs* do sistema.
- *Information Disclosure*: fornecer informação a uma entidade que não está autorizada a ter acesso à mesma comprometendo confidencialidade.
- *Denial of Service*: esgotar os recursos necessários ao funcionamento do sistema, violando assim a disponibilidade do sistema.
- *Elevation of Privilege*: permitir que uma entidade tenha acesso a funcionalidades ou recursos aos quais não devia estar autorizado violando autorização no sistema.

Neste sentido, será feita uma análise através deste modelo à aplicação em questão. Serão analisadas as principais componentes da aplicação do portador, leitor, da emissora e ainda do tipo de comunicações utilizadas entre as mesmas.

5.1 Aplicação do portador

Tal como na maioria das aplicações mobile IOS e android é assumido que esta aplicação tem também uma arquitectura servidor-cliente. O cliente encontra-se geralmente no sistema operativo. Este cliente é transferido para o dispositivo através das aplicações de plataformas distribuídas onde os desenvolvedores publicam os seus projectos. A outra componente ou seja, o servidor, localiza-se como host dos desenvolvedores. A maioria das aplicações funcionam com este sistema e já trazem mecanismos de segurança previamente instalados. Por default, um mecanismo de defesa é que qualquer aplicação instalada consegue apenas aceder a ficheiros nas suas próprias directorias na *sandbox* e impede que estas informações sejam indevidamente alteradas.

No entanto, erros da parte dos desenvolvedores no desenvolvimento da aplicação podem causar falhas na segurança que podem ser usadas como meio de ataque.

Numa primeira análise à parte portadora da aplicação é possível identificar diferentes tipos de ameaças e classifica-las de acordo com o modelo de ameaças.

- **Spoofing** - Visto que a aplicação do portador é algo que um utilizador carrega consigo é possível que exista roubo do dispositivo ou mesmo um ataque informático com o objectivo de um atacante tentar usar a identidade do utilizador para efectuar transacções e troca de dados. Este método de ataque viola a autenticação correta na aplicação e para que o mesmo não ocorra são necessário meios que permitam uma autenticação segura que garanta que a aplicação não esteja a ser utilizada por uma entidade estranha.

  Existem vários métodos de autenticação que visam a segurança neste contexto no entanto é necessário definir um conceito importante para este tipo de controlo:

  – Identidade Digital - Identidade digital corresponde à informação única, acerca de um agente que está envolvido numa qualquer transacção online;

Nesta situação é considerado um processo de autenticação denominado por *Authentication Assurance Levels* ou seja, AAL. Este processo tem 3 níveis que definem o método de autenticação necessário para obter acesso a uma aplicação. Quanto mais alto o nível, maior a exigência da informação presente. Os 3 níveis por ordem de sensibilidade são:

- **Nivel 1 - Passwords** - Passwords devem ser armazenadas de forma segura, e devem também poder ser alteradas. Para reforçar a segurança fornecida, estas devem cumprir alguns requisitos de modo a oferecerem maior resistência a ataques:
  - Possuir, pelo menos, 8 caracteres se existir Multi-Factor Authentication (MFA), caso contrário, este número deve ser aumentado para, pelo menos, 10 caracteres;
  - Garantir que as passwords utilizadas não são passwords comuns, isto é, que já tenham sido comprometidas. Para tal, bloquear as 1000/10000 passwords mais comuns que cumpram os requisitos de comprimento e que se encontrem em listas de passwords já comprometidas.
  - Optar por adotar MFA ou passwords mais longas;

  No que toca à recuperação de passwords, este processo deve utilizar, sempre que possível, elementos de MFA. Por exemplo perguntas de segurança ou envio de tokens para um dispositivo que pertença ao utilizador. Para além disso, as passwords armazenadas pelo sistema, devem estar guardadas de forma segura, existindo controlos criptográficos.
- **Nivel 2 - Autenticação Multi-Factor (MFA)** - Geralmente, este nível de autenticação é utilizado em aplicações que contenham PII (personal identifiable information) disponível online ou seja, o exemplo da aplicação em questão.

  MFA assegura a identidade de um utilizador, ao pedir que o mesmo se autentique utilizando uma combinação de *password* e PIN , *token* e telemóvel ou dados biométricos.

  Ao pedir que a autenticação seja feita desta forma, o sistema torna-se mais robusto, pois o atacante precisa de ganhar acesso a mais do que um elemento de um utilizador para se fazer passar pelo próprio.
- **Nível 3: Autenticação criptográfica** - Este nível de segurança é utilizado quando um sistema comprometido resulta em danos pessoais, financeiros ou do interesse públicos. A autenticação utilizada nestes casos, é baseada na prova de posse de uma chave através de protocolos criptográficos. Geralmente, é implementado usando módulos de hardware criptográfico, como por exemplo, o cartão de cidadão.

Um dos possíveis métodos contra este tipo de ameaça é a limitação do número de tentativas de autenticação que deram erro.

Após uma análise dos níveis de autenticação é considerava a situação da autenticação. Se for uma primeira autenticação em que é necessário fazer download dos dados da entidade emissora deve ser utilizado o nível 3 devido à importância dos dados e comprovar a identidade ainda com informações pessoais. No caso de inicialização da aplicação após a primeira vez em que é necessário utilizar os dados anteriormente obtidos deve ser utilizada preferencialmente um método de autenticação de nível 2 na aplicação para ter acesso aos dados armazenados no dispositivo.

- **Tampering** - Neste componente da aplicação total não há qualquer forma de adulterar dados na entidade emissora, no entanto, pode ser possível que haja uma tentativa de modificação de dados ao nível do armazenamento do dispositivo.

Este tipo de ameaça é uma preocupação recorrente visto que pode alterar a aplicação em si ou o seu comportamento. Por exemplo, uma aplicação pode não permitir por segurança efectuar alguma tarefa e através de tampering pode-se obrigar a aplicação a faze-lo. Existem diferentes tecnicas tampering em aplicações mobile nomeadamente :

- Binary Patching - que retrata a alteração da compilação da aplicação através da alteração dos executáveis. Geralmente para um atacante exterior este tipo de técnica é difícil de levar avante, no entanto é relativamente fácil para utilizador o fazer.

- Injecção de Código - É uma técnica poderosa que permite explorar e alterar os processos na sua execução. Esta técnica pode ser implementada de várias maneiras e permitem acesso directo para processar a memoria e objectos da aplicação. Neste caso é ainda mais difícil detectar a situação de *tampering*.

Em ambas as situações permite alterar as permissões ou comportamento da aplicação e assim alterar algum dos dados que o utilizador possua. No entanto, ao efectuar troca de dados com a entidade leitora, esta irá pedir confirmação à entidade emissora, que não permitirá a troca caso os dados não estejam correctos. No caso de uma entidade de leitura não requisitar confirmação de dados, por exemplo, ser apenas necessário comprovar a identidade a alguém poderá então estar a ser levado a cabo um crime de falsificação de identidade.

- **Repudiation** - Visto que a nível da aplicação do portador deve haver registo de acontecimentos, esta ameaça representa a possibilidade de partes deste histórico serem apagadas e negar situações que possam ter ocorrido. É importante ter então camadas de protecção contra qualquer manipulação destes dados que não seja ler.
- **Information Disclosure** - Este tipo de ameaça pode surgir do próprio utilizador apesar de muito pouco provável, visto que pode fornecer os seus dados pessoais a quem desejar mesmo a entidades que não deva ter acesso às suas informações, no entanto, como é do interesse do utilizador proteger os seus dados não é uma situação a ter em conta.

Por outro lado, se os dados pessoais de um determinado utilizador forem "roubados"estes passam a estar na posse de alguém que não devia ter acesso. No entanto, este tipo de ataque seria apenas possível se um atacante conseguisse ter acesso às informações pessoais armazenadas no dispositivo pessoal da vitima e para isso seria necessário recorrer primeiramente a um ataque do tipo *spoofing* ou *App-Spoofing/SDK-spoofing*. Esta ultima ameaça consiste na falsificação , de uma aplicação semelhante, onde o utilizador efectua o login. Esta aplicação falsificada pode então enviar as informações introduzidas da autenticação para o atacante responsável pela mesma.

Uma maneira de combater estas ameaças parte do próprio utilizador. Este não deve permitir a instalação ou transferência de aplicações *third-party*.

É também de elevada importância que os dispositivos possuam técnica de protecção do armazenamento de dados visto que uma falha a este nível tem impactos técnicos, de negócio, pessoais etc, devido à informação exposta. Geralmente dispositivos podem estar vulneráveis a ataques de roubo de informação através de XML data stores, binary data stories, cookie stores, cartão SD, cloud sincronizada e até mesmo ao nível do sistema operativo, frameworks etc. Maneira de tentar prevenir armazenamento de informação inseguro é verificar como é tratado cashing de URLs, do teclado, clipboard, logs, aplicações em backgorund etc.

Devido à existência de histórico e logs é necessário existir uma implementação correta do mesmo para que não causem brechas de segurança. Este tópico abrange também métodos de proteção contra ameaças do tipo *repudiation*.

- Implementação de um sistemas de logs - A implementação de um sistema de loggs, para além de ser uma ferramenta muito útil para debugging, se for correctamente implementada, permite servir como uma barreira de defesa para identificar actividade suspeita no sistema, por exemplo através da utilização de um IDS, aumentando assim a segurança da aplicação. Apesar de útil, se um sistema deste tipo não for correctamente implementado, pode servir como uma fonte para atacantes obterem informações sobre o sistema, por isso é necessário ter em conta certos factores aquando da construção de um sistema de loggs:
  - Não se deve armazenar informação sensível nos logs, tais como passwords;
  - Não se deve armazenar informação desnecessária nos logs, apenas o essencial para identificação de potenciais ataques;
  - Os formatos dos logs das várias partes do sistema devem ser consistentes entre si, e aquando da sua sincronização é necessário ter em atenção

    * Os logs de sistemas distribuídos devem ser encaminhados para um sistema central, de modo a permitir uma monitorização central e resiliência no caso da perda de um nodo.

É também importante existir um controlo de possíveis erros de código constantemente e das suas correcções. A ocorrência de erros é algo normal durante a utilização de um software, por isso é crucial que os mesmos sejam tratados de forma correta através do uso de excepções. Aquando do tratamento de erros, é preciso ter alguns factores em consideração de modo a não comprometer a segurança do sistema, nem a revelar informação crítica acerca da aplicação. Assim sendo, algumas recomendações para o correto tratamento de erros são as seguintes:

- Tratar as excepções de forma centralizada de modo a evitar duplicação de código;
- Assegurar que existem excepções capazes de 'apanhar' qualquer comportamento inesperado no sistema;
- Fazer logging das excepções com informação suficiente do erro ocorrido, de modo a permitir aos desenvolvedores perceberem o que se passou de errado no sistema;
- Após a detecção de um erro, corrigir o mesmo;
- Testar o código que trata de 'apanhar' erros, para garantir que funciona da maneira esperada

- **Denial of Service** - Geralmente, este tipo de ameaça em aplicações mobile surgem do download de aplicações que tem por objectivo efectuar um ataque propositado a uma rede ou aplicação. Existe também, maioritariamente em Android, possibilidade de ocorrer esgotamento de recursos do dispositivo que incapacitem a aplicação de funcionar como deveria, isto porque, ao contrário do sistema operativo IOS, a Android permite partilha de recursos entre aplicações que estejam em funcionamento ou em background. Um exemplo disso é a possibilidade de ter duas aplicações em concorrência no ecrã o que pode gerar race conditions em que um mesmo recurso esteja a ser necessário por duas ou mais aplicações. Uma forma de resolver esta situação seria mesmo ao nível de desenvolvimento do sistema operativo em que seria necessário remover todos os blocos de código que permitissem race conditions.
- *Elevation of Privilege* - Em algumas versões Android existe uma vulnerabilidade que permite elevation of privilege por atacantes exteriores. Esta vulnerabilidade existe porque o software erradamente validade pedidos de conexão. No entanto isto pode ser evitado se o dispositivo for actualizado.

Uma outra razão para que esta ameaça exista, é devido a um determinado tipo de *tampering* que altera o o comportamento da aplicação nomeadamente permitir acções não possíveis anteriormente. Esta situação é considerada elevation of privilege.

5.2 Aplicação leitora

A nivel da aplicação leitora que pode ser Android ou IOS as ameaças tem algumas semelhanças às presentes na aplicação portadora.

É necessário existir autenticações seguras na aplicação leitora devido à sua importância visto que ataque de interesse maligno seria ter acesso ao controlo desta aplicação para permitir e autorizar transacções a utilizadores que não deviriam.

- **Spoofing** - Caso ocorra um erro ou fraude a nível da autenticação que permita um atacante tomar controlo da aplicação passando-se por alguém que pode aceitar ou recusar a verificação de dados de um utilizador. É possível também que, do mesmo modo que na aplicação do portador, que exista *appSpoofing* o que leve erradamente ao utilizador enviar dados privados a uma entidade exterior. É por isso necessário e importante que exista um método de autenticação sensível como por exemplo AAL nível 2 ou nível 3.
- **Tampering** - É importante salientar esta ameaça no contexto da aplicação do leitor visto que a aplicação permite confirmar os dados de um cidadão tanto offline como online. Quando ocorre online, a emissora confirma os dados e por isso não há risco tão elevado, no entanto quando é feita uma acção offline, a aplicação basea-se nos dados que o dispositivo tem armazenado. Estes dados quando offline podem ser alterados e manipulados e posteriormente efectuar verificações

erradamente e por isso devem existir níveis de segurança que não permitam a adulteração destes dados. Também é necessário estes níveis de protecção quando existe actividade online visto que a informação do portador nunca deve ser alterada ante de ser confirmada na emissora para que não ocorram respostas incorrectas.

A nível da aplicação, esta pode ser alterada em casos semelhantes à aplicação do portador. é então necessário mecanismos anti-tampering como por exemplo *Android Root detection* ou *iOS Jailbreak Detection* que detecta se uma aplicação esta a ser executada ou iniciada de uma root correta visto que tipicamente aplicações que sofreram *tampering* executam de um ambiente root ou jailbroken. O jailbreak é uma alteração no sistema que permite ao usuário entrar na raiz do equipamento, tendo acesso a todos os arquivos do sistema operativo.

- **Repudiation** - É previsto que a aplicação leitora também tenha um sistema de históricos das acções aceites ou não. Esta informação deve estar registada e por isso é possível que ocorra um ataque com objectivo a apagar logs ou partes do histórico o que pode ter consequências graves de anulamento por exemplo de multas ou crimes etc. É necessário que exista camadas de protecção que não permitam a alteração ou eliminação destes dados e que este sistemas esteja desenvolvido da melhor forma como explicado anteriormente na aplicação do portador.
- **Information Disclosure** - Semelhante às outras ameaças, a aplicação do leitor tem os mesmos riscos que a aplicação do portador onde a informação guardada a nível de armazenamento do dispositivo pode vir a ser exposta. Esta informação pode conter históricos de verificações e dados pessoais de levada importância que não devem ser conhecidos por entidades exteriores daí que seja necessário que existam mecanismos de segurança que protejam o armazenamento do dispositivo. Como dito anteriormente, alguns sistemas operativos são mais expostos a esta ameaça de que outros devido à estruturação destes. Tem que ser garantido espaços de armazenamento para dados relativos à aplicação que não possam ser de maneira alguma partilhados com outras aplicações.

  Outros métodos de protecção contra este tipo de ameaças é evitar guardar credenciais de acesso, obrigando o utilizador a autenticar-se sempre, considerar métodos de encriptação de informação utilizar métodos de segurança através da biblioteca *javax.crypto* quando se trata de armazenamento em cartões SD entre outros.

  Tal como falado na aplicação anterior é também necessário desenvolver a estrutura de histórico e logs de maneira correta e corrigir qualquer erro de código que surja conforme o uso da aplicação.
- **Denial of Service** - Da mesma forma que a aplicação do portador pode sofrer falta de recursos, a aplicação do leitor pode sofrer o mesmo, devido à constante utilização de recursos no dispositivo que pode causar race conditions em alguns casos. As razões disto acontecer são as mesmas que na aplicação do portador.
- **Elevation of Priviliege** - Novamente, esta ameaça pode ocorrer pelas mesmas causas que na aplicação do portador. É necessário ainda verificar todos os dados recebidos do portador porque podem contém códigos que afectem a aplicação do leitor alterando lhe as funcionalidades como por exemplo aceitar ou recusar serviços.

5.3 Entidade Emissora

A entidade emissora é composta por vários componentes de desenvolvimento web, bases de dados, etc. Por isso a analise feita em stride é baseada em pesquisa sobre o mesmo.

- **Spoofing**

  No caso da parte da emissora que é acessível a partir da web existem várias fragilidades que possam permitir a falsificação da identidade de alguém para poder atacar o sistema. Um dos ataques possíveis é através de *webSpoofing* que se caracteriza por páginas web falsas com objectivo de roubar as informações inseridas por um utilizador na altura da sua autenticação. Deve ser sempre verificado a autenticidade da página em que o utilizador se encontra.

Outros comportamentos que levam ao risco de fraude e roubo de identidade é através de sessões guardadas e cookies. O tratamento destas situações deve ser desenvolvida em conjunto com autenticações seguras que neste contexto devem ser de nível 3 devido à importância dos dados no sistema.

- Sessões guardadas - O estado de autenticação do utilizador, é mantido numa sessão guardada num servidor. Um utilizador recebe um ID de sessão, para que saiba que sessão server-side possui a sua informação. O utilizador apenas precisa deste ID, o que implica que toda a informação sensível está do lado do servidor, o que é o recomendado.

  De seguida, encontram-se algumas boas práticas para implementar manutenção de sessões:

  * IDs de sessão devem ser longos, únicos e aleatórios;
  * Deve ser gerado um novo ID de sessão durante a autenticação e reautenticação;
  * Após um certo período de inactividade, a sessão deve expirar e o utilizador deve-se autenticar de novo. O intervalo de tempo utilizado deve ser inversamente proporcional ao valor da informação protegida.

- Bookies do Browser - Cookies são métodos utilizado por muitas aplicações web, para armazenar IDs de sessão. No entanto, aquando da sua utilização, devem ser implementadas as seguintes defesas:

  * Quando utilizadas para manter IDs de sessão, as cookies devem expirar em simultâneo com a sessão, ou logo de seguida;
  * As flags "secure" e "HttpOnly" devem estar activas, de forma a garantir que uma transferência é feita apenas através de uma conexão segura (TLS), e para impedir que as cookies sejam acedidas via JavaScript;
  * Deve ser adicionado o atributo "samesite" para impedir que browsers modernos enviem cookies com cross-site requests, o que ajuda a evitar cross-site request forgery e fugas de informação.

- **Tampering**

Visto que esta aplicação na parte da emissora tem componentes web serão analisados os métodos desta ameaça neste contexto.

O ataque de *tampering* Web é baseado na manipulação de parâmetros trocados entre cliente e servidor para modificar os dados da aplicação, como credenciais e permissões do usuário, preço e quantidade de produtos, etc. Normalmente, essas informações são armazenadas em cookies, de forma oculta campos ou URL Query Strings e são usadas para aumentar a funcionalidade e o controle do aplicativo.

Esse ataque pode ser executado por um usuário mal-intencionado que deseja explorar o aplicativo para seu próprio benefício ou por um invasor que deseja atacar uma terceira pessoa usando um ataque man-in-the-middle. Em ambos os casos, ferramentas como Webscarab e proxy Paros são mais usadas.

O sucesso do ataque depende da integridade e dos erros do mecanismo de validação lógica, e a sua exploração pode resultar em outras consequências, incluindo XSS (cross site scripting), injeções SQL, inclusão de arquivo e ataques de divulgação de caminho.

É necessário evitar injecções de código nas bases de dados através de um sistema de autenticação e controlo de acesso. Algumas boas práticas para implementar um acesso seguro a bases de dados pelo software. passam por:

- Queries Seguras - Para evitar ataques, como injecções de SQL, aquando da execução de queries com a base de dados é preciso implementar parametrização das queries, que consiste na pré-compilação de declarações SQL que apenas necessitam de parâmetros, impedindo assim que

sejam executadas queries maliciosas. No entanto, certas partes das bases de dados podem não permitir esta parametrização, sendo necessário fazer outro tipo de controlo de input de modo a evitar ataques de injecção SQL.

- Comunicação e autenticação segura - Durante a execução de queries, estas e os seus resultados devem ser sempre encriptados, de modo a não revelarem informações do sistema. Além disso, todos os acessos à base de dados devem ser autenticados através de canais seguros. Secure Configuration Por fim, antes da utilização de uma qualquer base de dados, devem ser activados os controlos de segurança das mesmas, configurando-os de modo a proteger os dados do sistema.

Alguns exemplos de Web Parameter Tampering que foi descrito são:

- Modificação de parâmetros em campos de formulários.
- Quando uma aplicação web usa campos ocultos para armazenar informações de status, um utilizador mal-intencionado pode adulterar os valores armazenados em seu navegador e alterar as informações referidas.
- Um atacante pode adulterar os parâmetros de URL directamente.

- **Repudiation**

Web services na sua generalidade predispõe de contramedidadas e tecnologia contra este tipo de ameaças como por exemplo XML signatures.

- **Information Disclosure**

Na sua maior parte o risco de Information disclosure é maior no lado do cliente. Um atacante pode ler todos os activos que residem na máquina cliente ou em trânsito através do canal HTTP. Isso leva às seguintes ameaças que são considerado mais relevante nesta categoria:

- As mensagens são divulgadas, possivelmente expondo informações especificas como números de cartão de cidadão.
- Os arquivos WSDL são divulgados desnecessariamente, fornecendo ao atacante informações sobre a estrutura da aplicação.
- outros.

Esta categoria de ameaças serve também para detectar quais as medidas necessárias para manter a informação de um sistema segura. Tendo em conta que nem todos os dados de um sistema possuem o mesmo grau de importância, é necessário criar uma espécie de classificação para os dados consoante a sua importância, de modo a ser possível mapear quais as medidas de segurança que serão necessárias implementar para proteger os diversos dados do sistema tendo em conta o seu grau de importância. Aquando da necessidade da transmissão de dados, estas devem possuir encriptação end-to-end, de modo a que estejam protegidos no caso se serem interceptados por atacantes activos. Se o sistema necessitar de armazenar dados localmente, tal como no caso das aplicações do portador e leitor, esses devem ser guardados de forma encriptada, de modo a que seja possível evitar que a mesma seja revelada ou modificada sem autorização. É também necessário garantir que as chaves utilizadas para proteger os dados são armazenadas de forma correta e que todos os certificados/chaves sejam renovados periodicamente, de modo a garantir uma maior segurança para os dados.

Devido aos pedidos de informações que existem dos outros dois tipos de aplicações, há sempre uma possibilidade de ataque com objectivo de roubar informação, daí que seja necessário técnicas de protecção e verificação dos pedidos das entidades exteriores. Este tipo de técnicas denominam-se por técnicas de controlo de acesso. Controlo de Acesso consiste num processo cujo objectivo é conceder/- negar pedidos feitos por utilizadores, programas ou processos. É também responsável por gerir os privilégios de acesso ao sistema por parte dos vários utilizadores do mesmo. Este tipo de controlo é importante, pois é indesejável permitir acesso indiscriminado à totalidade do sistema por parte de qualquer entidade. Assim sendo, antes da implementação de um protocolo de controlo de acessos, é necessário definir qual a política que o mesmo irá implementar. De seguida estão apresentadas as políticas de controlo de acesso mais populares:

- Discretionary Access Control (DAC) - Nesta política, é o dono do objecto que decide quem tem acesso ao mesmo;
- Mandatory Access Control (MAC) - Nesta política, os recursos aos quais se pretendem aceder possuem um determinado nível de autorização. Para se aceder a um dado recurso, é necessário possuir um nível de autorização igual ou superior ao do recurso;
- Role Based Access Control (RBAC) - Nesta política, o acesso a um recurso é determinado pela role de quem tenta aceder ao mesmo possui no sistema;
- Attribute Based Action Control (ABAC) - Por fim, esta política garante acesso aos recursos consoante atributos dos utilizadores, como por exemplo, ser necessário possuir mais de 18 anos para aceder ao recurso.

Depois de definida a política de acesso, é necessário implementar o protocolo de controlo de acesso, tendo em conta os seguintes passos:

- Os controlos de acesso devem ser planeados na sua totalidade antes de começar o processo de desenvolvimento pois, como estes tendem a evoluir para algo complexo, torna-se difícil fazer alterações após a implementação;
- Todos os pedidos devem passar por algum tipo de controlo de acesso;
- Deve ser standard negar todos os pedidos de acesso, a não ser que estes sejam especificamente permitidos;
- Dar o menor acesso possível a todos os utilizadores, programas e processos;
- Não definir as roles dos utilizadores de forma hardcoded, pois além de tornar a manutenção do sistema de controlo de acesso difícil e trabalhosa, pode levar a acessos indevidos ao sistema;
- Todos os eventos do sistema de controlo de acesso devem ser registados em logs, pois tentativas falhadas podem ser um sinal de probing. malicioso.

- **Denial of Service**

Esta é uma das ameaças mais relevantes neste contexto devido à estrutura da aplicação da emissora.

Existem muitas maneiras de tornar um serviço indisponível para utilizadores legítimos, manipulação de pacotes de rede, vulnerabilidades de programação, lógica ou gerência de recursos, entre outros. Se um serviço receber um número muito grande de solicitações, ele pode deixar de estar disponível para os utilizadores. Da mesma forma, um serviço pode parar se uma vulnerabilidade de programação for explorada, ou a maneira como o serviço gere os recursos falha.

O atacante pode injectar e executar código arbitrário enquanto executa um ataque DoS para aceder a informações críticas ou executar comandos no servidor. Ataques de negação de serviço destroem significativamente a qualidade do serviço experimentada por utilizadores. Esses ataques introduzem grandes atrasos de resposta, perdas excessivas e interrupções de serviço, resultando em impacto direto da disponibilidade do mesmo.

- Fatores de risco: Os fatores de risco podem ser divididos em várias categorias. Duas fontes principais de risco incluem recursos inadequados e motivadores de ameaças não técnicas.

  O primeiro exemplo de fator de risco, recursos inadequados, requer atenção se a arquitetura do sistema não foi projetada para atender a picos de demanda de tráfego. Esse risco reduz a probabilidade de execução bem-sucedida aquando um ataque DoS e pode, se não for atendido, resultar em DoS.

  O segundo exemplo e talvez o maior fator de risco, não é técnico e está no domínio das relações públicas ou comunicações estratégicas. Convém que uma organização evite tomar medidas que possam torná-la alvo de um ataque DoS, a menos que os benefícios de fazer isso superem os custos potenciais ou os controles de atenuação estejam em vigor.

Outros fatores de risco também podem existir, dependendo do ambiente específico.

- **Elevation of Priviliege**

  Tendo em conta os constituintes desta aplicação existem vários que podem levar a esta ameaça.

  Um dos softwares utilizados chama-se uWSGI e é um servidor de aplicação web, que implementa protocolos como WSGI/uwsgi/http, e oferece suporte para vários idiomas através de plugins. O uWSGI permite configurar aplicações web back-end de forma dinâmica através de variáveis protocolo uwsgi. Se a porta uWSGI for exposta, os invasores podem construir pacotes uwsgi e especificar a variável mágica UWSGI_FILE para executar comandos arbitrários usando o protocolo exec:/. Foi confirmado que a porta 8000 do uWSGI está acessível ao público. Esta situação permite a entidades exteriores não autorizadas passarem a ter acesso ao sistema.

  Um método para tentar remediar a situação é a porta uWSGI não ser acessível publicamente. uWSGI deve ser configurado para ouvir apenas na interface local (127.0.0.1).

  Um outro método de alterar permissões e privilégios á através de input com hacks já falado anteriormente na aplicação do leitor.

  No caso de a aplicação ser do tipo Cliente-Servidor, a validação de input deve ser sempre feita do lado do servidor por motivos de segurança, ou seja, na emissora, pois a validação do lado do Cliente pode ser facilmente ultrapassada, pois este não possui as ferramentas necessárias para validar correctamente o input.

  Validação de inputs é uma técnica utilizada com o intuito de garantir que todo o input de um programa esteja formatado correctamente. No entanto, é importante referir que mesmo validando o input utilizando as técnicas apresentadas em baixo, não significa que o mesmo não possa ser perigoso para a integridade do sistema. Existem dois tipos de validação que devem ser verificados a cada input, sendo eles:

  - Validade Sintáctica - Valida se os dados introduzidos estão na forma esperada, isto é, se for pedido dados do cartão de cidadão, verifica se o input recebido se trata informações correspondentes. Existem duas abordagens para fazer este tipo de validação, sendo elas:
    - *Whitelisting* - Verifica se o input cumpre um dado conjunto de regras 'boas' pretendidas;
    - *Blacklisting* - Verifica se existe algum tipo de conteúdo malicioso no input. Esta é uma técnica onde é difícil cobrir todos os possíveis casos, mas ajuda a evitar ataques mais óbvios, por exemplo, procurando ¡SCRIPT¿ no input.
  - Validade Semântica - Aceita apenas inputs que façam sentido no contexto da aplicação, por exemplo, uma data inicial deve ocorrer antes de uma data final.

  Outra forma de evitar estas situações de injeçoes é através do uso de expressões regulares. As expressões regulares , podem ser utilizadas para validar se um input cumpre certos padrões, mas a utilização de ER pode ser de difícil compreensão para certos programadores, o que pode dificultar o seu uso e manutenção.

5.4 Comunicações

Nesta análise decidiu-se incluir uma secção dedicada apenas às comunicações e interações que são realizadas entre entidades do sistema pois estas podem estar expostas a várias ameaças que podem comprometer o seu funcionamento normal. É necessário garantir que os canais criados são seguros e que sejam o mais resistentes possível a ataques que podem comprometer tanto a integridade como a disponibilidade e a confidencialidade do sistema.

5.4.1 Comunicações TCP-IP

As comunicações TCP/IP, em particular com o protocolo TCP (*Transmission Control Protocol*), demonstram ser um transporte fiável de dados fim-a-fim orientado à conexão. Isto significa que são criados canais para a comunicação entre duas entidades sendo oferecidos ainda mecanismos de controlo de erros, fluxo e de congestão. A base deste protocolo passa por iniciar a conexão com um 3-way handshake, a transferência de dados em si e o término da conexão como representado de forma simples na figura 3. Contudo, apenas a existência deste protocolo como está não é a melhor abordagem ao problema uma vez que, em termos de segurança, pouco é realizado para a garantir.

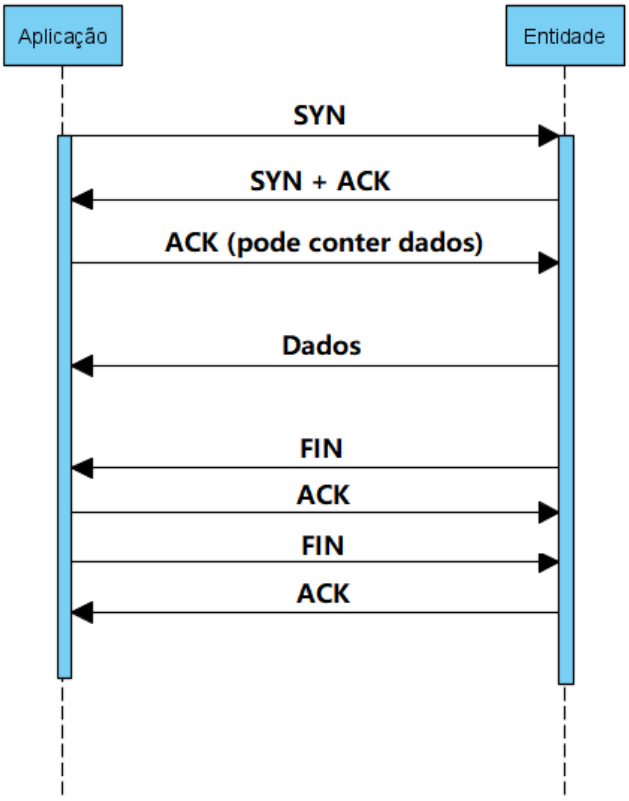

Figura 2: Representação do funcionamento do protocolo TCP

É de grande importância sublinhar o facto destas comunicações serem feitas em **redes públicas** pois é conhecido que grande parte de ataques são feitos neste contexto. No sistema, este tipo de comunicações ocorrem quando há transferência de dados entre a entidade emissora e a aplicação mID ou a aplicação leitora. Isso representa uma miríade de potenciais ameaças onde está em causa a documentação, os dados de autenticação do utilizador e o próprio funcionamento do sistema em si.

É comum observar-se casos de **hijacking**, onde alguém pode "tomar conta"da sessão em causa. Este tipo de ataques podem ter muitas naturezas (enquadram-se tanto na categoria de Spoofing como de Tampering e até de Denial of Service) dependendo do objetivo do atacante. O objetivo será sempre criar um estado em que o cliente e o servidor não possam trocar dados, permitindo-lhe fabricar pacotes aceitáveis para ambos, imitando os pacotes reais. Assim, o atacante consegue obter o controlo da sessão.

Analisando o tipo de ameaças que podem existir com estas características em conta:

- **Spoofing**: é relevante nas comunicações, uma vez orientadas à conexão, existe uma origem e um destino e neste caso podem tentar fazer-se passar por tanto um como o outro. Isto torna-se grave pois o utilizador pode ter as suas informações em causa. Um ataque comum neste protocolo é o **ARP Cache Poisoning** que envolve associar um IP a um endereço MAC incorreto. Como o ARP é um protocolo sem estado, a cache fica com conhecimento dos pedidos de saída para potenciais novas entradas. Isso significa que um atacante pode preparar e transmitir intencionalmente um

pacote ARP com uma mensagem específica. Além disso, não há protocolo de autenticação com ARP, portanto, todas as entradas recebidas são tratadas como aceitáveis.

É possível ainda acontecer Spoofing associado ao Denial of Service e ao Tampering, por exemplo num ataque como **TCP Reset** que tira partido da forma como no TCP uma conexão é terminada. Basta um atacante fingir ser a aplicação em causa e enviar um pacote RST (para dizer que a conexão vai ser reinicializada) para o destino, iniciando o término da conexão.

- **Tampering**: é muito possível acontecer interceção de pacotes e até a adulteração destes. O ataque **TCP Reset** mencionado anteriormente pode ser usado nesse sentido. É de extrema importância que os dados transferidos não sejam de forma alguma alterados por terceiros externos à comunicação em causa, e acima de tudo, sem autorização do utilizador ou da entidade. Este tipo de ataques são conhecidos como ataques **Man-in-the-Middle** e permitem que alguém externo consiga intervir na comunicação fingindo ser tanto a origem e o destino, enganando ambos, introduzindo um mecanismo potente para adulteração de dados.

  Desta forma pode ser possível adulterar tanto credenciais como dados em si. Se os dados do utilizador forem alterados podem existir inúmeros problemas e tal pode ser uma arma útil para quem queira causá-los. A título de exemplo, mudar a identificação de alguém pode impossibilita-la a realizar atividades quotidianas e criar descrepâncias entre varias entidades que necessitem destas informações (Finanças, Bancos, serviços no geral). Tal pode tornar-se perigoso.

- **Repudiation**: uma comunicação onde não existe nenhum tipo de cifragem ou de assinatura digital, é propícia a existir repúdio da informação e dos pedidos a nível dos pacotes enviados.

- **Information Disclosure**: este ponto torna-se dos mais importantes nesta componente do sistema, uma vez que todos os dados estão a ser transferidos entre a entidade emissora e as aplicações através deste protocolo. Adicionalmente, nas especificações do sistema é mencionado que os dados trocados são no formato JSON, contudo, não é mencionado se estes são cifrados. Qualquer tipo de interceção destas comunicações podem levar à revelação de informação privada sem o consentimento do utilizador e da própria entidade emissora. Nos dias que correm, é muito usual ocorrerem estas situações onde informação é exposta a terceiros por ataques do tipo "Man-in-the-Middle"e recorrendo a técnicas de **Sniffing**.

  Sniffing é um processo de monitorização onde se capturam pacotes de dados que passam por uma determinada rede. Desta forma, consegue-se ver as informações que estão nos pacotes capturados e até desviá-los. Estes são comuns de acontecer sobretudo em redes pouco fiáveis que é o caso das redes públicas, como mencionado anteriormente. Assim um atacante pode facilmente ver que dados e credenciais estão a circular na rede e roubá-los.

- **Denial of Service**: existem inúmeras ameaças deste tipo que podem ocorrer e no sistema em geral e é nestas comunicações que podem ser mais notáveis. Alguns dos ataques mais conhecidos passam por atingir a conexão e tentar congestioná-lo de forma a interromper a disponibilidade da conexão e do serviço ou mesmo desviar o próprio encaminhamento dos pacotes.

  Um dos ataques mais relevantes neste momento é o **SYN Flood**, que explora o *3-way handshake* mencionado anteriormente de forma a ocupar os recursos disponíveis (neste caso, o serviço oferecido pela entidade emissora) e tornando o sistema indisponível. Nestes, o atacante envia pacotes SYN repetidos para todas as portas do destino, geralmente usando um endereço IP falso. O destino, inconsciente do que se sucede, recebe vários pedidos (aparentemente legítimos) para estabelecer uma conexão e tenta responder a todos. Nenhum dos IPs que enviou os SYN envia um ACK de volta, sendo que o destino fica à espera da confirmação, e enquanto existe este tempo em que a conexão não é fechada é quando novos SYN são enviados, entupindo a conexão e sobrecarregando o destino.

  Adicionalmente, o **ARP Cache Poisoning** pode ser também usado como um ataque de DoS. Existem também de ataques que tomam partido do buffer que existe nestas comunicações do lado do destino e tentar enviar pacotes com tamanhos superiores tentando interromper o serviço (tal como o conhecido "Ping of Death").

- **Elevation of Priviliege**: neste caso, este tipos de ataques tornam-se menos relevantes.

Para estas ameaças existem um conjunto de precauções que podem (e devem) ser tomadas a fim de proteger melhor o sistema e a informação que nele circula. A implementação de uma camada de proteção **TLS** (Transport Layer Security) resolve a maior parte dos problemas mencionados, garantindo assim:

- **Cifragem dos dados**: É de extrema importância que os dados sejam cifrados para que não viagem na rede expostos a todos os perigos mencionados anteriormente. Tal pode ser feito através de métodos de cifragem de chave pública onde existe uma chave pública conhecida por todos e uma chave privada que apenas o recetor tem, para decifrar a mensagem. No TLS, no handshake inicial são partilhadas as chaves.Este método, além de ser um passo para garantir a integridade dos dados, ajuda ainda para garantir o não repúdio das informações. No entanto, existem outras formas mais eficientes para tal como incluir uma assinatura digital que garante integridade, autenticação do originador e o não repúdio do originador.
- **Autenticação**: o TLS oferece ainda mecanismos de autenticação para evitar situações de spoofing, baseada em mecanismos de confiança (ambos confiam um no outro quando há troca de chaves) e certificados.
- **Integridade**: por fim, o TLS oferece ainda mecanismos de message framing que introduz em cada mensagem um código de autenticação (MAC). O algoritmo MAC age como um checksum através de uma função hash criptográfica, cujas chaves são negociadas por ambos os pontos de conexão. Sempre que um registro TLS é enviado, um valor MAC é gerado e anexado a essa mensagem, e o receptor é então capaz de calcular e verificar o valor MAC enviado para garantir a integridade e autenticidade da mensagem.

Desta forma, as comunicações entre a entidade emissora e as aplicações podem ser realizadas de forma mais segura.

5.4.2 Comunicações Wifi-Aware

Relativamente às comunicações entre os dispositivos em si (entre o utilizador com a aplicação mID e a aplicação leitora) existem 3 tecnologias possíveis através das quais estas se podem realizar, como mencionado anteriormente. No entanto, ataques as estes tipos de tecnologias podem ser mais difíceis e improváveis de acontecerem, e por isso, apresentam um risco menor nalguns casos. Os tipos de ameaças são muito parecidos nas três, por isso serão descritas a detalhe as mais relevantes no tipo de tecnologia que esteja a ser abordado. A primeira que se aborda neste relatório é o Wifi-Aware.

Segundo a organização, o Wifi-Aware "estende a capacidade do Wi-Fi com descoberta rápida, conexão e troca de dados com outros dispositivos Wi-Fi - sem a necessidade de uma infraestrutura de rede tradicional, conexão à Internet ou sinal GPS (...) estabelecendo conexões Wi-Fi ponto a ponto independentes com base na localização imediata e nas preferências do usuário"(1). Este funciona criando clusters com dispositivos na sua vizinhança e a aplicação não tem controlo sobre o comportamento do cluster.

Esta arquitetura é um tipo de Neighbor Awareness Networking (NAN) composta com duas partes principais:

- *Discovery Engine* (DE): responsável por oferecer mecanismos à aplicação para Publish/Subscribe (anunciar e subscrever).
- *Medium Access Control* (MAC): responsável pela manutenção dos clusters, pela sincronização e por fornecer serviços de transmissão e recepção ao DE.

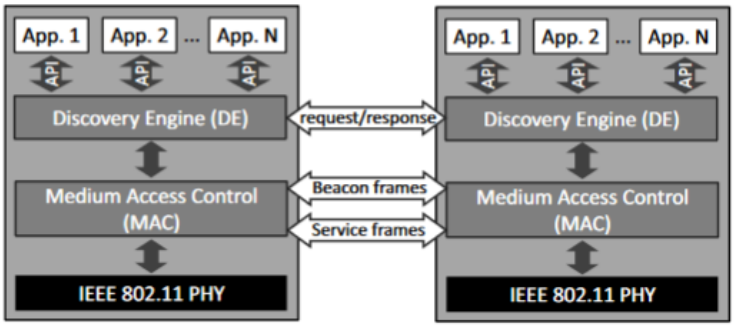

Figura 3: Representação da stack de uma arquitetura NAN(2).

O seu funcionamento passa primeiramente por descobrir outros NAN clusters através de comunicações em canais com bandas pré-definidas. Para isso, cada dispositivo envia um *Beacon Frame* em Broadcast periodicamente e está passivamente a procurar outros frames do género de outros dispositivos.
Para além disso, é necessário manterem-se sincronizados e para isso enviam Beacons de sincronização entre si.

Para esta tecnologia existem algumas ameaças ainda que à primeira vista pareça bastante robusto, isto pois, segundo a descrição desta, é mencionado que a segurança tem de ser assegurada a nível aplicacional:

- **Spoofing**: como em qualquer tipo de comunicação, este mecanismo é sustetível a ataques em que se queiram fazer passar por um dos intervenientes da conexão.
- **Tampering**: pode ocorrer durante o envio de Beacons de sincronização onde o atacante pode enviar Beacons forjados e incorretos de forma a interromper o processo de sincronização ou inserir informações falsas sobre os cluster neste fluxo. No entanto, o risco deste caso é menor uma vez que por anunciarem vários dispositivos em broadcast, o sistema torna-se robusto o suficiente para nem sempre deixar que isso aconteça.
- **Repudiation**: é possível também este tipo de ataques, ainda que não sejam tão relevantes.
- **Information Disclosure**: os processos de Publish e Subscribe incluem identificadores de serviço. Estes são resultados de aplicar uma função hash a um nome de serviço legível por humanos. Estes identificadores por estarem cifrados não são compreensíveis a terceiros e dificilmente se consegue rastrear a aplicação que o originou. É possível ainda formar identificadores de grupos únicos que são criados através da junção de uma chave partilhada no grupo com o nome do serviço. No entanto, a definição desta chave e consequente proteção do grupo não está ao nível das configurações das especificações da arquitetura em si. Estas são geridas a nível aplicacional e é importante manter a confidencialidade da informação para evitar ataques com o objetivo de furto desta.
- **Denial of Service**: apesar de a ameaça mencionada no ponto de Tampering ter um menor risco, este tipo de ataque pode ser usado com o objetivo de entupir a rede com Beacons deste tipo e afetar a disponibilidade do serviço.
- **Elevation of Priviliege**: não se torna relevante nesta componente.

Para prevenir estes tipos de ataques é recomendado que a nível aplicacional haja um bom planeamento de chaves assim como um bom estudo do que é requerido da aplicação para uma boa implementação da tecnologia.

5.4.3 Comunicações *Bluetooth Low Energy*

BLE, ou *Bluetooth Low Energy*, é um mecanismo que usa ondas rádio para comunicações short-range. É possível criar e configurar um dispositivo BLE que consiga transmitir dados até 30 metros ou mais numa linha de visão desempedida. No entanto, é usual que sejam usados apenas 2 a 5 metros de raio. Este mecanismo necessita de uma boa compreensão do seu funcionamento, uma vez que grande parte da segurança que pode ter tem de ser planeada a nível aplicacional.

Nesta tecnologia, existe primeiro uma fase de emparelhamento entre dispositivos e o posterior estabelecimento e verificação da conexão. Uma vez verificada, é praticamente impossível atacá-la com sucesso, segundo os developers. No entanto, a fase de emparelhamento é crucial e pode ser um ponto fraco no sistema.

O emparelhamento passa por dois (ou três) passos:

1. Partilha de dados de autenticação e de identificação. Estes identificam-se na rede especificando que tipo de dispositivo são, que input/output processam, entre outros. Esta partilha não é cifrada;
2. Criação e partilha das chaves;
3. Se decidirem conectarem-se, guardam os dados de autenticação partilhados para eventuais futuros emparelhamentos com o mesmo dispositivo.

Assim, torna-se evidente que o segundo passo do emparelhamento se torna o mais propício a ataques (e o mais relevante). Analisando as possíveis ameaças:

- **Spoofing**: como em qualquer tipo de comunicação, este mecanismo é sustetível a ataques em que se queiram fazer passar por um dos intervenientes da conexão.
- **Tampering**: ataques do tipo man-in-the-middle são possíveis de acontecer e intercetar os dados que estão a ser transmitidos. É uma janela aberta para inserir dados inválidos ou falso na comunicação.
- **Repudiation**: é possível também este tipo de ataques, ainda que não sejam tão relevantes.
- **Information Disclosure**: tal como nas tecnologias anteriores, este mecanismo pode sofrer de ataques de eavesdropping passivo, acrescendo o facto de este não precisar de se encontrar tão perto do alvo como no NCF e Wifi-Aware. Pode ser possível também que tentem aceder aos dados que foram guardados na fase 3 do processo, revelando informações sobre vários dispositivos que se conectaram anteriormente.
- **Denial of Service**: relativamente a ataques deste tipo, é também possível que através dos ataques de man-in-the-middle sejam introduzidos pacotes excessivos para interromper a troca de informações.
- **Elevation of Priviliege**: não se torna tão relevante.

Existem várias formas de evitar algumas das ameaças mencionadas. Primeiro, é aconselhado usar BLE Secure Connections (módulos 4.2+) pois estes oferecem proteção contra eavesdropping passivo e têm ainda métodos de pairing seguros (uma chave ou por comparação numérica) para proteger contra ataques man-in-the-middle. Exite também um pairing OOB (Out of Band) para evitar transmitir dados importantes por BLE.

Os próprios criadores desta tecnologia investiram na segurança da segunda fase do processo por esta ser tão crítica e para isso existem dois tipos de conexões BLE:

- *Legacy connections*: podem ser implementadas para as versões 4.0, 4.1 e 4.2 do BLE. Os dispositivos trocam uma chave temporária (Temporary Key) e usam-na para criar uma chave outra chave temporária (Short Term Key) a qual pode ser usada para autorizar a conexão. Estas ligações não são as mais seguras mas podem vir a ser com o método de pairing mais adequado.
- *Secure connections*: esta foi introduzida com a versão 4.2 do BLE e não é compatível com versões mais antigas. Estas implementam o algoritmo de Diffle-Hellman de curva elítica para a criação das chaves e tem um processo de autenticação de chaves mais complexo. Este introduz proteção contra eavesdropping passivo e permite que o dispositivo estja seguro mesmo com um método de pairing adequado.

Existem vários métodos de pairing adequados, no entanto cada um tem os seus prós e contras dependendo do objetivo da ligação que se quer proteger. Assim é recomendado o estudo destes e a adaptação de um deles ao sistema em causa.

5.4.4 Comunicações por NFC

NFC, ou *Near Field Communication*, basea-se na transferência de informação via ondas rádio. Os dispositivos têm de adotar certas especificações para comunicar entre si. A tecnologia usada no NFC passa pela indução eletromagnética a fim de transmitir a informação. Existem alguns modos em que o NFC é usado, tal como emular um cartão de crédito para realizar pagamentos. O mais comum em smartphones, e o que possivelmente poderá ser usado no sistema que estamos a estudar, é o modo peer-to-peer, em que os dispositivos se encontram ativos ao enviar informação e passivos ao recebê-la. Adicionalmente, o NFC tem um raio de cerca de 10cm onde é possível conectar-se com outros dispositivos.

Ainda que pareça uma tecnologia segura por ser short-range, como todos os casos, tem os seus pontos fracos:

- **Spoofing**: como em qualquer tipo de comunicação, este mecanismo é sustetível a ataques em que se queiram fazer passar por um dos intervenientes da conexão.
- **Tampering**: é possível que o atacante queira alterar as informações transmitidas e manipulá-las de alguma forma. O modo mais exposto a este tipo de ataque é quando o dispositivo se encontra em modo ativo, mas consegue ser contornado se a informação for transmitida a uma taxa de 106 Baud, tornando impossível a alteração de todos os dados a serem transmitidos.
- **Repudiation**: podem acontecer, ainda que não tão relevantes.
- **Information Disclosure**: uma vez que o NFC usa ondas rádio para comunicar, estas propagam-se para a sua vizinhança e não apenas para o raio definido. Ainda que pareça difícil alguém receber esse sinal fora do raio estipulado, é estimado que um atacante consiga receber o sinal até 1 metro de distância enquanto os dispositivos se encontram no modo passivo e até 10 metros quando estão em modo ativo.

  Isto pode levar a que o atacante consiga receber partes do sinal ou até a sua integridade e consequentemente a informação transmitida, dando origem a um ataque chama-se **eavesdroping**. Podem utilizar ainda antenas específicas para tal.
- **Denial of Service**: em vez de o atacante "escutar"os sinais e tentar intercetá-los, este nem precisa de os decifrar e pode apenas tentar quebrar a conexão, enviando dados inválidos ou entupindo o canal. No entanto, os dispositivos NFC conseguem detetar este tipo de ataques pois a energia necessária para o atacante realizar o ataque é muito superior a uma transmissão normal de dados via NFC.
- **Elevation of Priviliege**: não se torna relevante neste caso.

Existe também o risco de algum dispositivo ser aproximado de uma hardware com NFC ativado o que poderá ativar uma transação sem o portador se dar conta.

Uma solução possível seria criar um canal de comunicação seguro. Tal irá proteger contra eavesdropping e ataques que comprometam a integridade e confidencialidade dos dados. Pode ser usado um protocolo de criptografia de chave pública e tentar minimizar ataques do género man-in-the-middle.

5.5 Resumo

Como forma de resumo do que foi dito, na tabela seguinte apresentam-se os principais tipos de ataques a que cada componente / interação do sistema pode estar exposto.

	Aplicação mID	Aplicação Leitora	Entidade Emissora	TCP/IP	Wifi-Aware	BLE	NFC
Spoofing	X	X	X	X			
Tampering	X	X	X	X	X	X	X
Repudiation	X	X		X			
Information Disclosure	X	X	X	X	X	X	X
Denial of Service	X	X	X	X	X	X	X
Elevation of Privilege	X	X	X				

Figura 4: Resumo

6 Conclusão

Neste relatório abordou-se de forma detalhada que possíveis ameaças e vulnerabilidades o sistema mID pode apresentar. Passando pela descrição do sistema, entendeu-se como cada componente deste funcionava assim como estas comunicavam entre si e como os dados se movimentavam. Depois, definiu-se que objetivos queremos atingir a nível de segurança concluindo que era necessário maner a confidencialidade, a integridade e a autenticidade do sistema.

É de extrema importância ter as vulnerabilidades acima evidenciadas em consideração e adaptar o desenvolvimento das componentes que faltam. Desenvolver uma aplicação sabendo que existem vulnerabilidades que poderiam ser evitadas não demonstra um bom planeamento pela equipa responsável.

Após as ameaças descritas, é de notar que umas têm mais risco associado que outras (com grande impacto e grande probabilidade de acontecerem). Estas serão a maioria das ameaças relacionadas com as comunicações referidas, assim como as descritas com mais detalhe em cada entidade. O detalhe destas ameaças foram já realizadas tendo em consideração quais as que teriam mais risco, por isso uma leitura deste documento já deixa a equipa de desenvolvimento com uma ideia geral do que poderá correr mal.

Com este relatório, concluímos com um apelo à investigação das soluções propostas e eventuais modificações que podem ser realizadas de forma a tornar o sistema mais seguro e fiável.

7 Referências

https://www.makeuseof.com/tag/nfc-security-contactless-payment-issues/

https://money.cnn.com/2012/07/26/technology/nfc-hack/index.htm

https://www.simform.com/iot-bluetooth-security-vulnerabilities/#vul

https://resources.infosecinstitute.com/topic/near-field-communication-nfc-technology-vulnerabilities-and-principal-attack-schema/

https://owasp.org/www-community/attacks/Form_action_hijacking

https://www.ptsecurity.com/ww-en/analytics/mobile-application-security-threats-and-vulnerabilities-2019/

https://www.adjust.com/glossary/sdk-spoofing/

https://mobile-security.gitbook.io/mobile-security-testing-guide/general-mobile-app-testing-guide/0x04c-tampering-and-reverse-engineering

https://support.honeywellaidc.com/s/article/Android-Privilege-Elevation-Vulnerability

https://owasp.org/www-project-mobile-top-10/2016-risks/m2-insecure-data-storage

https://owasp.org/www-project-mobile-top-10/2016-risks/m8-code-tampering

https://owasp.org/www-project-mobile-top-10/2016-risks/m4-insecure-authentication

https://owasp.org/www-project-mobile-top-10/2014-risks/m2-insecure-data-storage

https://techbeacon.com/security/5-essential-steps-securing-enterprise-mobile-apps

https://owasp.org/www-project-mobile-top-10/2014-risks/m8-security-decisions-via-untrusted-inputs

https://www.cvedetails.com/cve/CVE-2018-6758/

https://www.acunetix.com/vulnerabilities/web/uwsgi-unauthorized-access-vulnerability/

https://owasp.org/www-project-proactive-controls/

https://owasp.org/www-project-top-ten/

https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

https://owasp.org/www-community/attacks/Web_Parameter_Tampering

https://link.springer.com/content/pdf/10.1007

https://owasp.org/www-community/attacks/Denial_of_Service

https://www.wi-fi.org/discover-wi-fi/wi-fi-aware

https://upcommons.upc.edu/bitstream/handle/2117/85888/NAN_overview.pdf

https://lembergsolutions.com/blog/how-secure-ble-communication-standard