Projeto DESOFS

Mestrado em Engenharia Informática - Desenvolvimento de Software Segurança

Beatriz Borges - 1201461
Davide Clemente - 1190497
Sara Borges - 1191053
Simão Gomes - 1191061

Março 2024

Conteúdo

1 Introdução

O presente documento foi desenvolvido no âmbito do projeto da cadeira de Desenvolvimento de Sofware Seguro, cujo objetivo é desenvolver uma aplicação seguindo o processo de **SDLC** (***Secure Software Development Life Cycle***. Este é um processo bastante importante para o desenvolvimento de *software*, uma vez que, contemplando todas as suas fases integrantes (**planeamento, desenvolvimento, *build*, testes, *deploy***), tenta tornar uma aplicação o mais segura possível. Deste modo, a sua utilização, permite alcançar vários benefícios, como:

- **redução de custos**: através da identificação antecipada de possíveis vulnerabilidades do sistema;
- ***security-first***: uma vez que promove uma cultura orientada à segurança, tornando as equipas mais focadas em desenvolver de forma segura;
- **estratégia de desenvolvimento melhor definida**: ao serem definidos os critérios de segurança da aplicação, o processo de toma de algumas decisões torna-se mais simples (como a decisão de qual a tecnologia a adotar para o desenvolvimento, por exemplo).

Assim, ao longo deste relatório, será exposto todo o processo de desenvolvimento seguro de uma aplicação de gestão de *streamings* de música, contemplando a subscrição de planos, a gestão de utilizadores e o *upload* de faixas por exemplo. Para tal, este documento encontra-se dividido em **Análise e Desenho**, **Threat Modeling**, **Implementação**, **Testes** e ***Deployment***.

2 Análise e Design

Neste capítulo, está exposto o processo de análise e de desenho que preparou a etapa da implementação do projeto. Numa fase inicial, será explorado o domínio do problema, suportado pelo seu Modelo de Domínio, representando a interação dos diferentes componentes do sistema. De seguida, serão expostos os requisitos do projeto, divididos em requisitos funcionais e não funcionais.

2.1 Modelo de Domínio

Na figura seguinte, é apresentado o domínio da ação do projeto descrito, contendo os intervenientes necessários.

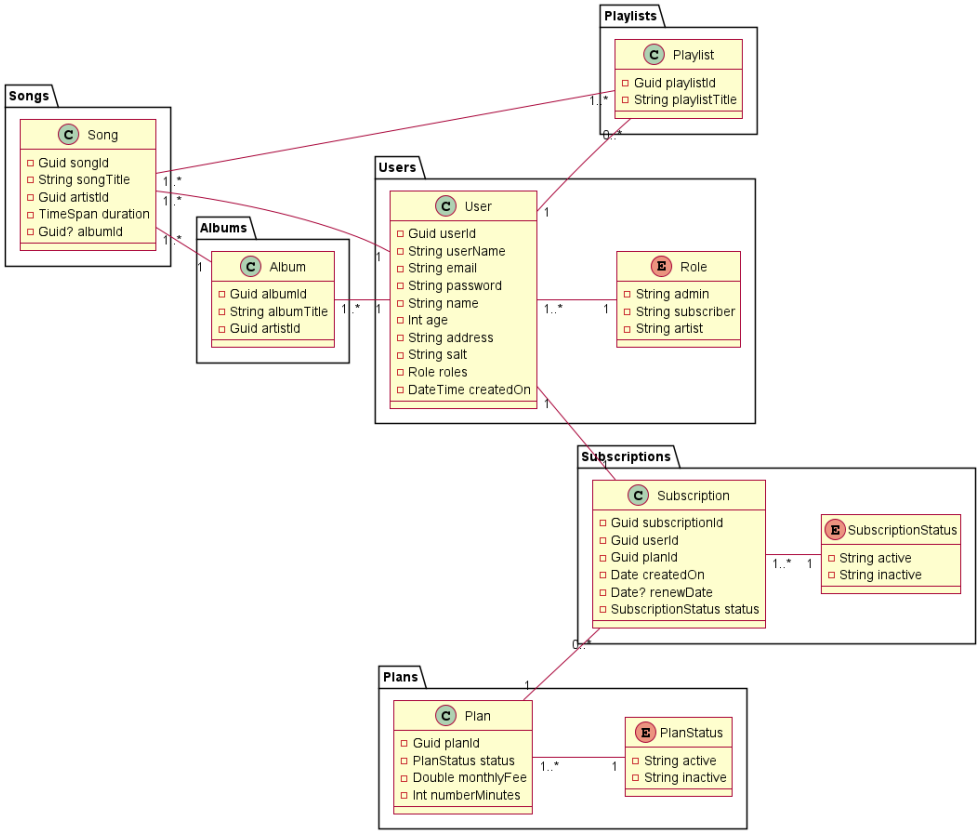

Figura 1: Modelo de Domínio

O domínio deste projeto é contituído por 6 agregados diferentes: **Plans**, **Subscriptions**, **Users** e **Songs**.

O agregado **Plans** é constituído pela classe **Plan**, representante dos planos existentes na plataforma que podem ser subscritos pelos clientes. É constituído por um ***id*** único, um ***status*** que indica se o plano ainda se encontra ativo ou não, um custo mensal (***mensalFee***) indicando o custo mensal que um subscritor deve pagar pelo plano e, ainda, por um ***numberMinutes***, indicando quantos minutos de *streaming* são permitidos no plano.

As subscrições representam um outro agregado e possuem, na sua constituição, os seguintes valores: ***id*** único, um ***userId*** remetendo para o subscritor, um ***planId*** indicando qual o plano a que se subscreve, uma data ***createdOn*** que marca o início da subscrição, uma data ***renewDate*** representando a data de renovação da subscrição quando aplicável e um ***status*** indicando se a subscrição se encontra ativa ou não.

O agregado ***Users*** é formado pela classe ***User***, que possui: o ***id*** único, o ***userName*** escolhido pelo utilizador aquando do seu registo no sistema, o ***email*** do utilizador, a ***password*** escolhida, o ***salt*** que é um valor aleatório adicional no processo de *hashing* da *password* para defender o utilizador de ataques de tabelas pré-calculadas (como acontece com as ***rainbow tables***[1]), o ***role*** do utilizador (podendo este ser um ***Admin*** com permissões de utilização e gestão da aplicação, um ***Subscriber*** ou um ***Artist*** e, ainda, uma data ***createdOn*** para registar o dia de registo do cliente na plataforma.

Existe, também, o agregado ***Songs*** contendo a classe ***Song***, que possui um ***id*** único, um título ***songTitle***, o ***artistId*** remetendo para o seu intérprete, a duração da música e o ***albumId***, um atributo opcional remetendo para o álbum a que a faixa pertence no caso de não ser um *single*.

[1]Tabelas pré-computadas que armazenam os resultados de funções de *hash* .

O agregado ***Albums*** possui a classe ***Album*** constituída pelo seu ***id***, título e o *id* do intérprete a que pertence.

Por fim, o agregado ***Playlists*** é constituído pela classe do mesmo nome representando listas de músicas que o utlizador pode construir associando as faixas que desejar. Esta classe possui um ***id*** e um título.

2.2 Requisitos Funcionais e Não Funcionais

Os requisitos de um sistema resultam das necessidades apontadas pelas partes interessadas no desenvolvimento de um projeto. Assim, representam regras de comportamento que uma solução deve ter em consideração, de forma a solucionar os problemas que a originaram.

2.2.1 Requisitos Funcionais

Nesta secção são apresentadas os requisitos funcionais do projeto a desenvolver. Estes requisitos expõem as funcionalidades do sistema, facilitando a sua compreensão.

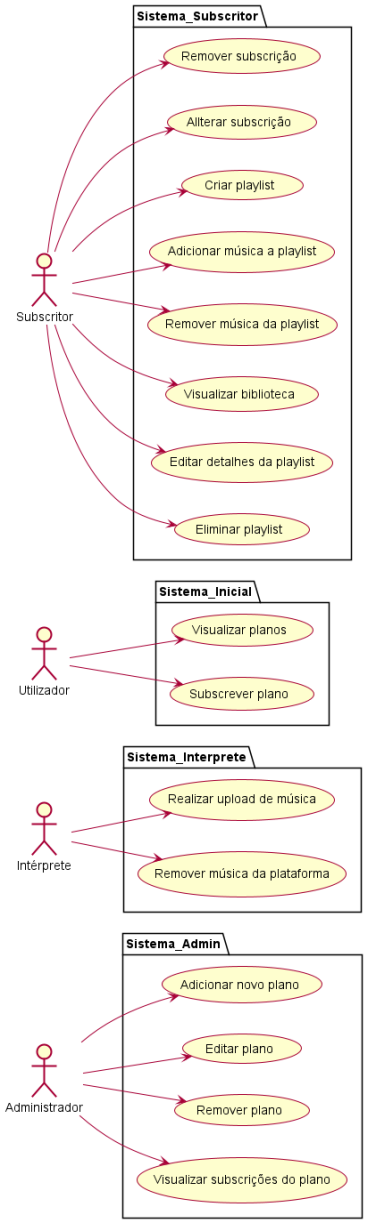

Figura 2: Diagrama de casos de uso

Assim, as funcionalidades do sistema são as seguintes:

- **US1**: Eu, **como Utilizador**, pretendo visualizar os planos existentes;
- **US2:** Eu, **como Cliente**, pretendo subscrever um plano;
- **US3:** Eu, **como Cliente**, pretendo remover a minha subscrição;
- **US4:** Eu, **como Cliente**, pretendo alterar a minha subscrição (mudar de plano);
- **US5:** Eu, **como Cliente**, pretendo criar uma nova *playlist*;
- **US6:** Eu, **como Cliente**, pretendo editar a minha *playlist*;
- **US7:** Eu, **como Cliente**, pretendo adicionar uma música à minha *playlist*;
- **US8:** Eu, **como Cliente**, pretendo remover uma música da minha *playlist*;
- **US9:** Eu, **como Cliente**, pretendo visualizar a minha biblioteca (*playlists*);
- **US10:** Eu, **como Intérprete**, pretendo realizar o *upload* de uma música da minha autoria;
- **US11:** Eu, **como Intérprete**, pretendo remover uma música da minha autoria;
- **US12:** Eu, **como Administrador**, pretendo adicionar um novo plano;
- **US13:** Eu, **como Administrador**, pretendo editar um plano;
- **US14:** Eu, **como Administrador**, pretendo remover um plano;
- **US15:** Eu, **como Administrador**, pretendo visualizar quantos subscritores possui cada plano.

2.2.2 Requisitos Não Funcionais

De seguida, serão expostos os requisitos não funcionais do sistema. Este tipo de requisitos representam as restrições e as capacidades de um determinado sistema. Para os identificar, recorreu-se ao modelo **FURPS+**.

- **Funcionalidade:**
  - O sistema deve permitir funcionalidades como a criação de diretórios/ficheiros, bem com a escrita/leitura de ficheiros.
- **Usabilidade:**
  - A utilização e compreensão da UI deve ser simples.
- **Confiança:**
  - O sistema deve possuir autenticação de utilizadores;
  - O sistema deve estar preparado para lidar com pelo menos 3 tipos de utilizador diferentes, possuindo permissões distintas (autorização).
- **Desempenho:**
  - O sistema deve possuir uma boa *performance*, não demorando muito tempo a processar os pedidos efetuados (para tal, devem ser utilizados DTOs, permitindo o envio de apenas os dados necessários para cada operação).
- **Suporte:**
  - As funcionalidades devem ser testadas por meio de testes unitários, de integração e *end-to-end*.

  - Todos os componentes devem ser facilmente implantados através de uma *pipeline*.
  - O sistema deve possuir um mecanismo de *logging* que permita identificar e registar diferentes tipos de eventos.
  -

- **Restrições de Design:**
  - O sistema a desenvolver deveria possuir um domínio com, no mínimo, 3 agregados diferentes.
- **Restrições de Interface:**
  - Devem existir interfaces distintas tendo em conta o estado do registo do utilizador (ex.: interface de utilizador não registado vs. interface de subscritor);
  - Deve ser possível a visualização dos detalhes de um plano, subscrição e biblioteca de músicas.
- **Restrições de Implementação:**
  - O projeto deve possuir duas aplicações que comunicam entre si: uma aplicação de ***back-end*** e uma de ***front-end***;
  - A aplicação de *back-end* deve ser consistir numa **WEB API** e numa **base de dados relacional** e não local.

2.3 Arquitetura

O sistema desenvolvido para este projeto tem dois componentes principais - a aplicação visível ao cliente (*o frontend*) que se trata de uma aplicação cliente *web* com que o utilizador

do sistema interage, e o *backend* que é o servidor que recebe pedidos da aplicação cliente e efetua as operações necessárias para ação ser concluída com sucesso.Como tal, é necessário que o *frontend* faça pedidos (no nosso caso usando *HTTP/s*) ao *backend*.

Adicionalmente, o *backend* pode ainda ter de interagir com outros componentes como consequência do pedido feito pelo cliente. Por exemplo, quando um cliente cria uma subscrição esta nova subscrição tem de ser armazenada numa base de dados de forma a informação se tornar persistente. No contexto deste projeto, o servidor comunica com uma base de dados *Microsoft SQL Server* e utiliza ainda funcionalidades da API de ficheiros do sistema operativo do servidor em que a aplicação *backend* esta implantada.

Na figura 3, é apresentado o diagrama de componentes do sistema desenvolvido no contexto deste projeto.

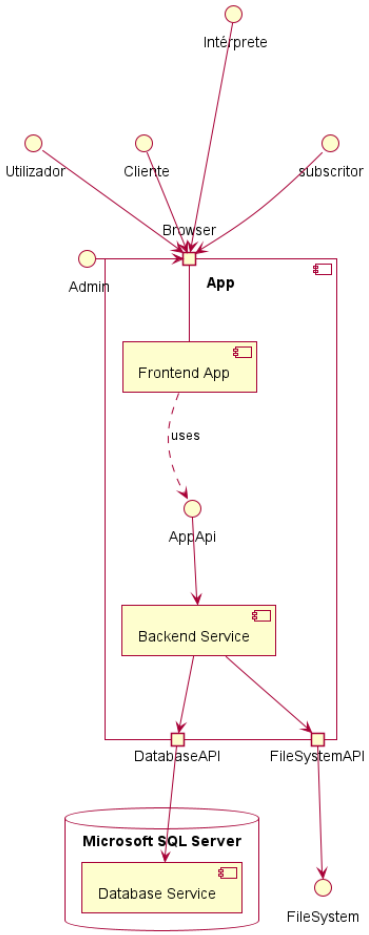

Figura 3: Diagrama de componentes

Adicionalmente, e de forma a ter um melhor entendimento da forma como o sistema esta desenvolvido, implantado, e bem como possíveis vulnerabilidades em termos de segurança foi ainda desenvolvido o diagrama com a vista física do sistema, onde são especificadas as diferentes partes da infraestrutura deste projeto e a forma como estas comunicam. Este diagrama pode ser analisada na figura 4

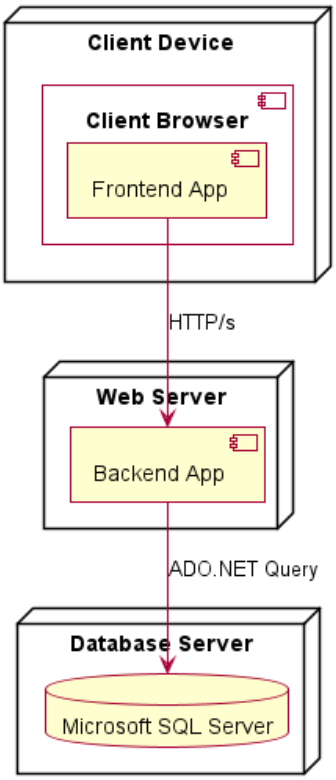

Figura 4: Diagrama de componentes

3 *Threat Modeling*

3.1 Decomposição do Sistema

3.1.1 Dependências Externas

Na tabela que se segue, são apresentadas todas as dependências externas que poderão ser utilizadas como pontos de ataque. A sua identificação será feita com recurso a:

1. **Id** - Um identificador único para cada dependência;

2. **Descrição** - Explicação da dependência.

Id	Descrição
1	O Sistema Web irá correr num servidor Linux que por sua vez irá usar contentores Linux para hospedar a API. A instalação é responsável pelo ISEP e não garante a instalação das mais recentes *Security Patches*.
2	A Base de Dados irá usar o Sistema SQL Server, hospedado num contentor Docker. A instalação é responsável pelo ISEP e não garante a instalação das mais recentes *Security Patches*.
3	Todos os sistemas estarão hospedados em servidores pertencentes à infraestrutura do ISEP

Tabela 1: Dependências externas

3.1.2 Pontos de Entrada

Os pontos de entrada representam interfaces com que os atacantes poderão interagir de forma a atacar a aplicação. Na seguinte tabela, são apresentados os pontos de entrada do sistema em análise, organizando a informação da seguinte forma:

1. **Id** - Identificador único para cada ponto de entrada;
2. **Nome** - Nome descritivo que identifica o ponto e fornece uma breve elucidação acerca do mesmo;
3. **Descrição** - Descrição que visa fornecer detalhes acerca da interação/processo que ocorre em cada ponto:
4. **Nível de confiança** - Nível de acesso requerido em cada ponto.

Id	Nome	Descrição	Nível de confiança
1	Portas HTTPS	Uma vez que a aplicação web será acessível via TLS, todas as suas páginas possuirão portas HTTPS como ponto de entrada.	1-Utilizador anónimo; 2-Utilizador com credenciais válidas; 3-Utilizador com credenciais inválidas; 4-Administrador.
2	Página de login	Administradores, subscritores e intérpretes têm de efetuar login na aplicação para a conseguir utilizar.	1-Utilizador anónimo; 2-Utilizador com credenciais válidas; 3-Utilizador com credenciais inválidas; 4-Administrador.

Tabela 2: Pontos de entrada

3.1.3 Ativos

Na tabela seguinte são apresentados todos os ativos recolhidos como possíveis fontes de interesse no caso de um ataque. Estes ativos podem ser tanto físicos como abstratos. A informação está organizada da seguinte forma:

1. **Id** - Um identificador único para cada ativo;
2. **Nome** - Um nome descritivo do ativo;
3. **Descrição** - Explicação do que é o ativo e a razão de ser protegido;
4. **Níveis de acesso** - Níveis de acesso necessários para aceder ao ativo.

Id	Nome	Descrição	Nível de confiança
1	**Utilizadores, Músicos e Administradores**	**Ativos relacionados com utilizadores gerais do sistema, músicos e administradores de planos.**	

Id	Nome	Descrição	Nível de confiança
1.1	Detalhes de login de utilizadores	As credenciais de login que um utilizador comum irá utilizar para aceder ao sistema	2-Utilizador com credenciais válidas; 4-Intérprete; 5-Administrador de base de dados; 7-Processo de serviço web; 8-Utilizador de leitura de base de dados; 9-Utilizador de leitura/escrita de base de dados.
1.2	Detalhes de login de Intérpretes	As credenciais de login que um intérprete irá utilizar para aceder ao sistema	4-Intérprete; 5-Administrador de base de dados; 7-Processo de serviço web; 8-Utilizador de leitura de base de dados; 9-Utilizador de leitura/escrita de base de dados.

Id	Nome	Descrição	Nível de confiança
1.3	Dados Pessoais	O sistema irá armazenar dados pessoais relativos aos seus utilizadores. Exemplo: Nome, Idade, Morada.	5-Administrador de base de dados; 6-Administrador de sistema; 7-Processo de serviço web; 8-Utilizador de leitura de base de dados; 9-Utilizador de leitura/escrita de base de dados.
2	**Sistema**	**Ativos relacionados com o Sistema**	
2.1	Disponibilidade do website de streaming de música	O Website deve estar disponível 24h por dia e poder ser acessado por todos os utilizadores e intérpretes	5-Administrador de base de dados; 6-Administrador de sistema;
2.2	Possibilidade de executar código fonte como processo de serviço web	Possibilidade de executar código no serviço como sendo o processo do serviço web	6-Administrador de sistema; 7-Processo de serviço web

Id	Nome	Descrição	Nível de confiança
2.3	Possibilidade de executar SQL como utilizador de leitura de base de dados	Possibilidade de executar consultas à base de dados, recolhendo assim qualquer informação contida na base de dados do sistema	5-Administrador de base de dados; 8-Utilizador de leitura de base de dados; 9-Utilizador de escrita/leitura de base de dados
2.4	Possibilidade de executar SQL como utilizador de escrita/leitura de base de dados	Possibilidade de executar tanto consultas como operações de inserção/atualização, tendo acesso de escrita e leitura a todos os dados do sistema	5-Administrador de base de dados; 9-Utilizador de escrita/leitura de base de dados
3	**Sistema**	**Ativos relacionados com o sistema de streaming de música**	
3.1	Sessão de login	Sessão de login de um utilizador do sistema. Este pode ser um utilizador comum ou um intérprete.	2-Utilizador com credenciais válidas; 4-Intérprete
3.2	Acesso ao servidor de base de dados	O acesso ao servidor de base de dados permite administrá-lo, garantindo acesso aos utilizadores da base de dados e às informações nela contidas.	5-Administrador de base de dados

Id	Nome	Descrição	Nível de confiança
3.3	Acesso a dados de auditoria(*logging*)	Os dados de auditoria mostram todos os eventos (configurados para tal) que ocorreram no sistema, despoletados pelos vários utilizadores.	6-Administrador de sistema

Tabela 3: Ativos

3.1.4 Níveis de confiança

Os níveis de confiança representam os tipos de acesso ao sistema que diferentes entidades externas poderão obter., definindo os privilégios e direitos de acesso necessários para cada ativo. Os níveis de confiança serão apresentados segundo os parâmetros:

1. **Id** - Identificador único para cada nível de confiança;
2. **Nome** - Nome descritivo que permite a identificar a entidade externa detentora do nível de confiança;
3. **Descrição** - Descrição explicativa acerca da entidade a que é concedido o nível.

Id	Nome	Descrição
1	Utilizador anónimo	Utilizador que interage com o sistema mas que não forneceu credenciais de login válidas.
2	Utilizador com credenciais válidas	Utilizador que interage e se conecta com o sistema após fornecer credenciais de login válidas.
3	Utilizador com credenciais inválidas.	Utilizador que interage com o sistema e se tenta conectar mas não fornece credenciais válidas.
4	Intérprete	O intérprete possui permissões para realizar o upload de músicas da sua autoria na aplicação.
5	Administrador de Base de Dados	O administrador de base de dados possui permissões para aceder e manipular as tabelas do sistema.
6	Administrador de Sistema	O administrador possui permissões para adicionar, remover e editar planos de subscrição.
7	Processo de serviço de web	O processo de serviço web tem acesso a todos os componentes do sistema.
8	Utilizador de leitura de base de dados	Utilizador com autorização para ler o conteúdo das tabelas da base de dados.
9	Utilizador de escrita/leitura de base de dados	Utilizador com autorização para ler e manipular o conteúdo das tabelas da base de dados.

Tabela 4: Níveis de confiança

3.2 *Data Flow*

Nesta secção, é apresentado o *Data Flow Diagram* que explica como os dados são transmitidos pelos diferentes componentes do sistema. Este diagrama é apresentado na Figura 5.

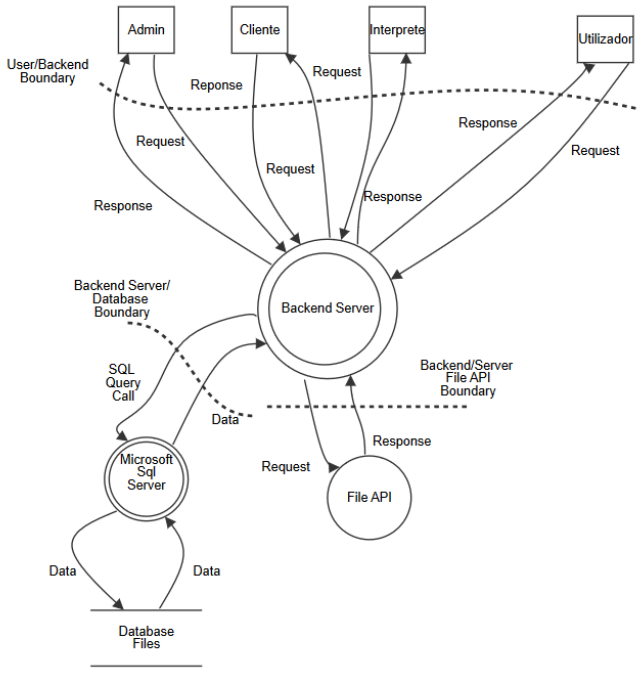

Figura 5: Diagrama de componentes

Assim sendo, tal como podemos observar na Figura 5 existem quatro componentes/ativos no sistema: o cliente (representa pelos diferentes tipos de utilizadores), o *Backend Server*, a base dados (*Microsoft SQL Server* e o sistema de gestão de ficheiros (*FileApi*). Adicionalmente podemos ver que os clientes fazem pedidos ao servidor *backend*, e este por sua vez dependendo do processo que tem de executar pode ter de fazer pedidos à base de dados ou ao sistema de gestão de ficheiros.

Cada um destes componentes, por sua vez envia a resposta ao componente que fez o pedido (ou seja a base de dados e o serviço de ficheiros enviam dados ao servidor *backend* ,

que por sua vez envia a resposta ao *cliente*).

3.3 Determinação das ameaças

Nesta secção, serão apontadas as potenciais ameaças do sistema, tendo por base os ativos que o constituem. De forma a conseguir avaliar e priorizar as ameaças encontradas, serão apresentadas as seguintes etapas:

- Categorização;
- Árvores de ameaça;
- Priorização de ameaças;
- *Misuse cases*.

3.3.1 Categorização

De forma a categorizar as ameaças encontradas, será aplicado o modelo ***STRIDE***, que permite a identificação de ameaças tendo em vista os potenciais objetivos do atacante. Para tal, deve ser feita uma análise dos ativos do sistema tendo em conta a possibilidade de ocorrência das seguintes ameaças:

1. ***S****poofing*: o *hacker* faz-se passar por uma outra entidade com acesso ao sistema;
2. ***T****ampering*: manipulação não autorizada dos dados do sistema;
3. ***R****epudiation*: negação ou rejeição de determinadas ações;
4. ***I****nformation Disclosure*: acesso não autorizado a dados ou exposição dos mesmos;
5. ***D****enial of Service*: comprometer o acesso do sistema a determinados recursos;

6. ***E**levation of privilege*: aquisição não autorizada de privilégios elevados de acesso ao sistema.

Tendo em conta as ameaças apresentadas, pode realizar-se a seguinte análise quanto à sua relação com os ativos do sistema:

- ***Spoofing***: tendo em conta a estrutura do sistema em análise, um atacante poderá explorar os seguintes componentes:
  - ***Backend Server***: um *hacker* pode falsificar requisições a este servidor possuindo um acesso não autorizado aos dados nele armazenados;
  - ***Client:*** um atacante pode falsificar a sua identidade, fazendo passar-se por um utilizador legítimo e aceder às funcionalidades da plataforma;
  - **Base de Dados:** possuindo acesso à base de dados, um atacante poderá fazer-se passar por um utilizador com privilégios de edição e manipular os dados dos clientes e/ou do sistema;
  - ***File Server:*** um invasor poderá tirar proveito deste componente para adicionar, remover ou editar ficheiros de forma a injetar *malware* na plataforma de *streaming* (que poderá ser transmitido aos clientes).

  Mitigações

  De forma a mitigar a ameaça da *spoofing*, deverão ser implementados métodos de autenticação reforçados para todas as partes do sistema. Para além disso, o recurso a metodologias de criptografia para proteger os dados do sistema e a adoção de medidas de validação de entrada de dados na base de dados, são também exemplos de medidas de prevenção deste ataque.

- ***Tampering:*** para efetuar a manipulação de dados não autorizada, um atacante poderá explorar os ativos identificados da seguinte maneira:

  - ***Backend Server:*** um utilizador mal intencionado pode tentar inserir código malicioso diretamente nas requisições enviadas ao servidor numa tentativa de explorar vulnerabilidades como *XSS* ou *SQL Injection*. Adicionalmente, ele pode intercetar e modificar os dados transmitidos entre o cliente e o servidor, alterando parâmetros ou valores enviados nas requisições.
  - ***Client:*** Neste ativo, um atacante pode fazer *tampering* modificando dados armazenados no cliente como *cookies*.
  - ***Base de dados:*** um *hacker* poderá aproveitar este componente para ganhar acesso não autorizado, ou ainda mudar dados sensíveis como os detalhes de um *utilizador* ou informações de pagamento.
  - ***File Server:*** um utilizador mal intencionado pode aproveitar este ativo do sistema de uma forma em que altere arquivos legítimos por outros com conteúdo malicioso ou alterar os meta-dados dos ficheiros (informações sobre o autor, a data de criação/modificação ou outros atributos associados aos arquivos).

Mitigações

De forma a mitigar a ameaça da *tampering*, serão implementados alguns mecanismos como a **encriptação** dos dados. Isto será feito em três níveis:

- Primeiramente, todas as suas páginas possuirão portas HTTPS como ponto de entrada, pelo que os dados transmitidos entre o cliente e o servidor serão transmitidos de forma encriptada.

- Adicionalmente, dados confidenciais como informação sensível do cliente (o seu NIF, morada, entre outros), passwords serão encriptados na base de dados.
- as respostas serão digitalmente assinados (utilizando o *JWT standard* de forma a garantir que o conteúdo nao foi alterado.

.

- ***Repudiation*:** para efetuar um ataque de negação, um utilizador mal intencionado pode explorar os seguintes detalhes dos ativos:
  - ***Backend Server:*** um utilizador mal intencionado pode, por exemplo, efetuar uma subscrição e depois afirmar que não a efetuou. Se o sistema, não tiver um sistema de *logging* adequado, então poderá ser complicado identificar se um utilizador foi responsável ou não por uma ação.
  - ***Cliente:*** um utilizador malicioso, pode efetuar uma subscrição paga no sistema, e depois afirmar que não o fez com o objetivo de obter um rembolso. Mais uma vez, s o sistema, não tiver um sistema de *logging* adequado, então poderá ser complicado identificar se um utilizador foi responsável ou não por uma ação.
  - ***Base de dados**:* Um atacante pode tentar alterar registos na base de dados relacionados a informações de pagamento, como datas de subscrição ou detalhes da assinatura, para negar ter feito um pagamento ou uma transação específica.
  - ***Sistema de Ficheiros**:* Um utilizador malicioso, pode tentar modificar ou remover o registo de *logs* do sistema de forma a negar ter efetuado determinadas ações.

Mitigações

De forma a prevenir ataques deste tipo, a aplicação *web* desenvolvida neste projeto, possuirá um sistema de *logging* que permita registar eventos/as ações de um utilizador e adicionalmente os pedidos efetuados serão digitalmente assinados - mais uma vez, utilizando o *JWT standard*, sendo que durante a etapa de autorização será gerado um *token* que terá de ser incluído no *header* de todos os pedidos e que permitirá identificar se um utilizador é quem afirma ser.

- ***Information Disclosure*:** para efetuar um ataque do tipo *Information Disclosure* um atacante pode explorar vulnerabilidades dos ativos tais como:
  - ***Backend Server*:** um atacante pode explorar má validação dos dados introduzidos, configuração dos *queries SQL*, para aceder a dados confidenciais de outros utilizadores como nome, morada, password entre outros dados.Adicionalmente um atacante pode explorar falhas no sistema de autenticação e autorização da *API* para aceder a *endpoints* que não devia e obter dados que não devia ou pode ainda explorar falhas na respostas a pedidos inválidos para entender a estrutura da *API*.
  - ***Cliente*:** Um utilizador malicioso pode explorar falhas de segurança no cliente para acessar informações pessoais dos usuários, como nome, endereço, informações de pagamento ou histórico de reprodução. Outra hipótese, é que este aproveite falhas na interface do cliente para aceder a conteúdo que só devia estar disponível para utilizadores com uma subscrição ativa.
  - ***Base de dados*:** Um atacante pode mais uma vez explorar vulnerabilidades como a utilização de segmentos de código SQL *(Structured Query Language)* para manipular uma base de dados de forma direta, com o objetivo de obter dados

de outros utilizadores como as palavras passes das suas contas, o nome, morada, entre outras informações.

- ***Sistema de ficheiros*:** Um utilizador malicioso, pode explorar vulnerabilidades no sistema de autenticação e autorização bem como na configuração do mesmo e obter dados dos ficheiros de *log*. Este ficheiro pode ter informações sobre atividades de outros utilizadores, bem como alguns dados confidenciais. Outra forma de um atacante explorar este ativo, passa por tentar modificar meta-dados de músicas de outros autores nos arquivos do sistema para inserir informações falsas ou enganosas - por exemplo mudar o nome do compositor, a duração da música ou o conteúdo da mesma.

Mitigações

De forma a mitigar os problemas mencionados, a aplicação *web* desenvolvida neste projeto apresentará as seguintes soluções :

- *ORM* [2], de forma a evitar a concatenação direta de *strings* transmitidas a base de dados.
- Tal como mencionado previamente, utilizará *tokens JWT (JSON Web Tokens)* para autorização e controle de acesso granular.
- Implementará um sistema de privilégios mínimos e um sistema de controlo de acesso baseado em papéis *(RBAC)* [3] de forma a garantir que cada utilizador só tem acesso as funções para o qual devia.

[2] *ORM:Object–relational mapping* é uma técnica para aproximar o paradigma de desenvolvimento de aplicações orientadas a objetos ao paradigma do banco de dados relacional

[3] RBAC: abreviatura de *Role Based Access Control*

  - Por fim, será garantido que as mensagens de *log* são poderão ser acedidas por utilizadores com as permissões devidas (por exemplo administradores) e será garantido que estas não guardam dados confidenciais do utilizador da plataforma.

- ***Denial of Service**:* para efetuar um ataque do tipo *Denial of Service* um atacante pode explorar vulnerabilidades dos ativos tais como:

  - ***Backend Server*:** um utilizador malicioso pode enviar solicitações maliciosas ou de carga elevada ao servidor, de modo a esgotar os recursos disponíveis e tornar o serviço inacessível para usuários legítimos.
  - ***Cliente*:** um atacante pode automatizar um *script* que permita o envio massivo de solicitações a partir do cliente com o objetivo de sobrecarregar o servidor e impedir utilizadores legítimos de aceder ao serviço.
  - ***Base de Dados*:** Um utilizador com intenções maliciosas pode enviar consultas complexas ou maliciosas à base de dados para consumir recursos significativos e retardar as operações legítimas.
  - ***Sistema de ficheiros*:** um atacante pode submeter vários arquivos de música num curto espaço de tempo de forma a gerar um grande volume de operações de escrita no sistema de ficheiros, e com objetivo de impactar o desempenho e a disponibilidade do serviço.

Mitigações

De forma a mitigar um ataque do tipo de negação de serviço serão implementadas as seguintes medidas no sistema:

  - Será implementado um sistema de ***rate limiting*** aos pedidos de um cliente. Este sistema limitará o número de pedidos efetuados por um utilizador num determinado período de tempo.
  - Será utilizado uma funcionalidade da ferramenta *ORM* que permite limitar tempo de consulta ou transação, de forma a a impedir que consultas lentas ou de longa duração impactem o sistema. O sistema de *logging* vai registar as consultas canceladas, com objetivo de permitir, posteriormente, a sua otimização e melhoria de segurança.

- ***Elevation of privilege****:* por fim, de forma a efetuar um ataque de elevação de privilégios um utilizador mal intencionado pode em cada um dos ativos explorar as seguintes vulnerabilidades:
  - ***Backend Server*:** um atacante pode enviar pedidos com *tokens* e identidades forjadas. Caso a segurança do sistema tenha falhas, ele poderá explorar essas vulnerabilidades para se passar por um utilizador legítimo e com privilégios elevados.
  - ***Cliente*:**um utilizador malicioso pode utilizar falhas de segurança e da resposta enviada pelo servidor para obter o *token* de autenticação e a partir deste obter acesso a funcionalidades *premium* ou de um administrador.
  - ***Base de dados*:** um atacante pode explorar vulnerabilidades de *SQL Injection* de modo a modificar, eliminar e acessar dados na base de dados. Estes dados incluem, por exemplo, dados de utilizadores que muitas vezes são sensíveis, como passwords, moradas, números de identificação fiscal.
  - ***Sistema de ficheiros*:** um utilizador mal intencionado pode explorar vulnera-

bilidades no sistema de conceder privilégios ao utilizador para aceder e modificar ficheiros do sistema ao qual não devia ter acesso. No contexto da API, isto poderia por exemplo levar a que o atacante conseguisse alterar meta-dados das músicas armazenadas no servidor que não são da sua autoria.

Mitigações

De forma a mitigar um ataque do elevação de privilégios no sistema a desenvolver serão implementadas as seguintes medidas no sistema:

- Implementação do principio do privilégio mínimo no acesso a API, de modo a garantir que cada utilizador só tem os privilégios necessários para conseguir executar as funcionalidades a que deve ter acesso.
- os dados transmitidos entre o cliente e o servidor serão transmitidos de forma encriptada, uma vez que a aplicação estará apenas acessível via HTTPS.
- Os *queries* da base de dados serão parametrizados, como mencionado previamente, de forma a evitar a concatenação de *strings* que possam resultar num ataque de Injeção de SQL.
- Idealmente, será utilizada uma autenticação multifator com o objetivo de aumentar a segurança do sistema.
- Adicionalmente as *passwords* terão de ter um mínimo de 12 caracteres, e terão de ter letras maiúsculas,minúsculas, números e símbolos.
- Por fim, as *passwords* selecionadas não poderão estar no *Top 1000* das *pwned password*.

3.3.2 Árvores de ameaça

As árvores de ameaça fornecem uma descrição da segurança dos sistemas baseando-se na exploração de vários ataques. Assim para a aplicação presente apresentamos vários resultados de ataques e as suas árvores de ameaça:

- Aquisição das credenciais de um utilizador:

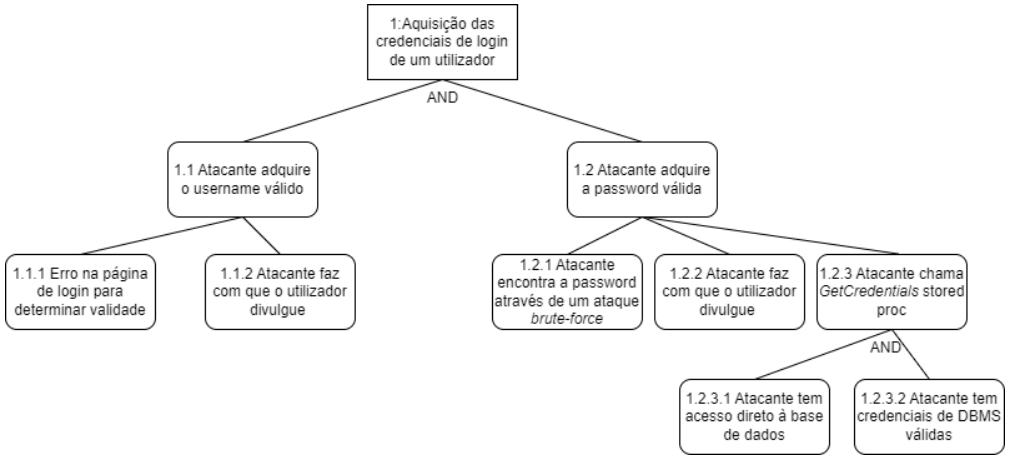

Figura 6: Árvore de ameaça da aquisição de credenciais de um utilizador

Neste cenário o atacante faz uso de técnicas de *brute-force* e exploração de vulnerabilidades na página de *login* tal como de coação dos utilizadores para obtenção das credencias.

Para **mitigar** a obtenção das credenciais, devem ser implementadas fortes medidas de validação na autenticação tal como consciencializar os utilizadores e responsáveis pela aplicação, deste tipo de ataques.

- Interrupção do serviço:

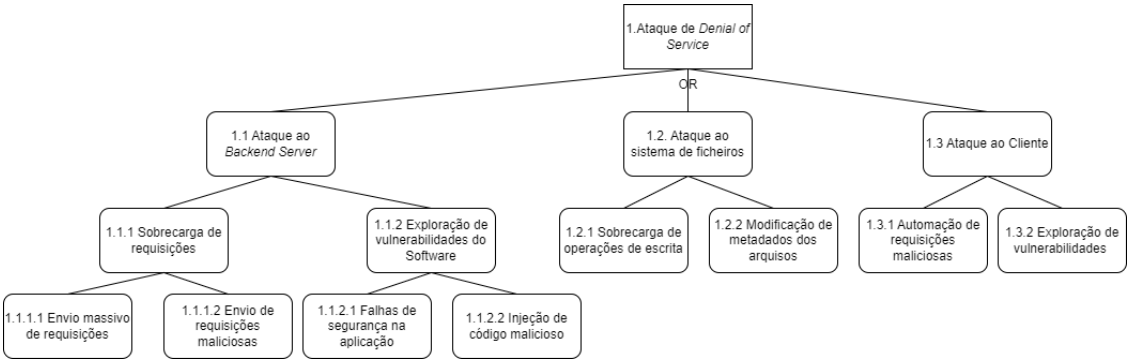

Figura 7: Árvore de ameaça da interrupção do serviço

Para **mitigar** os ataques de interrupção devem ser adotadas as medidas mencionadas na secção da categorização e dos *misuse cases*.

- Manipulação de dados de música:

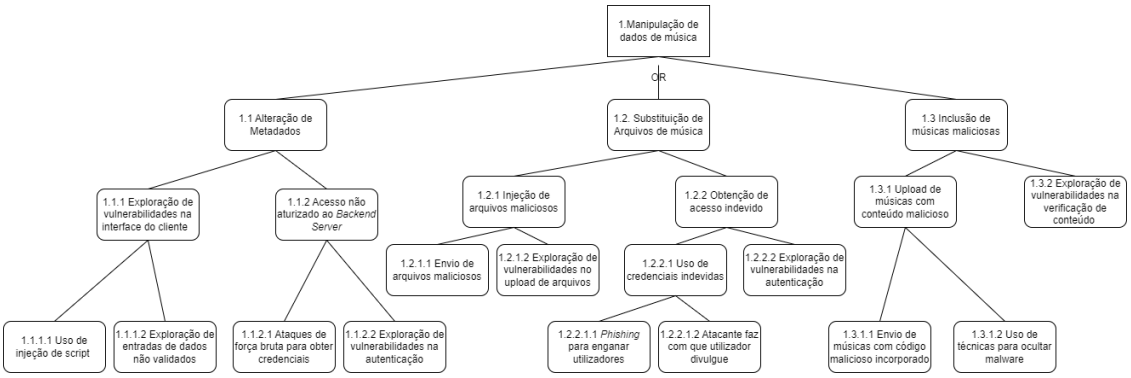

Figura 8: Árvore de ameaça de manipulação de dados de música

De forma a **mitigar** a manipulação de dados de música, devem ser implementadas **validações dos dados de entrada** rigorosas para prevenir injeções de *script*, tal como realizar uma verificação à integridade dos mesmos. Deve também haver **controlo de acesso**, com reforço na autenticação e autorização dos servidores, e utilização de técnicas de criptografia

para proteger a comunicação. Para além destas medidas é importante realizar **análises de segurança** e utilização de soluções antivírus e *anti-malware*, é necessário também **consciencializar os utilizadores** acerca das possíveis ameaças, e por fim, **monitorizar e ter um plano de resposta**.

3.3.3 Priorização de ameaças

De forma a priorizar as ameaças foi utilizado o modelo **DREAD**, este modelo responde a questões de análise de ameaças para cada fator de risco da seguinte forma:

- ***D**amage:* Qual o impacto do ataque sucedido?
- ***R**eproducibility:* Quão fácil é reproduzir o ataque?
- ***E**xploitability:* Quanto tempo, esforço e perícia é necessário para explorar a vulnerabilidade?
- ***A**ffected Users:* Qual a percentagem de utilizadores afetados pelo ataque?
- ***D**iscoverability:* Quão fácil é para um atacante descobrir a vulnerabilidade?

Assim, para cada um dos ataques já mencionados na categorização, fez-se a seguinte avaliação:

- **Spoofing**:
  - ***D**amage:* Alto. Um atacante poderá ter acesso não autorizado a dados sensíveis, a sua manipulação ou comprometendo a integridade do sistema. (9)
  - ***R**eproducibility:* Moderado dependendo das medidas de segurança implementadas. (7)

- ***E**xploitability:* Alto. Caso haja falhas de autenticação, é relativamente fácil ganhar acesso indevido. (9)
- ***A**ffected Users:* Todos os utilizadores do sistema. (10)
- ***D**iscoverability:* Moderadamente fácil, pois os métodos de falsificação são bem conhecidos na comunidade de hackers. (8)

Pontuação DREAD: 9

- **Tampering**:
  - ***D**amage:* Alto. Pode levar a manipulação de dados sensíveis, comprometendo a integridade do sistema. (9)
  - ***R**eproducibility:* Moderado. Dependendo das vulnerabilidades exploradas. (7)
  - ***E**xploitability:* Alto. Se houver falhas de validação ou autenticação é relativamente fácil explorar. (8)
  - ***A**ffected Users:* Todos os utilizadores cujos dados são manipulados. (9)
  - ***D**iscoverability:* Moderadamente fácil, pois as técnicas de manipulação são bastante conhecidas. (8)

Pontuação DREAD: 8

- **Repudiation**:
  - ***D**amage:* Moderado, porque pode resultar em disputas de responsabilidade por ações realizadas no sistema. (6)

- ***R**eproducibility:* Moderado. O mesmo utilizador que faz a negação não o consegue fazer várias vezes, tal como requer conhecimento sobre como negar ações realizadas. (6)
- ***E**xploitability:* Moderado, pois pode exigir informação sobre como manipular registos e logs. (7)
- ***A**ffected Users:* Potencialmente alto, pois pode afetar a confiança de todos os usuários no sistema. (7)
- ***D**iscoverability:* Moderadamente fácil. As falhas nos registos e logs podem ser identificadas com alguma facilidade.(8)

Pontuação DREAD: 7

- **Information Disclosure**:
  - ***D**amage:* Muito alto, visto que resulta na exposição de dados confidenciais. (9)
  - ***R**eproducibility:* Moderado, pois requer conhecimento sobre como explorar vulnerabilidades de divulgação de informações. (7)
  - ***E**xploitability:* Moderado, pois exige habilidades técnicas para identificar e explorar as vulnerabilidades associadas a este ataque. (7)
  - ***A**ffected Users:* Potencialmente alto, pois pode afetar a privacidade de todos os usuários do sistema. (9)
  - ***D**iscoverability:* Moderado a alto, o atacante pode fazer o uso de vários tipos de ataques para explorar vulnerabilidades no sistema e expor dados confidenciais. (8)

Pontuação DREAD: 8

- **Denial of Service**:
  - ***D**amage:* Alto. Pode resultar na interrupção do serviço para todos os utilizadores legítimos. (9)
  - ***R**eproducibility:* Moderado a alto. Os ataques de negação de serviço podem ser relativamente simples de executar. (8)
  - ***E**xploitability:* Baixo a moderado. Não requer necessariamente habilidades técnicas avançadas, mas pode exigir recursos significativos. (8)
  - ***A**ffected Users:* Potencialmente alto. Pode afetar todos os usuários do sistema. (10)
  - ***D**iscoverability:* Moderadamente fácil. Os métodos de negação de serviço são facilmente encontrados. (9)

Pontuação DREAD: 9

- **Elevation of privilege**:
  - ***D**amage:* Alto. Pode resultar na obtenção de acesso não autorizado e controlo total sobre o sistema. (9)
  - ***R**eproducibility:* Baixo a moderado. Pode exigir a exploração de múltiplas vulnerabilidades. (7)
  - ***E**xploitability:* Baixo a moderado, pois pode exigir um conhecimento profundo do sistema e das vulnerabilidades específicas. (7)

- ***A**ffected Users:* Potencialmente alto, porque pode comprometer todos os utilizadores do sistema. (9)
- ***D**iscoverability:* Moderadamente difícil. As vulnerabilidades de elevação de privilégios podem ser mais difíceis de detetar e explorar. (6)

Pontuação DREAD: 8

3.4 *Misuse cases*

Outra etapa importante do *Threat Modeling* passa por elaborar ***Misuse Cases***, que contemplam cenários de abuso que retratam ações maliciosas não desejadas pela organização. A elaboração destes casos ajuda a definir estratégias de proteção mais apropriadas.

Assim, serão apresentados alguns casos de abuso para o sistema em análise.

3.4.1 Acesso não autorizado:

Tal como mencionado previamente ao longo deste relatório, um atacante pode explorar vulnerabilidades do sistema de *login*, e com isto aceder a conta de outras utilizadores que podem ter permissões para funcionalidades *premium* do sistema, ou no pior dos casos permissões administrativas.

Deste modo, um dos casos de abuso desenvolvidos foi o caso de abuso para o processo de *login*. Este diagrama pode ser observado na Figura 9.

Assim sendo, este diagrama permite observar não só como o processo de *login* vai funcionar - o utilizador vai introduzir um email e a password (sendo que esta password tem de ter um mínimo de doze caracteres que incluam letras maiúsculas, minúsculas, símbolos e números e não pode estar no top 1000 das *passwords pwned*), depois caso seja bem sucedido

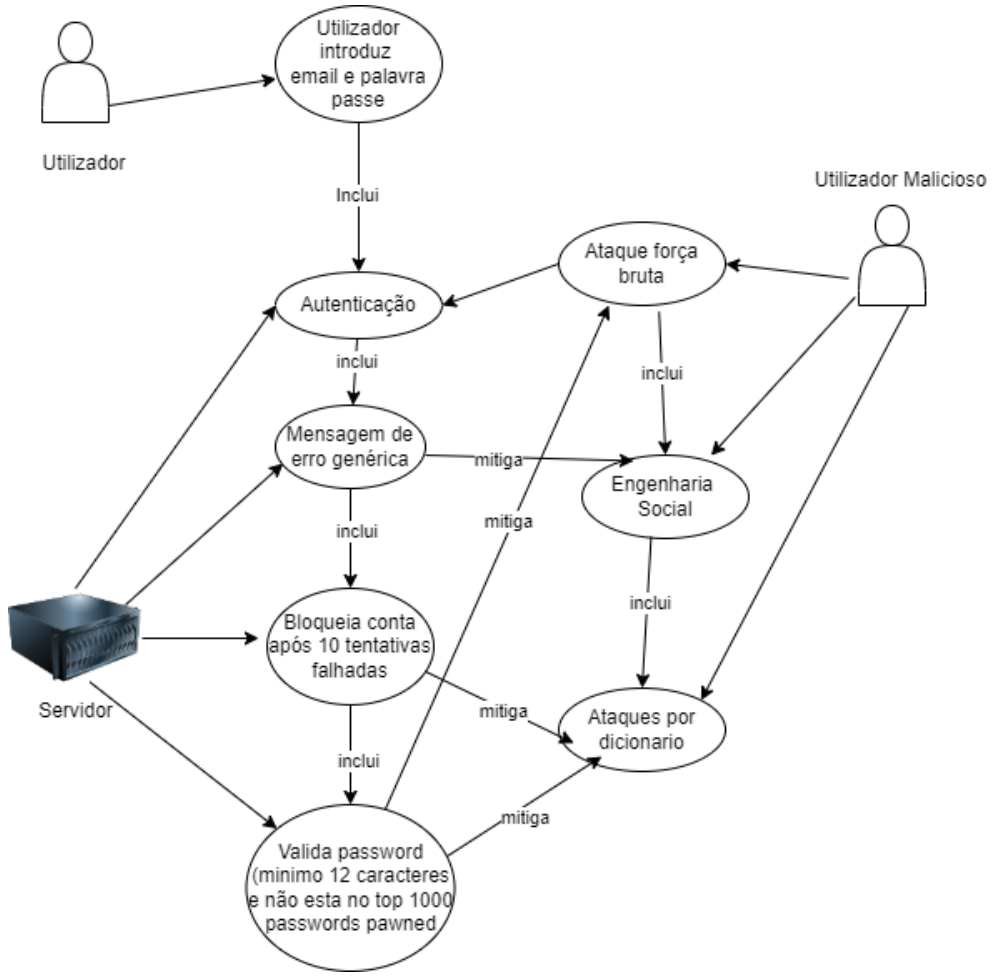

Figura 9: Caso de abuso para o processo de *Login*

o *login* fica completo, se não é definida uma mensagem de erro genérico e a conta é bloqueada ao fim de dez tentativas -, como também os possíveis ataques que um utilizador mal intencionado pode explorar (ataques de força bruta, ataques de Engenharia Social, Ataques por dicionário).

Adicionalmente, este caso de abuso apresenta ainda a forma como cada uma das medidas do sistema de *login* ajuda a mitigar um ataque. Neste sentido, por exemplo as regras de tamanho de password ajudam a mitigar ataques de força bruta e de dicionário, e uma

mensagem de erro genérica ajuda a mitigar ataques por Engenharia Social, pelo que ficam explicitas as estratégias de proteção que serão utilizadas no sistema, de modo a garantir a segurança dos utilizadores.

3.5 *Denial of Service*

Um *Denial of Service* é um ataque cujo objetivo é sobrecarregar um servidor de forma a torná-lo lento ou, até, indisponível para os seus utilizadores. Este tipo de ataque pode ser realizado para qualquer sistema ou aplicação que possua uma ligação à internet. Assim sendo, também o sistema em análise pode ser afetado por ataques deste género, uma vez que funciona com recurso à internet.

Deste modo, é importante efetuar um caso de abuso para um *Denial of Service*, de modo a compreender melhor o seu processo e conseguir proteger o sistema.

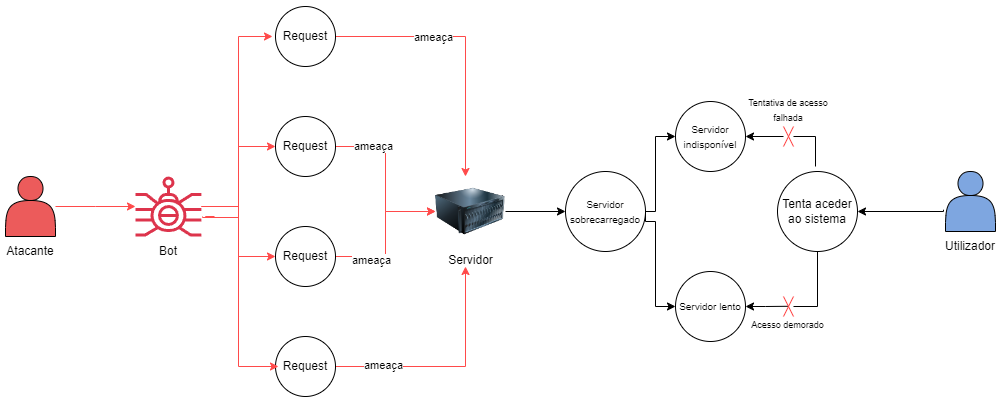

Figura 10: Caso de abuso de um *Denial of Service*

No diagrama, é apresentado um cenário hipotético no qual um atacante recorre a um *bot* ou uma rede de *bots* maliciosos para enviar enormes quantidades de pedidos para a aplicação. Como resultado, o sistema fica sobrecarregado, o que o torna mais lento ou indisponível. Assim, utilizadores legítimos que pretendam aceder à aplicação acabam por

não suceder.

Impacto:

Como resultado deste ataque, dependendo da sua duração, podem ocorrer baixas significativas na receita do negócio, uma vez que os utilizadores ficam impedidos de aceder ao site durante o ataque, não conseguindo efetuar subscrições dos planos de *streaming*. Para além disso, no caso de o ataque ser realizado durante um período de tempo considerável, a reputação do sistema poderá ficar comprometida, uma vez que os clientes poderão perder confiança na segurança do mesmo.

Mitigação:

De modo a previnir este tipo de ataque, existem algumas medidas que podem ser tomadas:

- **Implementação de *firewall*** capaz de bloquear pedidos suspeitos;
- **Adoção de um sistema de deteção de intrusão (IDS)** que detete qualquer atividade anormal e alerte o administrador do sistema;
- **Limitar a quantidade de pedidos** que um utilizador possa realizar num determinado período de tempo, impedindo que sistemas como *bots* sejam capazes de injetar uma grande quantidade de pedidos.

3.5.1 Utilização indevida da funcionalidade de *upload*

O uso indevido da funcionalidade de *upload* de músicas pelos intérpretes também pode resultar no comprometimento do sistema, no caso de se realizar o *upload* de ficheiros maliciosos. No diagrama que se segue, é apresentada de que forma o sistema pode ser atacado através desta funcionalidade.

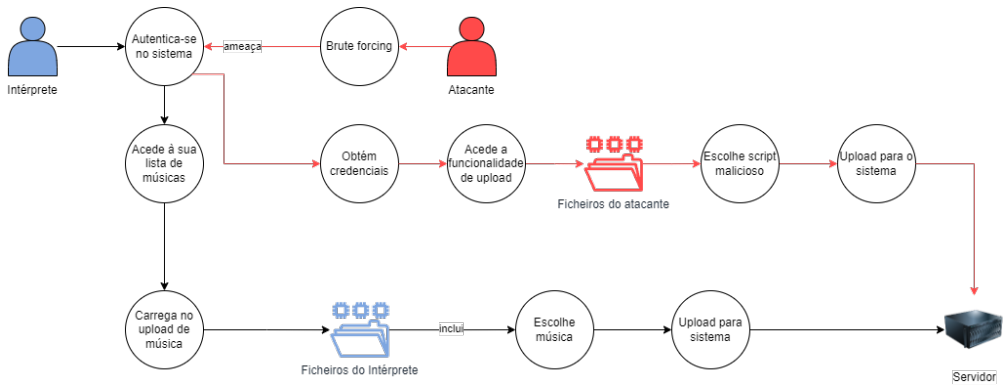

Figura 11: Caso de abuso da funcionalidade de *upload* de músicas

De forma a conseguir utilizar a funcionalidade de carregamento de músicas para a plataforma, o intérprete tem de se registar, de modo a validar os seus privilégios. No caso de um atacante conseguir obter as credenciais de autenticação de um intérprete utilizando métodos como o *brute forcing*, também obtém os privilégios necessários para realizar *uploads*, carregando *scripts* maliciosos em vez de faixas musicais.

Impacto:

Neste cenário, o atacante possui permissões para realizar o carregamento de ficheiros do seu computador para a aplicação, o que pode causar graves consequências quer na organização, quer nos seus clientes:

- **Inatividade**: à semelhança do que acontece num *Denial of Service*, os *scripts* maliciosos injetados pelo atacante na aplicação poderão resultar na sua indisponibilidade durante um intervalo de tempo indefinido (e, consequentemente, na redução da receita do negócio);

- **Perda ou manipulação de dados:** dependendo do seu conteúdo, o *script* injetado poderá aceder a dados confidenciais, como as *passwords* dos administradores e, com

elas, manipular ou remover os dados armazenados no sistema.

Mitigação: De forma a reduzir a possibilidade de ataques deste género, devem ser implementadas algumas ações como:

- **Restrição do tipo de ficheiros de podem ser carregados**: deverá ser feita uma análise do tipo de ficheiro a ser carregado e apenas permitir ficheiros do tipo ***mp3*** ou ***flac***;
- **Limitação do tamanho dos ficheiros**: deste modo evita o carregamento de ficheiros maliciosos de grande tamanho;
- **Utilização de medidas de criptografia na base de dados**, de forma a dificultar o seu acesso e a proteger os dados.