UNIVERSIDADE DO MINHO
MIEI

SEGURANÇA DE SISTEMAS INFORMÁTICOS

# TP3

**Grupo 12**
Alexandre Pinho - a82441
Luís Martins - a82298

Braga, Portugal
18 de Janeiro de 2020

# Conteúdo

# 1 Introdução

No âmbito da unidade curricular de Segurança de Sistemas Informáticos, foi proposto aos alunos que realizassem um trabalho que, com um mecanismo adicional de autorização de operações (libfuse), complementasse os mecanismos de controlo de acesso de um sistema de ficheiros tradicional do sistema operativo *Linux*.

Assim, a ideia deste mecanismo complementar passa por cada ficheiro tem uma lista de permissões sobre os utilizadores, sendo que estes apenas poderão aceder ao ficheiro, através do envio de um código por email para o utilizador.

# 2 Implementação da solução

Neste capítulo, é explicado como o grupo implementou a solução da gestão de permissões de ficheiros.

Assim, proceder-se-á à explicação do ficheiro `JSON`, a arquitetura de classes utilizada, assim como o método `open` e os seus métodos auxiliares.

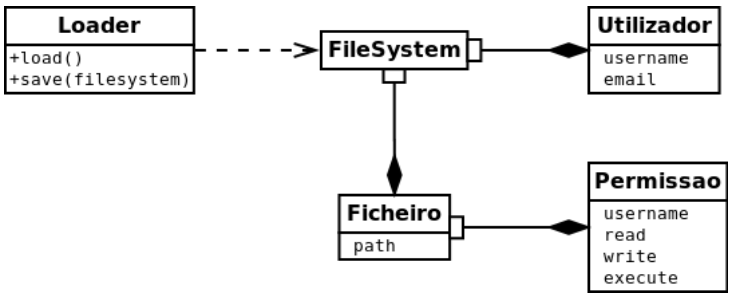


Figura 1: Arquitetura de classes da *package* `models`

## 2.1 Ficheiro JSON

Este ficheiro serve para guardar informações relativas aos utilizadores que têm acesso aos ficheiros e são guardadas informações como o nome do utilizador e respetivo e-mail. Além disso, são guardados o caminho dos ficheiros e respetivas permissões que um dado utilizador têm sobre ele (read, write, execute).

## 2.2 Arquitetura de classes

A figure 1 é um diagrama UML de classes da *package* `models`.

A classe `Loader` serve para carregar os dados do ficheiro JSON para um objeto do classe `FileSystem` e também para guardar um objeto desta classe no ficheiro JSON.

A classe `FileSystem` é a representação em memória central dos metadados do sistema de ficheiros, ou seja, os utilizadores registados e as suas permissões em relação aos vários ficheiros. Assim, é guardado um dicionário de objetos da classe `Utilizador`, cuja a chave é o nome de utilizador, e um dicionário de objetos da classe `Ficheiro` cuja chave é o caminho do ficheiro.

Na classe `Utilizador` são armazenadas as informações relativas a um utilizador, ou seja, nome do utilizador e respetivo e-mail.

A classe `Ficheiro` contém as informações relacionadas a um ficheiro, ou seja, o seu caminho um dicionário de objetos da classe `Permissao`, cuja a chave é o nome do utilizador, de maneira a saber os utilizadores que têm acesso ao ficheiro e respetivas permissões. Se um utilizador não pertencer ao dicionário de permissões, então não tem permissão de acesso ao ficheiro.

Na classe `Permissao`, são guardadas informações relativas às permissões de um dado utilizador, tal como o utilizador e respetivas permissões de *read*, *write* e *execute*.

### 2.3 Método `open` e auxiliares

Quando o utilizador acede a um ficheiro, ele causa a invocação da *system call* `open`. Para implementar o mecanismo de controlo de acesso complementar, foi alterado este método em duas maneiras: primeira são verificadas as permissões do sistema de ficheiros criado para este propósito; e depois é pedido o código de confirmação.

A primeira funcionalidade é implementada pelo método auxiliar `_check_permissions`. Este método compara as permissões que o utilizador em causa tem com as permissões que ele necessita de ter para aceder ao ficheiro no modo em que o quer aceder. Se não tiver permissões suficientes, o pedido é logo rejeitado.

O método auxiliar `_authenticate` gera uma *one-time password*, envia-a para o endereço de email associado ao utilizador (caso ele esteja registado no sistema), e abre uma janela com um campo onde o utilizador pode reproduzir essa senha. Se o utilizador introduzir uma senha incorreta, ou se não introduzir nenhuma senha em 30 segundos, então o pedido é rejeitado.

## 3 Demonstração do sistema

Neste capítulo, é apresentado como o utilizador deve interagir com o sistema construído.

### 3.1 Arranque do sistema

Para arrancar o sistema, deverá ser executado o ficheiro `passthroughs.py` e passado como argumentos, em primeiro, a pasta que servirá de base para a pasta utilizada para **MountPoint**, sendo que esta segunda pasta deverá ser utilizada como segundo argumento. Por exemplo, da seguinte maneira:

```
python3 fs.py ficheiros/ mountpoint
```

Assim, depois de executado o comando com as pastas pretendidas, está tudo pronto para começar a serem geridas as permissões sobre os ficheiros.

### 3.2 Código de Acesso

Quando o utilizador tenta aceder a um ficheiro e tem as permissões necessárias para fazer aquilo que pretende, será pedido um código de acesso (figura 2), que será recebido por email, esse que lhe foi definido, aquando do seu registo no sistema. De referir que quando passar 30 segundos, a janela de submissão do código fechar-se-á e não será dado o acesso ao ficheiro.

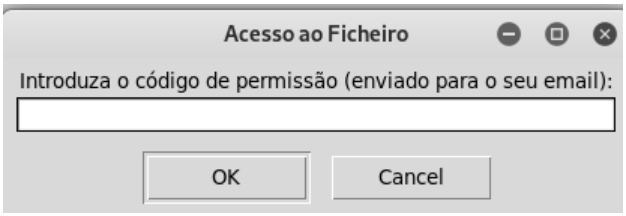


Figura 2: Pedido de submissão do código

### 3.3 Administração do sistema

Para manipular as permissões dos diferentes ficheiros e ter controlo sobre que utilizadores lhes têm acesso, podem ser utilizados três *scripts*: `add_user.py`, `rm_user.py`, e `chperm.py`.

Para adicionar um novo utilizador, ou para atualizar a informação de contacto de um utilizador já no sistema, deve ser utilizado o seguinte comando: `python3 add_user.py <username> <email>`.

Para remover um utilizador, deve ser utilizado o comando `python3 rm_user.py <username>`.

Para alterar as permissões de um utilizador a um ficheiro, deve-se verificar a existência do utilizador (em caso contrário adicionando-lo), e de seguida correr um comando com o seguinte formato: `python3 chperm.py chperm.py [+r] [-r] [+w] [-w] [+x] [-x] <username> <path>`. As flags servem para permitir ou proibir, respetivamente, a leitura, escrita ou execução (também respetivamente) do ficheiro especificado. Serão rejeitadas invocações do comando com operações contraditórias (simultaneamente retirar e adicionar uma determinada permissão).

## 4 Conclusão

Este relatório descreve mecanismo de acesso de controlo baseado em listas de acesso de controlo implementado pelo grupo e apresenta uma explicação de como utilizar o sistema implementado.

Para o sistema ter garantias de segurança, seria necessário proteger o ficheiro JSON que guarda a informação acerca das permissões dos utilizadores, e limitar o acesso aos *scripts* utilizados para administrar o sistema, talvez ao necessitar privilégios especiais para aceder aos scripts relativos à administração.

Na resolução do trabalho houve algumas dificuldades na configuração inicial do FUSE. De resto, a utilização do Python facilitou a implementação do sistema de ficheiros e das funcionalidades desejadas.