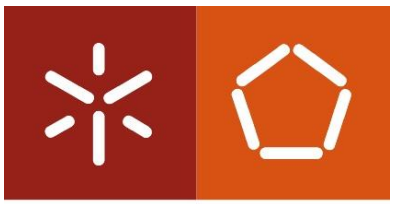

Mestrado em Engenharia Informática
Universidade do Minho

Tecnologias de Segurança

# *Deteção de Modificações Não-Autorizadas no Sistema Operativo*

**Trabalho Prático 3 - Grupo 16**


| | |
|---|---|
| pg47157 | Duarte Oliveira |
| pg47032 | António Guerra |
| pg47520 | Melânia Pereira |
| pg47554 | Paulo Pereira |


11 de junho de 2022

# Conteúdo

# 1. Introdução

Este projeto, desenvolvido no âmbito da unidade curricular de Tecnologias de Segurança, tem como objetivo a implementação de uma componente de software capaz de detetar modificações não autorizadas num conjunto de ficheiros considerados críticos para a segurança do sistema operativo.

Esta componente pode ser desenvolvida para funcionar de forma assíncrona, como serviço, ou em tempo-real, como sistema de ficheiros virtual.

Depois de alguma pesquisa e debate, o grupo decidiu seguir a abordagem do sistema de ficheiros virtual baseado em libfuse.

Ao longo deste documento será feita uma descrição do processo de desenvolvimento da componente e também a forma de instalação e utilização da mesma.

# 2. Descrição da solução

O grupo optou por implementar a solução em python, por ser uma linguagem com a qual os elementos se sentem mais à vontade.

Assim, o início do processo de desenvolvimento foi caracterizado por uma pesquisa por bibliotecas do python que implementassem o FUSE (Filesystem in Userspace), assim como exemplos de utilização e programação dessas bibliotecas, como recomendado no enunciado.

Finalmente, a biblioteca escolhida foi a llfuse, que disponibiliza um exemplo da sua utilização em passthroughfs.py.

## 2.1 Arquitetura e Estrutura

Assim, o foco central da componente desenvolvida é o ficheiro `passthroughfs.py`, que foi adaptado do exemplo disponibilizado na documentação oficial da biblioteca. Além deste programa, há ainda o `mail.py` onde são enviados emails para os utilizadores que pretendem aceder a um determinado ficheiro (desde que estejam autorizados) com o código de acesso (previamente gerado). Finalmente, é necessário um programa que faça a gestão de controlo de acesso e o grupo decidiu implementar uma réplica da técnica usada pelo Linux, ou seja, Access Control List. Para isso, recorreu a um software de bases de dados – MongoDB – onde são armazenados os dados dos utilizadores que tem acesso a cada ficheiro, desta forma, é possível fazer uma verificação de permissão sempre que um utilizador pretende aceder a um ficheiro. Para adicionar permissões, há o programa `access_control.py` que estabelece uma conexão com a base de dados para adicionar os dados necessários.

Há ainda um ficheiro de configuração muito simples, onde são definidos os utilizadores e passwords a serem utilizados para a autenticação no software MongoDB e ainda para o servidor SMTP do Gmail.

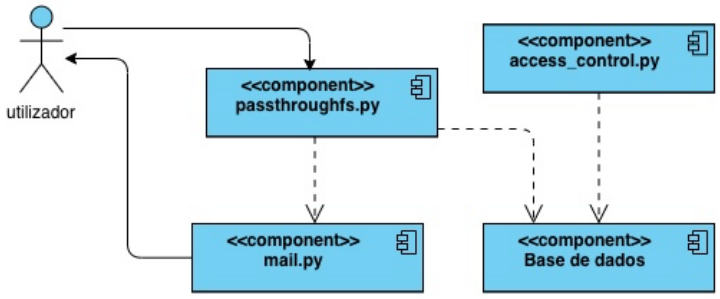


Figura 2.1: Arquitetura do sistema

### 2.1.1 passthroughfs.py

Como referido no enunciado, a interceção da chamada ao sistema `open()` ou `read()` é suficiente para detetar e permitir ou negar modificações nos ficheiros em causa.

Assim, o grupo alterou a função `open()` do exemplo fornecido pela llfuse, para que esta faça uma verificação na base de dados de que o utilizador que está a tentar abrir o ficheiro tem, realmente, as permissões necessárias para tal, se sim, procede para o envio de um código de acesso para o seu email, que é gerado aleatoriamente e que o utilizador terá que introduzir numa janela popup para que o ficheiro abra, se não, o pedido de acesso é rejeitado por falta de permissões.

O grupo decidiu ainda adicionar um parâmetro de controlo de tempo, que faz com que o utilizador, depois de inserir o código de acesso uma vez, não tenha que o voltar a fazer nos 5 minutos seguintes. Isto é um exemplo de um certo compromisso entre a segurança e a conveniência e o grupo achou que seria bom realizá-lo e implementá-lo, pois cria uma comodidade maior na utilização do programa.

### 2.1.2 access_control.py

Este programa tem como objetivo guardar e organizar os dados necessários relativos a que utilizadores tem acesso a que ficheiros, para isso, permite a inserção de novas permissões na base de dados.

Começa por verificar se esta inserção é válida, ou seja, se quem está a adicionar permissões para um determinado ficheiro é o owner desse ficheiro, e ainda, claro, se o ficheiro existe.

Depois desta verificação feita, segue-se a fase de inserção na base de dados, com a qual é feita a conexão, verificando depois se já existe uma coleção com o nome do ficheiro em causa, se não existir, esta é criada, se existir, é adicionada uma entrada a essa coleção. Os dados guardados para cada ficheiro são um objeto contendo o UID (id do utilizador que passa a ter acesso) e o seu email.

### 2.1.3 mail.py

Para implementar o envio de um código OPT por um segundo canal, o grupo optou por usar o email, mais concretamente, o servidor SMTP do Gmail.

Este programa serve para fazer a ligação ao servidor e enviar o email para o utilizador. O conteúdo do email é o código de acesso gerado no aquando da chamada de sistema open no passthroughfs.

### 2.1.4 config.ini

Neste ficheiro estão definidos os nomes de utilizador e passwords para autenticação no MongoDB e na conta Gmail de onde os emails serão enviados.

Este ficheiro deve ser preenchido pelo utilizador com os seus dados.

# 3. Segurança

É do conhecimento do grupo algumas vulnerabilidades muito comuns e conhecidas como a validação de input e buffer e integer overflow. Estas duas últimas, pela implementação do programa estar a ser feita em python, não são um problema, pois a própria linguagem tem mecanismos de controlo em relação a isso.

No entanto, a validação do input continua a ser uma vulnerabilidade que deve ser tratada. Para tal, o grupo realizou, sempre que há interação com o utilizador, uma validação para cada dado introduzido. Sendo que maior parte destes dados são caminhos de ficheiros ou diretorias, é verificada a existência dos mesmos e ainda se o utilizador tem acesso a eles e, no caso de se tratar de uma adição de permissão, se o utilizador que está a executar o programa é o dono do ficheiro.

Além disto e agora mais concretamente na adição de permissões, o email introduzido é também verificado, para confirmar que se trata de um email válido e que não contém caracteres que não podem fazer parte de um endereço de email e que poderiam ser usados para injeção de código ou outros dados na base de dados. Finalmente, é também verificado se o utilizador ao qual se está a dar acesso é realmente um utilizador existente no sistema operativo.

A maior parte das fraquezas mais comuns enumeradas no top 25 da CWE tem como base as vulnerabilidades descritas acima. No entanto há algumas que têm em conta permissões e tratamento de credenciais, como é o exemplo da *CWE-798 – Use of Hard-coded Credentials* e da *CWE-276 – Incorrect Default Permissions*. Para tentar mitigar estas fraquezas, o grupo faz uma configuração das permissões associadas ao ficheiro config.ini aquando da instalação dos programas, ou seja, no script de instalação.

Esta configuração é feita da seguinte forma:

```
chmod u+rwx config.ini
chmod g-rwx config.ini
chmod o-rwx config.ini
```

e dá permissão de acesso, escrita e execução do ficheiro config apenas ao utilizador que está a efetuar a instalação. Desta forma, nenhum outro utilizador poderá aceder às suas credenciais.

# 4. Instalação e Utilização

## 4.1 Pré requisitos

Para que o programa corra normalmente, é necessário ter instalado:

- python@3 (versão 3 do python)
- mongodb (e ter o serviço do mongo ativo)

Além disto, no caso do sistema operativo onde o programa será usado ser macOS, é necessário que o macFUSE esteja instalado e tenha todas as permissões necessárias.

É ainda necessário que o utilizador possua uma conta no serviço Gmail.

## 4.2 Instalação de dependências

Para instalar todas as dependências necessárias à execução do código, foi desenvolvido um script `install.sh` que deve ser executado num terminal, através do seguinte comando:

```
bash install.sh
```

## 4.3 Configuração da base de dados

Para ser estabelecida uma conexão com a base de dados, é necessário que um utilizador seja criado na mesma. Para isso devem seguir-se os seguintes passos:

1. abrir uma shell do mongo, escrevendo 'mongo' num terminal
2. alterar para a base de dados 'fs':

```
use fs
```

3. criar um novo utilizador (aqui damos o exemplo de um username 'root' e password 'root' que são os default no ficheiro de configuração):

```
db.createUser({
  user:"root",
  pwd:"root",
```

```
  roles:[
    {role:"readWrite",db:"fs"}
  ]
})
```

Ainda, é necessário preencher/alterar o ficheiro config.ini com os dados necessários:

- credenciais do utilizador criado na configuração da base de dados
- credenciais do utilizador Gmail do qual os emails serão enviados (para a autenticação ser bem sucedida, é necessário ter uma palavra-passe de aplicações, que pode ser obtida aqui)

## 4.4 Utilização

Finalmente, para usar o programa, deve ser aberto um terminal na pasta `dist` que foi criada na instalação. Nessa diretoria, o uso resume-se à execução dos programas para as duas seguintes opções:

- adicionar permissões

  ```
  ./access_control <caminho do ficheiro> <nome do utilizador>
  ↪  <email>
  ```

- iniciar o sistema de ficheiros virtual

  ```
  ./passthroughfs <caminho da diretoria a copiar> <caminho da
  ↪  diretoria para onde copiar>
  ```

  para abrir um ficheiro, se tiver acesso ao mesmo, será pedida a introdução de um código, previamente enviado para o email indicado aquando da adição de permissões.

# 5. Conclusão

O grupo aprendeu bastante sobre a forma como a segurança em acessos a ficheiros funciona a nível dos sistemas operativos e a realização deste trabalho foi muito enriquecedora para a consolidação de conceitos e conhecimentos adquiridos nas aulas da UC.

É de notar que, ao longo do processo de desenvolvimento, algumas dúvidas e complicações foram aparecendo, mas o grupo conseguiu sempre ver resolvidas as suas questões, o que levou a que conseguisse sempre avançar e até ir melhorando alguns aspetos da implementação.

Um grande aspeto a sublinhar é o facto de o sistema operativo macOS, para abrir um ficheiro, faz mais do que uma chamada de sistema open, o que levava a que vários códigos de acesso fossem gerados para abrir um ficheiro apenas uma vez. O grupo acredita que isto se deve ao facto de serem necessárias estas chamadas para, por exemplo, a alteração das miniaturas no design do sistema de ficheiros do sistema operativo. No entanto, o grupo conseguiu mitigar este problema com a implementação de um "período de graça"em que, como já referido neste relatório, depois de o utilizador introduzir o código correto uma vez, não necessita de o introduzir novamente durante 5 minutos. Esta solução foi dada ao grupo pelo docente da UC depois de ser abordado com o problema descrito e, apesar de este ser um problema que acontece apenas no sistema operativo macOS e não em Linux, o grupo decidiu implementar esta solução pois achou que seria uma mais valia a nível de comodidade e funcionalidade para o utilizador.

Finalmente, de sublinhar que a satisfação do grupo com o trabalho desenvolvido é muito grande.