



**Universidade do Minho**

Escola de Engenharia

Mestrado Integrado em Engenharia Informática

## **Unidade Curricular de Segurança de Sistemas Informáticos**

Ano Letivo de 2019/2020

### **Autorização de Operações ao nível do Sistema de Ficheiros**

# SSI

**A78824 Mariana Lino Lopes Costa**

**A76867 Sarah Tifany da Silva**

Janeiro, 2020

## Introdução:

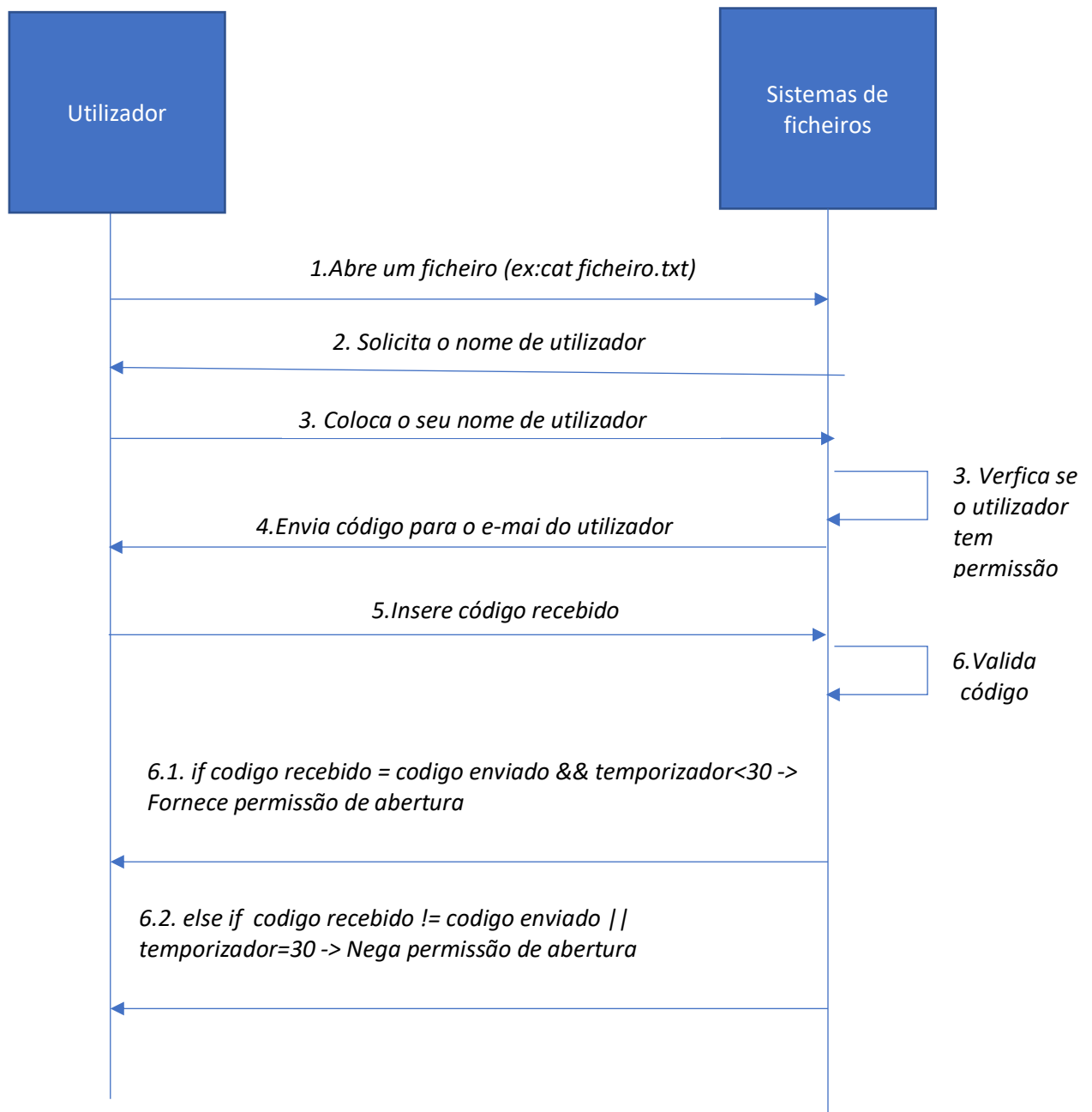
No âmbito da unidade curricular de Segurança de Sistemas Informáticos do Mestrado Integrado em Engenharia Informática da Universidade do Minho, tem como principal objetivo fazer um sistema de autorização de operações a nível de Sistema de Ficheiros.

Foi necessário implementar um mecanismo que complemente os mecanismos tradicionais de controle de acesso de um sistema de ficheiros tradicional do sistema operativo Linux/Unix com um mecanismo adicional de autorização de operações de abertura de ficheiros. Deste modo foi sugerido a utilização da biblioteca libfuse como base para todo o sistema implementado. Esta biblioteca proporciona uma API que possibilita a criação de sistemas próprios de ficheiros sem ocorrer alterações ao nível do Kernel.

Deste modo, o sistema deve ser capaz de autorizar a operação de abertura de um dado ficheiro por parte de um utilizador.

## Arquitetura da solução

O esquema apresentado demonstra o funcionamento geral de todo o programa implementado, após a tentativa de aceder a um ficheiro.

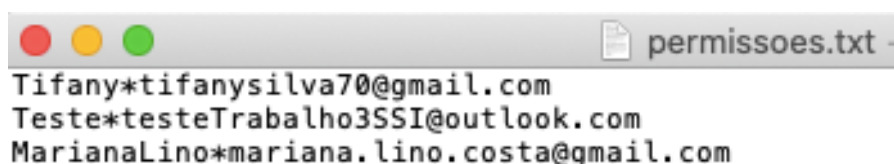


Quando o utilizador tenta aceder ao ficheiro.txt, é-lhe solicitado o nome do utilizador, este insere na linha de comandos o seu username, o programa verifica se o nome está na lista de permissões e procura o e-mail associado. Seguidamente, um código é gerado com números aleatórios e é enviado para seu e-mail. A partir deste momento o temporizador é despoletado tendo o utilizador 30 segundos para inserir o código de verificação. Se o utilizador inserir o código correto dentro do tempo limite, a permissão de abertura é concedida, caso contrário, é negada. A permissão também é negada no caso de o código de verificação ser inserido incorretamente.

## Implementação

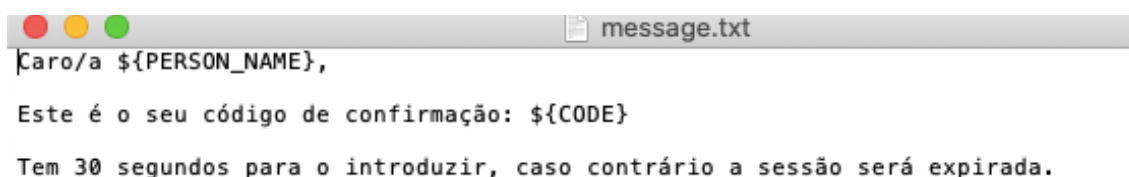
Para a elaboração do programa foi decidido se programar na linguagem *python* e usar a biblioteca do **fusety** disponibilizada pelo mesmo. Como base para a elaboração de todo o trabalho, foi utilizado o ficheiro **passthrough.py**, tendo-se acrescentado algumas funções e feito alterações de modo a cumprir com os requisitos.

Foi decidido colocar todas as permissões de acesso num ficheiro.txt (permissões.txt), que contem todos os nomes de utilizador e o seu mail correspondente. Cada linha representa um contacto e o utilizador está separado do seu e-mail pelo caracter “\*”.



```
Tifany*tifanysilva70@gmail.com
Teste*testeTrabalho3SSI@outlook.com
MarianaLino*mariana.lino.costa@gmail.com
```

Foi necessário criar também um ficheiro message.txt que contem o body do e-mail, em que a string `${PERSON_NAME}` é depois substituída pelo nome de utilizador da pessoa e o `${CODE}` pelo código gerado aleatoriamente.



```
Caro/a ${PERSON_NAME},

Este é o seu código de confirmação: ${CODE}

Tem 30 segundos para o introduzir, caso contrário a sessão será expirada.
```

A pasta Ficheiro contém o ficheiro.txt que o utilizador quer aceder e também é criada uma diretoria Montar onde vai ser montado o sistema de ficheiros.

Bastou-se alterar a função `open()` de forma a que a validação do utilizador fosse realizada sempre que este invoca a operação.

Para o envio do código de confirmação via e-mail foi utilizada a biblioteca do *python*: **smtplib**, que facilitou imenso o código e onde se utilizou o **MIMEMultipart()** e **MIMEText()** dessa biblioteca para se criar a mensagem, como está apresentado na imagem em baixo.

```
#geração do código de verificação aleatório
code = str(random.randint(000000,999999))

#criar a mensagem
mensagem = self.ler_template('message.txt')
msg = MIMEMultipart()
message = mensagem.substitute(PERSON_NAME=text, CODE=code)
msg['Subject'] = "Código de Verificação"
msg['From'] = 'testeTrabalho3SSI@gmail.com'
msg['To'] = mail
msg.attach(MIMEText(message,'plain'))

#criar o servidor e fazer o login
server = smtplib.SMTP('smtp.gmail.com',587)
server.ehlo()
server.starttls()
server.ehlo()
server.login('testeTrabalho3SSI@gmail.com','2019.20TesteSSI')
#print('fez login')
server.send_message(msg)
#server.sendmail('testeTrabalho3SSI@gmail.com',mail,msg.tostring())
#print('mail enviado')
server.quit()
```

Foi criada uma função auxiliar **getMail** para se obter o mail associado ao nome de utilizador e também uma função **ler\_Template** para conseguir ler um ficheiro texto e retornar um objeto *Template*, que vai ser necessário para o funcionamento do código do envio do e-mail.

```
#função que procura o mail associado ao utilizador na lista de permissões
def getMail(self,filename, nome):
    with open(filename, 'r') as f:
        for line in f:
            x=line.split('*')
            if(x[0] == nome):
                return x[1]

#função que transforma para ler em um arquivo de modelo
def ler_template(self,filename):
    with open(filename, 'r', encoding='utf-8') as template_file:
        conteudo = template_file.read()
    return Template(conteudo)
```

Para o controlo do tempo fez-se o import e uso do signal, uma biblioteca do Python que possibilita criar sinais e alarmes associados a um timeout de 30 segundos para a verificação do código enviado para o e-mail.

## Execução

Para compilar e executar o projeto é necessário seguir os seguintes passos:

1. Instalar o *python3*
2. Instalar o *libfuse*

**sudo pip3 install pyfuse3**

3. Criar uma diretoria que permite montar o novo sistema de Ficheiros:

**mkdir Montar**

4. na linha de comandos basta colocar o comando seguinte:

**python3 passthrough.c Ficheiro Montar**

5. Testar o programa: entrar na pasta Montar e colocar o comando **cat** **ficheiro.txt**

## Conclusão

Terminada a implementação de todo o código, passou-se a redigir as observações finais e conclusões. Em relação ao projeto, entendeu-se bem o sistema Fuse e como este pode ser utilizado para a criação de um Sistema de Ficheiros, percebeu-se o ficheiro disponibilizado `passthrough.py` e tentou-se solucionar o problema da forma mais simples possível em linguagem python.

É importante se referir que este trabalho permitiu que se desenvolvessem os conhecimentos acerca de sistema de ficheiros e mostrou o quão é importante a segurança da informação nos sistemas atuais.