

Segurança de Sistemas Informáticos

Trabalho pratico nº3

Relatório de Desenvolvimento

Pedro Gomes (a84220)

Filipa Silva (a81015)

28 de Janeiro de 2021

Contents

1	Introdução	2
1.1	O que é a libfuse?	2
1.2	Tarefas propostas	3
2	Implementação	3
2.1	Tarefa 1	3
2.2	Tarefa 2	4
2.3	Tarefa 3	4
2.4	Tarefa 4	5
3	Conclusão	6

1 Introdução

O objetivo primordial deste trabalho prático passa por complementar os mecanismos de controlo de acesso de um sistema de ficheiros tradicional do sistema operativo Linux com um mecanismo adicional de **autorização de operações de abertura de ficheiros**. Este mecanismo a desenvolver deverá ser baseado em **libfuse**.

1.1 O que é a libfuse?

A **libfuse** é uma biblioteca que existe do lado do espaço do utilizador para que este use a interface *Filesystem in Userspace* (FUSE). Esta interface de *software* para sistemas operativos **Unix**, permite que utilizadores não privilegiados criem os seus próprios sistemas de ficheiros sem editar o código do *Kernel*. Isto é obtido executando o código do sistema de ficheiros num espaço de utilizador, enquanto que o módulo FUSE estabelece uma "ponte" com as interfaces reais do *Kernel*.

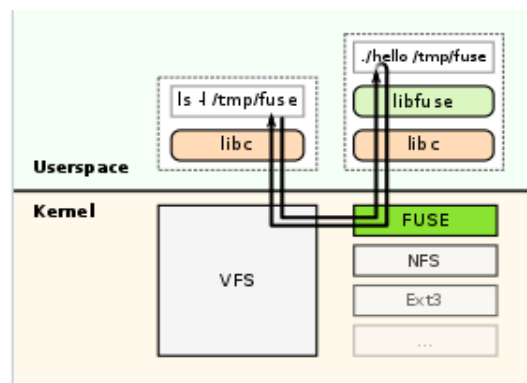


Figure 1: Diagrama de fluxo do funcionamento do FUSE.

Através do diagrama pode ter-se uma noção real daquilo que foi explicado anteriormente. Deste modo, quando há uma solicitação do espaço do utilizador para listar ficheiros (neste caso `ls -l /tmp/fuse`), esta é redirecionada para o *Kernel*, através do sistema virtual de ficheiros (VFS), para o FUSE. O FUSE executa o programa registado (`./hello`) e transmite a solicitação para esse programa que retorna uma resposta ao FUSE, sendo esta redirecionada até ao espaço do utilizador pelo caminho inverso.

1.2 Tarefas propostas

Para a concretização deste trabalho prático, há várias tarefas propostas pela equipa docente que devem ser executadas por forma a obter o resultado final pretendido. Sendo assim, dá-se uma breve noção daquilo que terá que ser feito:

1. Autorizar a operação de abertura apenas depois da introdução de um código de segurança único enviado ao utilizador que o despoletou;
2. Manter o registo de todos os utilizadores que poderão aceder ao sistema de ficheiros;
3. Ao invocar a operação `open()` de abertura de ficheiros, deve ser retornado: sucesso quando tiver recebido o código de segurança enviado ao utilizador; ou insucesso quando não corresponde ao código ou tiverem sido ultrapassados 30 segundos;
4. Criar uma interface *web* para o servidor, ao qual o utilizador irá comunicar o código de segurança recebido;

2 Implementação

Para a implementação, o grupo optou por recorrer à linguagem de programação **Python**, em que foram criados dois ficheiros distintos, um que funciona como o servidor (`server.py`) e outro (`FileSystem.py`) onde é declarada a classe `passthrough` semelhante ao que já é implementado na biblioteca *libfuse*.

2.1 Tarefa 1

O código de segurança é gerado aleatoriamente e composto por 8 dígitos e tomou-se a decisão de enviá-lo via email. Para isto, foram utilizadas bibliotecas de envio de email e de SMTP que permitem o envio de emails a partir do Google. O email é enviado para o utilizador e através de `ssitp32021@gmail.com`.

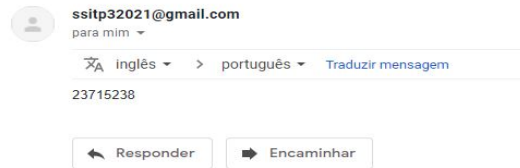


Figure 2: Exemplo do envio de um email.

2.2 Tarefa 2

Para guardar o registo de todos os utilizadores que podem aceder ao sistema de ficheiros, criamos um ficheiro chamado `users.txt` com o *username* e o email respetivo do utilizador. Quando o sistema de ficheiros é inicializado este fica com as permissões de 400, ou seja, apenas o proprietário do ficheiro o pode ler e ninguém consegue alterar o conteúdo do mesmo. Deste modo garantimos que não pode ser alterado por possíveis atacantes que queiram ganhar acesso aos ficheiros.

2.3 Tarefa 3

Ao ser invocada operação `open()` é realizada uma validação do utilizador, onde se verifica em primeiro lugar se o utilizador já foi autenticado(já inseriu o código correto anteriormente). Caso seja a primeira vez a tentar aceder a um ficheiro recentemente, é feita a validação do mesmo verificando que este existe na lista de utilizadores permitidos e enviando, no caso de pertencer, um email com um código. Se o código for inserido no servidor dentro de 30 segundos correctamente é dado acesso ao utilizador aos ficheiros durante um certo determinado tempo (variável "usertimeOut" no código), depois desse timeout expiar, o utilizador se quiser continuar a aceder aos ficheiros terá de inserir um novo código correctamente.

O tempo que nos definimos que o utilizador tem até ter de voltar a por o código novamente foi de 300 segundos devido a ser mais fácil para dar debug, num contexto real aumentávamos um pouco esse numero para a experiência do utilizador ser melhor.

Caso o utilizador não esteja na lista de utilizadores permitidos, insira um código errado ou não insira código nenhum durante os 30 segundos é devolvido um erro do tipo *Permission Denied*.

```

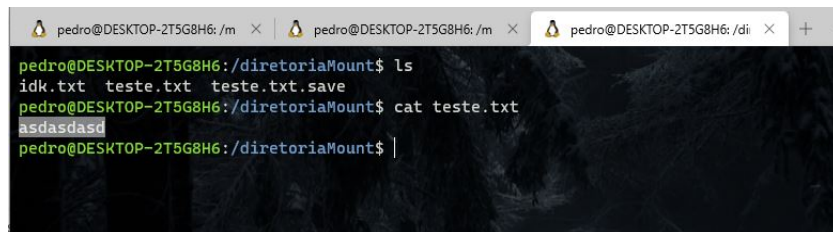
def open(self, path, flags):
    print("Pedido para Abrir ficheiro recebido")
    valido = checkUsr()
    if valido:
        full_path = self._full_path(path)
        return os.open(full_path, flags)
    else:
        raise FuseOSError(errno.EACCES)

```

2.4 Tarefa 4

O servidor que comunica com a *libfuse* não foi implementado com uma interface *web*, acedendo-se ao mesmo pelo terminal.

Através de um *socket*, é realizada a comunicação entre as duas entidades. Assim, o servidor espera por uma conexão do sistema de ficheiros, que quando acontece aguarda que o utilizador introduza o código de segurança. Após isto, o código é devolvido ao sistema de ficheiros e é encerrada a conexão. Dá-se ao utilizador a permissão de acesso ao ficheiro e o seu conteúdo é mostrado no terminal do cliente.

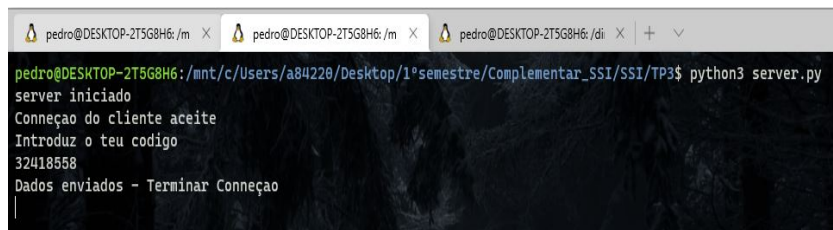


```

pedro@DESKTOP-2T5G8H6: /m  x  pedro@DESKTOP-2T5G8H6: /m  x  pedro@DESKTOP-2T5G8H6: /di  x  +  v
pedro@DESKTOP-2T5G8H6:/diretoriaMount$ ls
idk.txt  teste.txt  teste.txt.save
pedro@DESKTOP-2T5G8H6:/diretoriaMount$ cat teste.txt
asdasdasd
pedro@DESKTOP-2T5G8H6:/diretoriaMount$ |

```

Figure 3: Terminal do cliente.

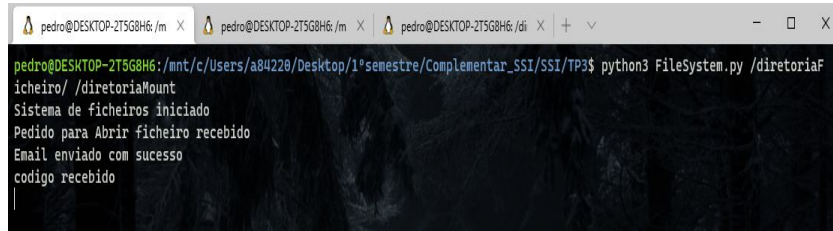


```

pedro@DESKTOP-2T5G8H6: /m  x  pedro@DESKTOP-2T5G8H6: /m  x  pedro@DESKTOP-2T5G8H6: /di  x  +  v
pedro@DESKTOP-2T5G8H6:/mnt/c/Users/a84228/Desktop/1ºsemestre/Complementar_SSI/SSI/TP3$ python3 server.py
server iniciado
Conexão do cliente aceite
Introduz o teu código
32418558
Dados enviados - Terminar Conexão
|

```

Figure 4: Terminal do servidor.



```
pedro@DESKTOP-2T5G8H6:/mnt/c/Users/a84228/Desktop/1ºsemestre/Complementar_SSI/SSI/TP3$ python3 FileSystem.py /diretoriaFicheiro/ /diretoriaMount
Sistema de ficheiros iniciado
Pedido para Abrir ficheiro recebido
Email enviado com sucesso
codigo recebido
```

Figure 5: Terminal do *FileSystem*.

3 Conclusão

A realização deste trabalho prático trouxe ao grupo algumas dificuldades, uma vez que em termos práticos a execução destes mecanismos de acesso a ficheiros não é tão simples quanto em termos teóricos.

Essas dificuldades levaram a que o trabalho não atingisse todas as tarefas propostas de uma forma exímia, no entanto, considera-se que se obteve uma apreciação satisfatória e o mecanismo adicional de autorização de operações de abertura foi implementado com sucesso.

Importa referir que os elementos do grupo adquiriram conhecimentos relativos à utilização da biblioteca *libfuse* e do sistema de ficheiros no espaço de utilizador, bem como as questões de segurança associadas ao mecanismo em questão.