



Universidade do Minho
Escola de Engenharia

UNIVERSIDADE DO MINHO

TRABALHO PRÁTICO 3#

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

Autores:

Bruno Rodrigues: pg41066

Carlos Alves pg41840

1 ÍNDICE

2	Introdução.....	3
3	Arquitetura - Funcionamento	3
4	Aplicação Web.....	3
5	Execução.....	5
6	Conclusões.....	5

2 INTRODUÇÃO

Para realizar o trabalho, foi desenvolvido um mecanismo de autorização de abertura de ficheiros, que usa a biblioteca *libfuse*, e tem como base o exemplo *passthrough_fh.c*. Para receber os códigos de segurança introduzidos pelo utilizador foi também criado um servidor muito básico que recebe o input.

3 ARQUITETURA - FUNCIONAMENTO

Primeiramente tem de ser adicionado um novo utilizador que possa aceder ao sistema de ficheiros.

Este quando é montado vai requerer a introdução de um nome de utilizador. Caso este exista, um email será enviado quando a função *open()* for chamada, quando não exista, a operação vai ser abortada, porque como o utilizador não existe, o email vai falhar ao ser enviado.

Quando o sistema de ficheiros é montado, este fica associado apenas a um utilizador, logo, alguém que tente aceder ao sistema montado só o poderá fazer com o código de segurança que foi enviado para o utilizador associado

Quando o utilizador invoca a função *open()*, isto é, tenta aceder a algum ficheiro do sistema, vai ser aberto a pagina web para a introdução do código de segurança, sendo que um email com este código também é enviado. O utilizador terá trinta segundos para introduzir o código. Se for o código correto, a operação é permitida, caso contrário, falha.

4 APLICAÇÃO WEB

A aplicação web funciona de forma simples. Criamos uma página com *html*, que tem o seguinte aspeto:

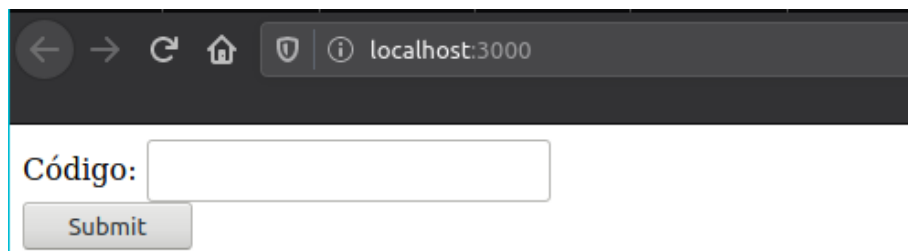
A screenshot of a web browser window. The address bar shows 'localhost:3000'. Below the address bar, there is a label 'Código:' followed by a text input field. Below the input field is a 'Submit' button.

Figura 1 – Página Web

Quando o utilizador introduz input e submete, vai ser criado um ficheiro na diretoria `/tmp/`, chamado `"codigo.txt"`, ficheiro que mais tarde vai ser lido para obtenção do código de segurança.

Todas as modificações necessárias ao exemplo `passthrough_fh.c` para realizar este sistema de segurança foram feitas à função `xmp_open()`, que é responsável por permitir a abertura de ficheiros. As alterações foram as seguintes:

Introduzimos uma função que envia o email (`SendMail`) com recurso a biblioteca ***libcurl***. Para isto utilizamos um código exemplo que a biblioteca providencia, com pequenas adaptações para acomodar as necessidades. Uma função (`system`) para abrir a página web. A função `gen_random()` vai gerar uma *string* com números e letras aleatórias, com tamanho `"size"`.

Um `"alarm(30)"`, que vai esperar 30 segundos, e quando estes passarem, vai sinalizar a função `handle_alarm`, que vai tornar o `"time_out"` **true**. Quando isto acontece, o utilizador já não vai poder introduzir o código de segurança, e terá de repetir novamente as ações necessárias.

Enquanto estes 30 segundos não passam, a função vai ficar a espera que haja texto no ficheiro `"codigo.txt"`, que é criado quando o utilizador insere o código de segurança na página web.

```
code = gen_random(size);
SendMail(supermail,code);
system("xdg-open http://localhost:3000/");
alarm(30);
while(!time_out && recievedcode[0] == '\0'){
    if( access( "/tmp/codigo.txt", F_OK ) != -1 ) {
        FILE *file = fopen ("/tmp/codigo.txt", "r+");
        fscanf(file, "%s", recievedcode);
        fclose(file);
    }else{// file doesn't exist
    }
}

if(!time_out && strcmp(code,recievedcode) == 0) {
    time_out = 0;
    recievedcode[0] = '\0';
    fd = open(path, fi->flags);
    if (fd == -1)
        return -errno;
    fi->fh = fd;
    return 0; }
else { time_out = 0;
    recievedcode[0] = '\0';
    return -1;
}
```

Figura 2 – Excerto da função `xmp_open()`.

5 EXECUÇÃO

Para executar o código:

- Primeiramente é necessário adicionar um utilizador com email valido:
 - Na pasta FUSE, executar o script adduser.py:


```
python3 adduser.py
```
- De seguida, basta voltar para à pasta TP3TS e executar no terminal:
 - **make**
- Por fim, será necessário introduzir o nome de utilizador adicionado anteriormente quando input for pedido.

Se quiser fazer tudo individualmente:

- Na pasta JavaScript: (necessário **nodejs**)
 - **node index.js**
 - Na pasta FUSE: (necessário **libcurl** e **libfuse**)
 - Compilar o sistema de ficheiros
 - ***gcc -Wall openwfuse.c `pkg-config libcurl fuse3 --cflags --libs` -o teste***
- De seguida executa o ficheiro:
 - **./teste FileSys**
- E para testar:
 - **cat FileSys/etc/passwd**

Caso tenha de repetir o processo de compilar e montar o sistema de ficheiros, não esquecer de primeiramente desmontar o sistema:

- **fusermount -uz FileSys**

6 CONCLUSÕES

Como o nosso conhecimento de *javascript* e *html* é limitado a página web resultante ficou bastante simples, e também acontece que quando o usuário introduz o código e submete, a página permanece num estado de *refresh* infinito, mas o código fica submetido, e a página tem de ser fechada manualmente. Para concluir, realçamos que foram implementadas todas(quase) as funcionalidades propostas no enunciado. É igualmente importante referir que este trabalho permitiu-nos obter conhecimento entre os assuntos dos sistemas de ficheiros e ainda na aproximação com a biblioteca **libfuse**.