Universidade do Minho
Mestrado Integrado em Engenharia Informática
SSI-Trabalho Prático 3
Grupo 1

João Bernardo Freitas a74814
Rui Pereira pg42853

January 23, 2021

1 Introdução

Na realização do trabalho prático 3 de Segurança de Sistemas Informáticos é proposto a criação de um mecanismo de controlo de acesso a ficheiros. Para a sua realização será usada a biblioteca *libfuse*.

Neste trabalho, o objetivo é a criação de uma ferramenta que consiga gerir acessos a um filesystem. Sempre que um utilizador tente abrir um ficheiro, a ferramenta irá autenticá-lo através de um código, enviado via email, sendo que este só será válido durante um intervalo de 30 segundos, caso seja validado, o utilizador terá acesso aos ficheiros.

2 Trabalho desenvolvido

Antes de iniciar o desenvolvimento do projeto em si, foi necessário saber que *syscalls* é que são executadas quando certas ferramentas são utilizadas. Dessa forma podemos perceber qual o processo de abertura de ficheiros em *UNIX*.

Para tal, utilizamos a ferramenta ***strace*** que lista as *syscalls* que uma ferramenta executa ao ser utilizada. Tendo em conta o enunciado fornecido pelos docentes decidimos utilizar a opção **-e** para filtrar o *output* e desta forma encontrar ferramentas que usem a *syscall* **open()** ou **openat()**.

```
openat(AT_FDCWD, "teste.txt", O_RDONLY) = 3
```

Listing 1: strace -e openat cat <ficheiro>

Como a biblioteca *libfuse* invoca as suas próprias funções em vez das *syscalls* do Sistema Operativo temos que alterar a função correspondente à *syscall* **open()**, nomeadamente a função *xmp_open*.

Esta alteração consiste na introdução da função *getAuth()* que irá autenticar o utilizador.

```
static int xmp_open(const char *path, struct fuse_file_info *fi)
{
        int res;
        int auth;
        auth = getAuth();
        if (auth == -1)
        {
                return -1;
        }
        res = open(path, fi->flags);
        if (res == -1)
                return -errno;
        fi->fh = res;
        return 0;
}
```

Listing 2: xmp_open modificado

A seguinte estrutura de dados foi criada visto que é necessário manter um registo de todos os utilizadores que podem aceder ao sistema. Esta estrutura de dados é inicializada pela função *main* da ferramenta **passthrough**.

```
typedef struct dadosUtilizador
{
    char *username;
    char *email;
} * utilizador;

utilizador *listaUsers;

int NumUsers = 0;
```

Listing 3: Estrutura de dados dos utilizadores

A função **getAuth()** começa por obter o **username Ubuntu** do utilizador, de seguida, verifica se esse **username** já está registado no sistema, no caso de não estar então registado pede o email ao utilizador e adiciona esse utilizador ao registo. Por fim envia um email contendo um código alfanúmerica ao utilizador através da ferramenta *sendEmail*, esse código só é válido por 30 segundos. Se o utilizador inseriu o código correto então tem acesso ao ficheiro.

2.1 Makefile

Para facilitar a execução da ferramenta e limpeza dos ficheiros foi criada uma **Makefile** com 3 funcionalidades:

- Compila a ferramenta e cria as diretorias
- Executa a ferramenta
- Limpa todos os ficheiros/diretorias criadas pelas duas funcionalidades anteriores

```
build: passthrough.c permissoes.c
	@echo "A compilar"
	gcc -Wall passthrough.c permissoes.c `pkg-config fuse3
	    --cflags --libs` -o passthrough
	make mount
	@echo "Fim da compilacao"
mount:
	@echo "Criando Diretoria"
	@echo "Conteudo do ficheiro" > ~/teste.txt
	sudo mkdir /mnt/SSI/
	sudo chmod 777 /mnt/SSI/
	@echo "Diretoria criada"
exec:
	@echo "Executando"
	./passthrough -f /mnt/SSI/
clean:
	rm -f passthrough
	if mount | grep /mnt/SSI > /dev/null;
	    then sudo umount /mnt/SSI/; fi
	sudo rm -r /mnt/SSI/
	sudo rm ~/teste.txt
```

Listing 4: Conteúdo da Makefile

3 Segurança

Para precaver ataques relacionados com **Buffer Overflows**, como por exemplo o *CWE-120*, a ferramenta recebe *inputs* do utilizador através da função *fgets* que nos permite indicar o tamanho máximo da *string* a ser copiada. Também por esta razão decidimos então evitar a utilização da função *strcpy* visto que esta não tem proteções contra *buffer overflows*.

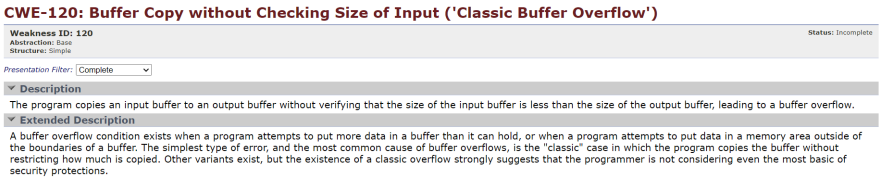

Figure 1: CWE-120

Para evitar que o código de acesso seja lido por atacantes, utilizamos a opção **TLS** que irá encriptar o email.

4 Resultados

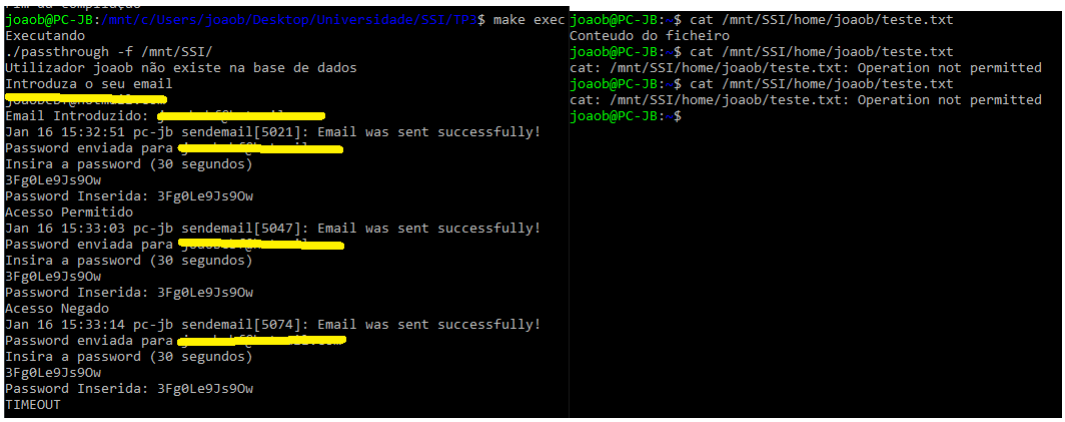

Figure 2: Ferramenta a executar

Na figura 2 temos 3 tentativas diferentes de abrir o ficheiro **teste.txt**.

- Na primeira o utilizador inseriu o código correto, como tal conseguiu abrir o ficheiro **teste.txt**
- Na segunda o utilizador inseriu o código incorreto, como tal não foi possível abrir o ficheiro.
- Na terceira o utilizador deixou os 30 segundos passar antes de inserir o código, como tal não foi possível abrir o ficheiro

Também se pode observar que na primeira tentativa o utilizador não existia na base de dados sendo então pedido um *email* para este mesmo ser registado. Nas tentativas seguintes não foi pedido email porque o utilizador já estava registado.

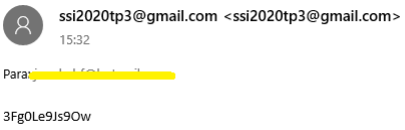

Figure 3: Email com o código enviado pela ferramenta

5 Conclusão e trabalho futuro

Após a realização deste traballho prático foi possível entender um pouco melhor os controlos de acesso a sistemas de ficheiros, conseguir perceber como entregar correspondência eletrónica através do linha de comandos do *Ubuntu* e ainda a utilização da biblioteca *libfuse*.

Uma possível melhoria deste projeto seria através da utilização de *docker containers* de forma a permitir a simulação da utilização do *filesystem* num servidor web. Para além disso o presente método para gerir os utilizadores podia ser alterado para um ficheiro em vez de estar em memória.