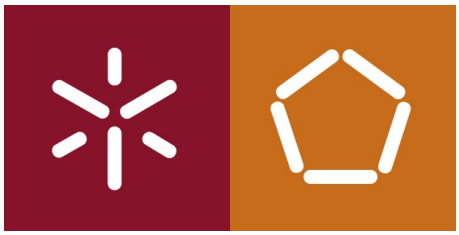

Universidade do Minho - Escola de Engenharia

Mestrado Integrado em Engenharia Informática

Tecnologia de Segurança

TP3

Autorização de Operações ao nível do Sistema de Ficheiros

Autores :
Daniel Maia (A77531)

Diogo Costa(A78034)

Versão 1.0
16 de Janeiro de 2019

1 Introdução

Os sistemas operativos baseados em Unix disponibilizam um conjunto de mecanismos de controlo de acesso baseado em permissões para manter a segurança do seu sistema de ficheiros (FS). Ainda assim, é por vezes necessária uma camada de proteção adicional de sobre os mecanismos disponibilizados. Uma tal medida adicional trata-se da geração de um sistema de ficheiros idêntico ao que se pretende disponibilizar que é depois controlada através de mecanismos externos.

O objetivo deste trabalho prático é de implementar um mecanismo de autorização de operações de leitura de ficheiros baseado na interface `libfuse`. Esta é composta por duas componentes; o módulo *kernel fuse* e a biblioteca de código *fuse*. Utilizando-a como ponto de partida, pretende-se construir um sistema conceda acesso a um dado ficheiro a um utilizador apenas após à introdução de um código de segurança gerado no momento e enviado ao utilizador através de correio eletrónico. Para tal, será mantido um registo dos utilizadores do sistema e respetivos endereços de email. A execução do login e a inserção do código serão feitas através de um servidor web que comunica com o sistema.

No final, será feita uma retrospetiva do trabalho efetuado, dos resultados obtidos e do trabalho futuro com *libfuse*.

2 Estrutura da Solução

A estrutura do projeto baseou-se em componentes programados na linguagem C e em Python. Por um lado, tem-se a porção relacionada com o `libfuse` que foi desenvolvida em C. Por outro lado, as componentes que envolveram servidores web e tratamento de emails foram realizadas em Python recorrendo à *framework* de desenvolvimento web **Flask**. Cada uma destas é iniciada quando requerida pelo programa *fuse* no momento em que é feita uma tentativa de leitura de um ficheiro do sistema de ficheiros, ocupando o endereço `http://127.0.0.1:5000/`.

Desta forma, o projeto segue o seguinte *workflow*:

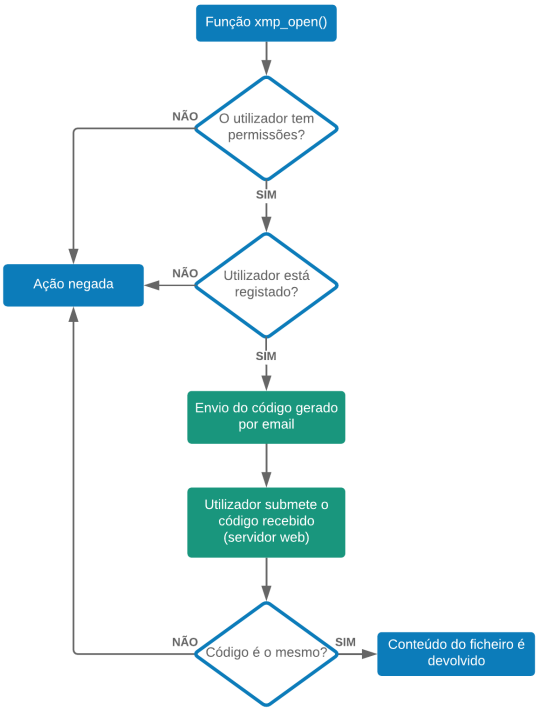

Figura 1: *Workflow* da solução.

Tudo o processo se encontra incorporado na função xmp_{open} do FUSE. Assim sendo, a primeira etapa passa por averiguar se o utilizador que está a tentar abrir o ficheiro tem permissões para o fazer. Esta parte é realizada pelo FUSE, pois o sistema de ficheiros é montado com a *flag* `default_permissions`. De salientar que ao montar o sistema de ficheiros foi usada a *flag* `auto_unmount` que permite desmontar o sistema quando este é desligado, a *flag* `-f` para indicar a diretoria onde o FS deve ser montado, a *flag* `subdir=$HOME` que indica o conteúdo do FS e a *flag* `allow_other` que permite que utilizadores, que não sejam o que montou o FS, possam ter acesso aos ficheiros. No entanto, para esta última característica funcionar é preciso descomentar a linha `user_allow_other` no ficheiro `fuse.conf` do sistema.

```
./passthrough -omodules=subdir,subdir=$HOME -o default_permissions \
              -o allow_other -o auto_unmount -f mnt/
```

De seguida, é verificado se o utilizador já se encontra registado no sistema de ficheiros, isto é, se já existe uma entrada no ficheiro `database.txt` que indica qual o email do utilizador em questão. Se esta existir, será composta pelo `uid` do utilizador, conseguido através da estrutura `fuse_context`, que o FUSE disponibiliza para perceber quem realizou a operação no sistema de ficheiros.

```
int uid = fuse_get_context()->uid;
```

Mais ainda, a informação do utilizador no ficheiro `database.txt` é complementada com o email para o qual deve ser enviada a palavra-chave gerada.

```
uid::email
```

Assim sendo, caso o utilizador ainda não esteja registado então o FUSE retorna o erro "Permissão negada".

```
if (!exists) {
        return -EACCES;
}
```

De facto, neste ponto da execução é enviado o código para o email do utilizador em questão que se encontra registado. Esta ação foi implementada com a ajuda de um *script* em Python, que recebe o `uid` do recetor do email, procura o email associado no ficheiro `database.txt`, envia o email com o código gerado e, posteriormente, envia-o também ao FUSE para que este possa fazer a comparação mais à frente.

```
context = ssl.create_default_context()
with smtplib.SMTP(smtp_server, port) as server:
    server.ehlo()  # Can be omitted
    server.starttls(context=context)
    server.ehlo()  # Can be omitted
    server.login(sender_email, password)
    server.sendmail(sender_email, [to_addrs], message)
```

O código é composto por um número variável de carateres, entre 10 e 20 (inclusive), sendo que estes podem ser compostos por letras (maiúsculas e minúsculas) e os dígitos de 0 a 9.

```
n = randrange(10, 21)
newpass = ''.join(random.SystemRandom().choice(string.ascii_lowercase +
                                               string.ascii_uppercase +
                                               string.digits) for _ in range(n))
```

Posteriormente, o FUSE liga o servidor web responsável por recolher a palavra-chave do utilizador e devolver para o output.

```
strcpy(cmd, "cd ");
strcat(cmd, homedir);
strcat(cmd, "/.web-server-pass; flask run 2> /dev/null");
fp = popen(cmd, "r");
while (fgets(try, pass_size, fp) != NULL) {
    if (try[1] != '*') {
                            break;
                }
}
```

```
if request.method == 'POST':
    if request.form['password']:
        print(request.form['password'])
        shutdown_server()
        return render_template('done.html')
return render_template('index.html')
```

Desta forma, após todos estes passos o FUSE tem na variável `pass` a *password* válida e na variável `try` a palavra-chave introduzida pelo utilizador. Neste momento basta comparar a duas. Caso sejam iguais então o conteúdo do ficheiro pode ser devolvido ao utilizador. Contudo, se forem diferentes então é devolvido o erro `-EACCES`, ou seja, acesso negado e o utilizador não consegue observar o conteúdo do ficheiro.

```
if (pass != NULL && try != NULL && strcmp(pass, try) == 0) {
          fi->fh = res;
          return 0;
}
return -EACCES;
```

2.0.1 Bibliotecas utilizadas

Para desenvolver o sistema descrito, recorreu-se predominantemente a duas bibliotecas Python; as bibliotecas `flask` e `python-dotenv`.

A biblioteca `flask` é responsável em grande parte pela implementação do servidor web do sistema. É graças a esta que pode ser transmitido uma interface *user friendly* através da qual é transmitido o código de segurança do utilizador. Esta pode ser obtida através do comando `pip`:

```
python3.7 -m pip install flask
```

A biblioteca `python-dotenv` é um auxiliar que permite a leitura de ficheiros `.env` que contém as variáveis de ambiente do sistema e a utilização dos seus conteúdos de modo a produzir código mais simples, legível e personalizável de acordo com a máquina em que este corre. Esta obtém-se através do comando:

```
python3.7 -m pip install python-dotenv
```

3 Conclusões e Trabalho Futuro

A segurança da implementação depende do acesso ao ficheiro `database.txt`, uma vez que é neste que se encontram os utilizadores que têm acesso ao conteúdo do FS. Dado que este apenas pode ser modificado pelo administrador, isto é, pelo utilizador que montou o FS, não existe então a possibilidade de terceiros alterarem o conteúdo do `database.txt` e, consequentemente, acederem aos ficheiros no FS.

A título de trabalho futuro, propõe-se a implementação de uma forma mais cómoda de gerir os utilizadores do FS. Para o efeito poderia-se utilizar o *browser* através de um servidor web como os que foram utilizados neste projeto.

Por fim, é de referir que o resultado final implementa todas as funcionalidades necessárias para permitir que o acesso ao sistema de ficheiros seja controlado, de forma a que só através do envio de códigos por correio eletrónico a um dado utilizador é que este pode interagir com o sistema de ficheiros.