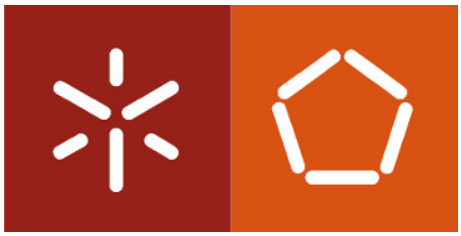
Tecnologia de Segurança

28 de Janeiro de 2020

Trabalho Prático 3 - Grupo 7

a83899	André Morais
a84485	Tiago Magalhães

Autorizacção de Operações ao nível do Sistema de Ficheiros



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1 Introdução

Foi nos proposto o desenvolvimento de um mecanismo 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, baseado em *Libfuse*. Sendo este mecanismo caracterizado por autorizar a operação de abertura apenas depois da introdução de um código de segurança único enviado ao utilizador que a despoletou. O código de segurança é enviado via correio electrónico.

O nosso relatório vai estar divido em 6 secções: a concepção da resolução, o Fuse System , o server, a config , a segurançs a ter e o como instalar. Mais à frente veremos a importância de cada um, e o papel que cada um desempenha.

2 Concepção/Desenho da resolução

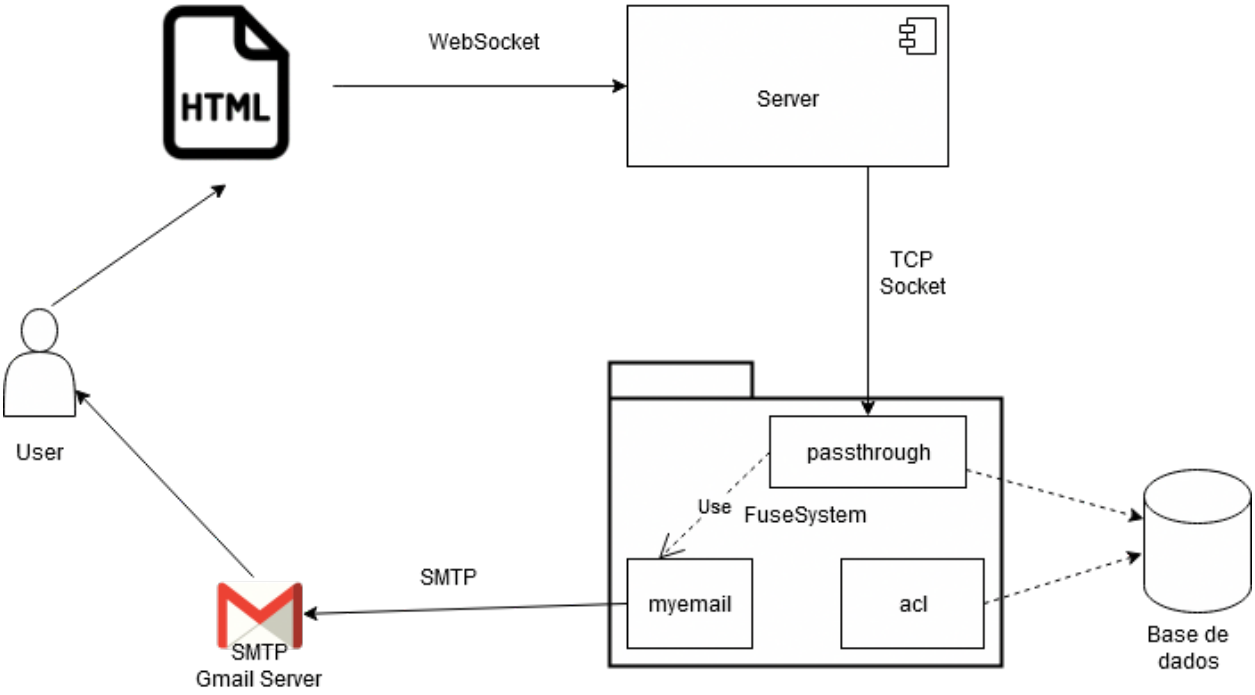


Figura 1: Arquitetura do sistema

2.1 Descrição da arquitetura

Através do modelo do sistema podemos observar que existem 3 componentes principais sendo estas:

- **Servidor**: É um servidor que é responsável por receber o código introduzido pelo utilizador numa página de *html* com *javascript* estática através de um *websocket* e após isto comunicar à classe *passthrough* o código introduzido através de um *socket*.
- **Fusesystem**: Componente na qual será tratada a operação de abertura de um ficheiro e o envio de um email com o código de acesso, também é responsável de definir para cada ficheiro que será replicado num sistema de ficheiros FUSE o utilizador que terá acesso ao ficheiro e o seu email de contacto.
- **Base de dados**: Base de dados responsável por guardar para cada ficheiro os uid's dos utilizador e o seu emails que têm permissão para aceder ao ficheiro.

2.2 Tecnologias utilizadas

As tecnologias utilizadas foram *mongo* para a base de dados, uma vez que não existem estruturas de dados relacionadas, pretendemos apenas guardar coleções de ficheiros onde em

cada coleção se encontram utilizadores que podem aceder e sua forma de contacto(email) e *websockets* que permitem comunicação entre o navegador do utilizador e um servidor.

3 Fuse System

3.1 passthrough.py

Nesta classe foi utilizado como exemplo o *passthroughfs.py* disponível no *github* do *pyfuse3* [1], para isso introduzimos modificações na operação `open` onde se faz uma *query* para verificar se o uid do utilizador se encontra presente, caso se encontre presente irá retirar o seu email e enviar no corpo deste o código de acesso, após isto irá esperar que o servidor envie o código, através da conexão de um *socket* num tempo de 30 segundos. Para geração de um código foi utilizado um UUID.

3.2 myemail.py

Módulo utilizado pelo *passthrough.py*, cuja função é gerar um email em que o conteúdo da mensagem é o código de acesso para depois o utilizador poder inserir na nossa página html e garantir o acesso a este sistema de ficheiros. Foi usado o protocolo SMTP para comunicar com os servidores da Google, ou seja, o email presente no ficheiro de configuração tem de ser obrigatoriamente **gmail.com**.

3.3 acl.py

No *acl.py* tentamos replicar o comportamento de uma ACL(Lista de controlo de acessos), sendo na nossa implementação discricionário, na medida em que o utilizador dono do ficheiro é que pode definir o utilizador e email correspondente que terá acesso ao ficheiro que se tenha intenção de replicar no sistema de ficheiros FUSE.

3.4 index.html

Uma simples página HTML estática gerada, para ter acesso a operações de através do código enviado para o email, o código introduzido pelo utilizador será enviado por um **websocket** para o servidor que terá responsabilidade de o redirecionar o código para o *fusesystem*

Figura 2: Página HTML

4 Server

Esta componente é responsável por garantir a comunicação entre o utilizador através de um *browser* o *fusesystem*.

5 Configs

Dentro do ficheiro **config.data** existem 3 secções, sendo estas:

- **Mongo**: Contém o Mongo User e a sua respetiva password
- **GmailUser**: Contém as credenciais Gmail e a sua respetiva password
- **SocketTCP**: Para definir o host e a port do Socket

Exemplo sintaxe:

```
[Mongo]
mongo_user = ...
mongo_password = ...

[GmailUser]
gmail_user = ...
gmail_password = ...

[SocketTCP]
host = ...
port = ...
```

6 Segurança

Quanto à segurança tivemos em atenção o *CWE-798: Uso de credenciais Hard-coded*, uma vez que lidamos com credencias de email e de base dados, assim colocamos as credenciais num ficheiro de configuração, de forma a evitarmos o *CWE-256: Não proteção no armazenamento de credenciais* no ficheiro de instalação restringimos o seu acesso apenas ao utilizador que está a fazer a instalação e também para evitar *CWE-276: Permissões padrão incorretas*, como podemos observar:

```
# Set config permissions
os.chmod("src/configs/config.data", S_IRUSR | S_IWUSR | S_IXUSR)
os.chmod("src/configs", S_IRUSR | S_IWUSR | S_IXUSR)
```

No ficheiro *acl.py*, uma vez que existe interação entre o utilizador, existe uma validação de *input*, de forma a verificar se o utilizador inserido existe, bem como o ficheiro e se o utilizador que está a executar é o responsável pelo ficheiro.

7 Instalação

Para informação mais detalhada encontra-se um ficheiro README.

- Primeiramente, é necessário instalar o **pyinstaller**

```
pip install pyinstaller
```

- Correr script **install** para gerar directorias com certas permissões e executáveis

```
python3 install.py
```

7.1 Configurar Database

- Criar um user no mongo com permissões para aceder à Base de Dados do filesystem exemplo:

```
db.createUser({
  user: "root",
  pwd: "root",
  roles: [
    { role: "readWrite", db: "filesystem" }
  ]
})
```

- Restart mongo
- Adicionar as credenciais mongo e gmail em ./src/configs/config.data, bem como host e port do socket.

7.2 Uso

- Depois de instalado na diretoria dist/, encontram-se os executáveis.

```
cd ./dist
```

- Definir os utilizadores que têm acesso a um ficheiro e o seu email:

```
./acl < path_ficheiro > < username > < useremail >
```

- Servidor:

```
./server
```

- Mount a filesystem:

```
./passthrough < diretoria_a_replicar > < mount_directory >
```

8 Conclusão

Quanto a dificuldades, tivemos algumas ao gerar executáveis através de código *python*, uma vez que não tínhamos experiência e foi necessário adicionar algumas funções de modo a identificar ficheiros quando se encontram no formato executável, a nível de melhorias na base de dados apenas guardamos o nome do ficheiro, assim existe uma limitação se existirem ficheiros com o mesmo nome em diretorias diferentes.

Referências

[1] Pyfuse3 example.Acedido em Janeiro 2021, em:
`https://github.com/libfuse/pyfuse3/blob/master/examples/passthroughfs.py`