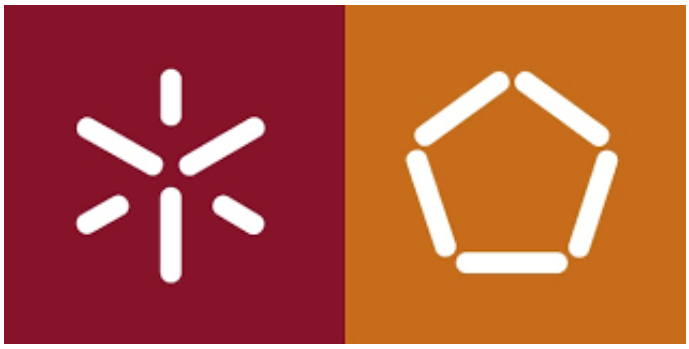

UNIVERSIDADE DO MINHO

MESTRADO EM ENGENHARIA INFORMÁTICA

Tecnologias de Segurança

TRABALHO PRÁTICO 3

Francisco Felícia Correia Pinto (PG46531)

João Esteves Gonçalves (PG46535)

Sara Cristina Freitas Queirós (PG47661)

1 Introdução

Neste trabalho, foi-nos proposta a implementação de uma componente de software que fosse capaz de detetar alterações em ficheiros críticos de um sistema Linux, através de duas opções: a utilização de um serviço assíncrono com uma verificação periódica baseada na infraestrutura de gestão de log's *rsyslog*, ou com base na criação de um sistema de ficheiros virtual que nos permite o controlo dos acessos.
Para tal, o nosso grupo decidiu optar pela opção de utilização de um sistema de ficheiros virtual, recorrendo para isso à utilização de *Libfuse*, que é uma interface que permite a exportação do nosso sistema de ficheiros para o Kernel Linux e sua posterior edição/configuração.

2 Arquitetura

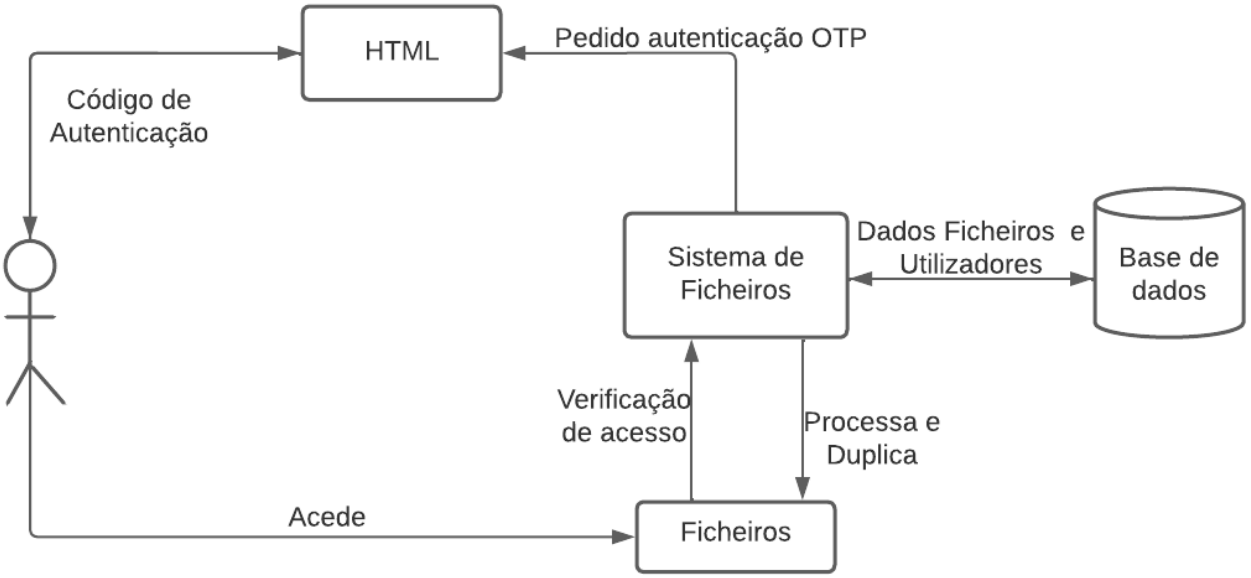

Figura 1: Arquitetura do Sistema

De forma a desenvolver um programa capaz de responder aos requisitos necessários, começamos por definir a arquitetura da aplicação.
Assim, podemos distinguir 5 componentes principais do sistema:

1. **Utilizador**: Tenta aceder a ficheiros e em função das permissões existentes para tal, pode ser requerido um código OTP para autenticação 2FA.
2. **Sistema de Ficheiros**: Trata de toda a gestão de duplicação e permissões fornecidas aos ficheiros para cada utilizador.
3. **Ficheiros**: Tratam-se dos ficheiros duplicados aos quais o utilizador tentará aceder.
4. **HTML**: página web a partir da qual o utilizador insere o código OTP para verificação de identidade.
5. **Base de Dados**: Recursos utilizados para registar os dados relativos aos ficheiros e utilizadores, relevantes para a aplicação.

3 Estrutura da solução desenvolvida

O sistema de ficheiros foi desenvolvido tendo por base um passthrough para montar os ficheiros na pasta que se pretende ”simular”. Assim que este é iniciado, as informações relevantes de todos os ficheiros considerados críticos são obtidas e guardadas na Base de Dados. Desta forma, o sistema está pronto para utilização.
Neste ponto, é possível que o utilizador dono do ficheiro forneça permissões a outros utilizadores face aos seus ficheiros, com recurso ao ficheiro ACL.py.
Para além disso, é possível tentar aceder a qualquer ficheiro na diretoria montada, ao qual o programa identificará se o utilizador tem permissão para tal.
Caso o utilizador possua permissão (mas não seja o criador do ficheiro), deverá inserir o código OTP gerado pelo Google Authenticator, em função do segredo pré-estipulado e guardado na Base de Dados, para finalizar o processo de acesso ao ficheiro.

4 Tecnologias Utilizadas

4.1 Bibliotecas

Para a implementação de todo o nosso sistema, foram usadas as seguintes bibliotecas:

- **fusepy** - Versão da biblioteca *Libfuse* em python que permite a implementação do sistema de ficheiros virtual;
- **pyotp** - Biblioteca **pyotp** que permite a implementação de código OTP e a integração juntamente com Google Authenticator;
- **MongoDB** - Base de Dados utilizada para gerir e guardar as informações relativas a ficheiros e utilizadores;
- **Flask** - Biblioteca utilizada para apresentar a página HTML ao utilizador

A instalação destas bibliotecas deverão ser feitas na root. A instalação das bibliotecas foi feita da seguinte forma:

```
#fusepy
pip install fusepy

#pyotp
pip install pyotp

#Base de Dados Mongo
pip install mongo

#Flask
pip install flask
```

Cada um dos comandos encontra-se no ficheiro *setup.py*, juntamente com outras devidas configurações de instalação permitindo assim uma automatização de todo o processo inicial de configuração e instalação.

4.2 Ficheiros

4.2.1 Passthrough.py

Este é o ficheiro responsável pela montagem de todo o sistema de ficheiros virtual e de toda a configuração do mesmo. Esta classe baseia-se na implementação da classe *Operations* que retrata todas as funções que permitem a configuração do sistema de ficheiros, implementando estas e permitindo a sua alteração de forma a que possamos configurar o nosso sistema da forma que pretendemos.

4.2.2 ACL.py

Ficheiro que funciona como uma ”access control list”que permite adicionar, apenas ao criador do ficheiro, fornecer permissões para outros utilizadores acederem ao ficheiro em causa.

4.2.3 authent.py

Ficheiro que gera a página HTML responsável por questionar ao utilizador o seu código, de forma a ser possível efetuar a verificação do segundo fator de autenticação.

4.2.4 GoogleAuthenticationSetup.txt

Ficheiro que possui uma explicação em relação ao modo de utilização e funcionamento da autenticação com recurso ao Google Authenticator, que exige código de confirmação por parte do utilizador, que é gerado a partir de um segredo partilhado entre si e o sistema de ficheiros.

5 Aspetos relevantes de Segurança

Durante o desenvolvimento de todo o programa, foram feitas várias reflexões acerca de como poderíamos aumentar a segurança do sistema. Então algumas coisas que tivemos em consideração foi:

- **Definir ficheiros críticos -** O definir de um conjunto de ficheiros permite-nos retirar informações sobre estes, tais como o *user_owner* e o *group_owner* e assim permitir gerir o sistema todo de ficheiros, dado este ser baseado essencialmente neste conjunto de ficheiros críticos que será mais protegido.
- **Guardar os dados numa BD -** O guardar dados de utilizador e dos ficheiros na base de dados permite aumentar a segurança do sistema, por várias razões, tal como falha do sistema, permitindo que estes dados não sejam perdidos e que possam facilmente ser implementados noutro sistema. O ser um sistema externo e logo em caso de acesso ao computador não puder aceder a estes dados, entre outros. Ou seja, a adição de mais uma camada permite aumentar a segurança deste.
- **Cada utilizador possui um segredo -** No que toca à utilização de autenticação de 2 fatores, cada utilizador possui um segredo próprio, estando este guardado na Base de Dados do sistema. Isto permite aumentar a segurança dado que a descoberta de um segredo não implica a descoberta do segredo de todos os utilizadores.
- **Access Control List -** A criação de uma access control list permite que durante a execução do sistema de ficheiros seja adicionados à Base de Dados, os dados de um utilizador que poderá passar a pertencer ao conjunto de utilizadores que poderá aceder ao ficheiro com a inserção do código OTP pedido na altura do acesso ao mesmo.
- **Criação de um utilizador na base de dados** - Apesar não ter sido possível implementar, esta estratégia baseia-se na restrição de acesso à base de dados em questão, por parte de qualquer utilizador, permitindo apenas que o administrador do sistema o faça. Isto baseia-se na utilização de *roles* correspondentes ao nível de autorização para consulta da BD.

5.1 CWE's e CVE's

Durante o desenvolvimento do sistema foi tido em conta alguns aspetos de segurança, tal como os descritos acima. Isto permite evitar fraquezas no sistema e aumentar a probabilidade de existência de vulnerabilidades. Então alguns dos CWE's que foram tidos em conta foram:

- **CWE-798 (*Hard Coded Credentials*) -** A utilização da base de dados permitiu-nos também ter esta vantagem. Os dados relacionados com a parte de autenticação 2FA encontram-se guardados na base de dados. Isto é, o segredo de cada utilizador usado para a geração do código encontra-se guardado na base de dados, não sendo exposta nenhuma informação diretamente, não exisitndo assim *Hard Coded Credentials*;
- **CWE-862 (*Missing Authorization*) -** A base do nosso sistema funciona precisamente para evitar que isto aconteça. Todo o acesso a ficheiros críticos é protegido pela autenticação 2FA (exceto o acesso do user_owner), sendo feita a verificação individual a cada utilizador que acessa ao mesmo;

6 Modo de utilização

Para inicializar o sistema devem ser executados os seguintes comandos:

1. Fazer o setup de todas as independências (Deverá ser feita pelo utilizador Root):

```
$ python setup.py
```

2. Verificar a conexão à Base de Dados:

```
$ mongo status
```

3. Inicializar o Sistema de Ficheiros (Deverá ser feita pelo utilizador Root)

```
$ python Passthrough.py folder mountpoint
```

4. Carregar para o sistema as informações dos utilizadores

```
$ mongoimport -d filesystem -c users users.json --jsonArray
```

5. Adicionar permissões a ficheiros(opcional)

```
$ python ACL.py user_id user_email ficheiro
```

6. Aceder à diretoria onde o sistema foi montado e tentar aceder a um ficheiro (sugestão!)

```
$ cat ficheiro
```

7. Introduzir o código de autenticação.

Figura 2: Página de inserção do código de autenticação

8. O sistema deve: apresentar o ficheiro ou informar a impossibilidade de o fazer, devido ao código incorreto, ou não possuir permissões.

7 Conclusão

Após a resolução deste trabalho, podemos considerar que as principais dificuldades foram encontradas no âmbito de garantir a segurança das informações necessárias à execução do programa, nomeadamente no que toca à base de dados.
além disso, perceber a documentação e funcionamento da biblioteca Fuse foi algo que nos exigiu bastante tempo e dedicação, para ser possível a integração das funções necessárias de gestão de acesso a ficheiros.

Referências

[1] Fermentas Inc., ”Phage Lambda: description & restriction map”: November 2008, em "http://www.fermentas.com/techinfo/nucleicacids/maplambda.html"

[2] ”Setup pyotp with google authenticator”: 2021, em https://www.youtube.com/watch?v=C-jkO6coJkk

[3] ”Pyotp”: 2021, em https://pyauth.github.io/pyotp/

[4] ”Setup MongoDB in kali”:2021, em https://medium.com/cyber4people/setup-mongodb-in-kali-linux-3ab86731e3ec

[5] ”Fusepy”: 2021, em https://pypi.org/project/fusepy/

[6] ”Flask”: 2021, em https://flask.palletsprojects.com/en/2.1.x/

[7] ”CWE 862”: 2021, https://cwe.mitre.org/data/definitions/862.html

[8] ”CWE 798”: 2021, em https://cwe.mitre.org/data/definitions/798.html