Universidade do Minho
Mestrado em Engenharia Informática

Trabalho Prático 1
Tecnologias de Segurança

GRUPO:
Carlos Miguel Luzia de Carvalho - PG47092
Maria Beatriz Araújo Lacerda - A89535
Tiago Carvalho Freitas - PG47687

27 de Março 2022


PG47092


A89535


PG47687

Índice

1 Introdução

No âmbito da unidade curricular Tecnologia de Segurança desenvolvemos este projeto que consiste na implementação de um sistema de ficheiros no qual o acesso aos ficheiros seria feito através da introdução de um username, uma password e ainda um código OTP.

Para isto, usamos a API Libfuse e a linguagem C. Tal como mencionado no enunciado, utilizamos o exemplo *passthrough.c* que permite o *mirror* da pasta root. Adicionalmente, para o envio do código OTP optamos por fazê-lo através de um SMS, utilizando, para isto, a API Twilio.

2 Implementação

Primeiramente, analisamos as *system calls* que se encontram na *struct fuse_operations*. Dado que queremos manipular a abertura dos ficheiros, dando autorização a apenas utilizadores autorizados para tal, optamos por alterar a *system call open*, visto que esta é chamada sempre que um ficheiro é aberto.

Deste modo, será nesta função que iremos implementar a autenticação dos utilizadores, bem como a geração e o envio do código OTP.

2.1 Autenticação, Autorização e Verificação

De forma a registar todos os utilizadores, criamos um ficheiro .txt que guarda as credenciais destes. Este ficheiro será guardado na pasta Teste dentro da *root*. Relativamente à estrutura deste documento, cada linha representa um utilizador sendo que contem quatro informações sobre este: *username*, *password*, contacto de telémovel(que será utilizado para o envio do código) e *id* do user. Para além deste ficheiro, criamos, ainda um ficheiro permissions.txt. Neste caso, cada linha terá um ficheiro seguido dos ids dos *users* que têm autorização de acesso a ele.

Para obtermos toda a informação necessária para a execução do programa, por parte do utilizador, recorremos a *pipes* e usamos as funções *dup* e *dup2* para conseguirmos direcionar o *stdin* para os nossos *file descriptors* e assim guardarmos os dados que o utilizador insere na *bash*.

Assim, quando um utilizador quer aceder a um ficheiro é, primeiramente, pedido o seu *username* e *password*. Para a recolha destes dados, foi utilizada a função *system* bem como o *zenity*. O *Zenity* permite que haja "diálogo" com o user através de interação gráfica. Por sua vez, a função *system* permite a execução dos comandos. Após a introdução dos dados do utilizador, guardamos os dados introduzidos em duas variáveis (myuser e mypass) e passamos os resultados obtidos para a função **verifica_credenciais**.

Esta função é responsável pela autenticação do utilizador. Para este efeito, percorremos o ficheiro mencionado anterioremente (autenticação.txt) e percorremos linha a linha este ficheiro. Quando encontramos um utilizador no ficheiro com o mesmo *username* introduzido, guardamos os seus dados na estrutura credenciais, na qual credenciais[0] corresponde ao *username*, credenciais[1] à sua *password* e credenciais[2] ao contacto registado. Sendo assim, comparamos credenciais[1] à *password* introduzida pelo utilizador. Caso estas coincidam, consideramos que o utilizador se autenticou com sucesso.

Caso as credenciais estejam certas, isto é, a *password* estar de acordo com o *user*, vamos verificar se este tem acesso ao ficheiro em questão. Se não estiver na lista de permissões, o acesso irá lhe ser negado enquanto que, se corresponder a um dos ID's de utilizadores que têm acesso, poderá continuar o processo.

2.2 Geração e envio do código

Após autenticarmos o utilizador e verificarmos que este tem autorização de acesso ao ficheiro em questão, vamos proceder à geração de do código que vamos enviar ao utilizador. Para isso, desenvolvemos a função **string_alloc** que gera um código com 7 caracteres selecionados aleatoriamente de um array de caracteres. Seguidamente, utilizando novamente a função *system* executamos o *script* em python sms.py que tem como argumentos o código gerado e o contacto telefónico do utilizador autenticado (credenciais[2]). Neste *script*, através da API mencionada anteriormente (Twilio) enviamos o código ao utilizador.

Outra vez utilizamos o *zenity* para recolher o código introduzido pelo utilizador. Através da flag *–timeout* conseguimos definir o tempo que o utilizador tem para inserir o OTP que recebeu, que, neste caso, será de 60 segundos. Finalmente, comparamos esse código com o código gerado previamente e que foi enviado para o utilizador. Se eles forem iguais, então o utilizador terá acesso ao ficheiro. Se isto não acontecer, será devolvida uma mensagem de erro.

2.3 Proteção dos registos de utilizadores e permissões

Adicionalmente, quisemos proteger quer o ficheiro *autenticação.txt* quer o ficheiro *permission.txt*. Deste modo, no documento *permission.txt*, ambos os ficheiros apenas estão associados a um único id: 0. Este id pertence ao utilizador que tem privilégios *root*. Assim, qualquer outro utilizador que aceda a estes ficheiros será confrontado com uma mensagem de erro que o informa que não tem autoridade para o fazer. Mais uma vez, utilizamos o *zenity* para gerar a mensagem de erro que irá surgir.

3 Execução do Programa

De modo a facilitar o processo de compilação do programa utilizamos as sugestões presentes no repositório do libfuse, o que inclui a utilização de duas ferramentas: *ninja* e o

meson. Assim, temos 2 *scripts*: setup.sh make.sh.

O ficheiro setup.sh é responsável por configurar as pastas. Isto é criar uma pasta Teste e passar para lá o ficheiro autenticação.txt bem como outros ficheiros não protegidos para testar o sistema. Este script deve ser executado com *sudo* dado que cria pastas na root.

De seguida, com o script make.sh fazemos os primeiros passos sugeridos no repositório do libfuse: descomprimimos a pasta que contem o fuse (retirada do repositório) e passamos para dentro dela o passthrough.c que nós modificamos. Fazemos depois uma pasta build, entramos na pasta e executamos o *meson*. De seguida executa o ninja.

4 Conclusão e Trabalho Futuro

Este projeto permitiu-nos expandir o nosso conhecimento visto que ganhamos mais consciência do que é implementar um *filesystem*, mesmo que simples, e aplicar os conhecimentos obtidos nas aulas teóricas. Adicionalmente trabalhamos com APIs que nunca tinhamos usado.

Após uma reflexão sobre o nosso trabalho consideramos que o concluímos com sucesso dado que todos os objetivos principais foram cumpridos.