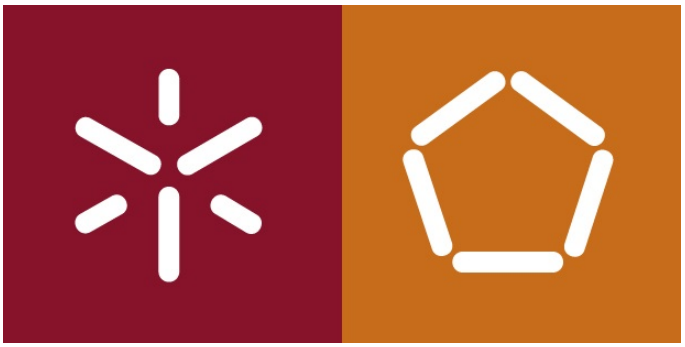

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

SEGURANÇA DE SISTEMAS INFORMÁTICOS

Autorizacão de Operações ao nível do Sistema de Ficheiros

Grupo 7

Sérgio Jorge (A77730)

Vítor Castro (A77870)

18 de Janeiro de 2019

Introdução

A informação surge, cada vez mais, como algo de suma importância para pessoas e, principalmente, para as organizações sendo fundamental em negócios. Por isso, é importante que seja protegida e, uma vez que há um grande crescimento da interconectividade, esta fica exposta a uma grande variedade de ameaças.

Daqui manifesta-se uma necessidade de implementar um conjunto de controlos adequados que visam aumentar a segurança e prevenir eventuais riscos.

Este trabalho tem, por base, uma tecnologia que busca, de facto, uma melhoria na segurança dos sistemas de informação e enquadra-se numa política de segurança moderna, a Multi-Factor Authentication. Consiste em utilizar mais do que um processo para autenticar um utilizador.

Implementou-se, por isso, um sistema que gera um código aleatório que é enviado para o utilizador. O utilizador terá então 30 segundos para escrever esse código na janela sugerida, sendo-lhe concedido ou não o acesso.

Arquitetura

Colocadas as exigências a alcançar para o sistema de ficheiros personalizado, implementaram-se diversas alterações ao *xmp_open* original do *passthrough.c* disponibilizado no *GitHub* oficial.

As personalizações implementadas foram, pela seguinte ordem de execução na função personalizada, as seguintes:

- Verificar se o utilizador que está a utilizar o sistema de ficheiros consta da lista de utilizadores permitidos ao uso;
- Obter o *email* do respetivo utilizador, a partir do ficheiro em que se verificou a permissão do utilizador;
- Gerar a chave a ser enviado para o utilizador;
- Enviar o email contendo a chave gerado;
- Abrir uma janela local para inserção do código enviado;
- Verificar a inserção da chave e verificação da sua correção.

Assim, nas próximas subsecções, explica-se com maior detalhe a implementação destas funcionalidades.

Verificação do Utilizador e Email

No sistema implementado, utilizou-se um ficheiro de texto para se guardarem todos os utilizadores permitidos ao uso e os respetivos contactos. Assim, quando o programa inicia, a primeira verificação que faz é se o utilizador logado no sistema está registado no sistema de ficheiros personalizado. Em caso afirmativo, o programa trata de extrair o contacto (email) para futuro uso e, em caso negativo, o programa mostra uma janela de ”Utilizador não autorizado”.

Geração da Chave

De seguida, procedeu-se à geração da chave que será posteriormente enviada para o utilizador. Por isso, definiu-se um array de caracteres constituído por letras maiúsculas e por algarismos e definiu-se um random que gera um número aleatório que mapeia um caracter nesse array. A partir de um ciclo, é então possível criar uma chave aleatória com 6 caracteres.

Envio da Chave por Email

O envio de emails, usando a linguagem de programação C, foi um dos principais obstáculos que o grupo encontrou para a realização e implementação deste sistema. O problema foi resolvido com o uso de uma biblioteca open-source que está disponível na rede: **libquickmail**.

Ultrapassada essa dificuldade, foi necessário subscrever um serviço de envio de emails o que foi possível graças ao site *sendgrid.net*.

Janela para Inserção da Chave

Havendo a necessidade de inserção da chave enviada, foi necessário perceber qual a melhor ferramenta a utilizar para recolher a mesma. Depois de ter explorado opções como servidores com interface web, optou-se por utilizar uma simples interface gráfica através do utilitário *yad*. Esta é uma versão melhorada do já muito conhecido *zenity*, que aplica o GTK+ de C, na sua conceção.

O *yad* permite, de entre várias coisas, implementar o *timeout* requerido de 30 segundos.

Verificação e Correção da Chave Inserida

A verificação da chave, que é de 6 caracteres, como vários sistemas atuais, é feita nos seguintes passos:

- Verificar introdução de chave. Se esta não for introduzida ao fim de 30 segundos, quebra-se logo a possibilidade de acesso ao ficheiro;
- Caso introduzida uma chave, ela é truncada aos primeiros 6 caracteres. Deste modo, é possível introduzir uma chave com mais de 6 caracteres que esteja correta, desde que os 6 primeiros sejam a chave. Com isto, não há possibilidade de ocorrerem *overflows* pois a truncação é feita na coleção do *input*;
- Verifica-se se a chave é a correta. Se for, dá-se permissão de acesso. Caso contrário, naturalmente, esta é negada.

Aspetos de Segurança

- O ficheiro *users.txt* que tem que ser bem protegido, uma vez que é um forte vetor de ataque para eventuais utilizadores maliciosos. Um acesso inesperado a este ficheiro permitiria adicionar um **id** e **email** para envio de códigos, dando acesso a todos os ficheiros;
- O *input*colocado pelo utilizador na janela do *yad* está protegido contra overflows, uma vez que apenas se selecionam os 6 primeiros caracteres introduzidos. De facto, mesmo que haja um conjunto de caracteres inserido imensamente superior aos 6 necessários, apenas os 6 primeiros contam para a verificação do código;
- Um acesso a uma conta de utilizador não permite uma abertura de ficheiros, uma vez que é também necessário possuir os códigos que são enviados para o *email*.

Dependências, Requisitos e Compilação

Devem instalar-se algumas dependências para a execução do *FUSE* da forma personalizada feita pelo grupo. Assim, sugerem-se os seguintes passos:

- Instalar o *gcc*, compilador necessário ao projeto em C, com o comando **sudo apt-get install gcc**. Adicionalmente, pode também correr-se o comando **sudo apt-get install build-essential**;
- Instalar o utilitário yad, com o comando **sudo apt-get install yad**;
- Instalar o curl, com o comando **sudo apt-get install libcurl4-openssl-dev**;
- Fazer download da biblioteca libquickmail: https://sourceforge.net/projects/libquickmail/ e extrair;
- Instalar libquickmail (**./configure, make e sudo make install**);
- Adicionar a linha **/usr/local/lib** ao ficheiro **/etc/ld.so.conf.d/local.conf**;
- Executar o comando: **sudo ldconfig**;
- Executar o comando **make all**, na pasta onde se encontram os ficheiros correspondentes ao trabalho;
- Copiar o ficheiro de output **my_passthrough** para **/home** e conceder permissão **755** para o mesmo (podia ser outra, mas esta é mais *standard* e não constitui risco);
- Criar um ficheiro com o nome **users.txt** em **/home**, com um utilizador ***root*** ou limitar a escrita no ficheiro com a permissão **744**, tendo em atenção o *user owner* e *group owner*. Em caso de um sistema com apenas um ***root***, criar o ficheiro com esse utilizador é suficiente. Caso hajam vários ***root***, é necessário modificar o *user owner* e criar um *group* específico. Este ficheiro deve ser criado, inserindo-se o id de utilizador a dar permissão, seguido de ponto e vírgula, seguido do email correspondente, uma linha por entrada (como exemplo que segue no trabalho);
- (REPETIR ESTE PASSO PARA CADA USER) Criar uma pasta com o nome de utilizador (podia ser outro, mas por convenção deverá ser assim). Por exemplo, o utilizador **user1** deverá criar a pasta, em **/home**, com o comando **mkdir user1**;
- (REPETIR ESTE PASSO PARA CADA USER) Executar o comando **./my_passthrough -f user1**, sendo **user1** a localização escolhida para ser montado o sistema de ficheiros. Naturalmente, caso seja outro utilizador, o comando deve ser **./my_passthrough -f NOME_DE_UTILIZADOR**;

Conclusão

Com este trabalho foi possível conhecer melhor o *FUSE* e, no fim, reconhecer a sua utilidade em diversas situações em que é necessário um mecanismo de controlo especial.

Tendo em mente os aspetos de segurança abordados na UC de SSI, foi possível fazer uma programação mais consciente e, no final, avaliar o trabalho a nível de segurança.

Reconhece-se assim a importância do conhecimento dos problemas habituais no domínio da programação informática e dos métodos de programação segura, tendo este sido um ótimo trabalho para a coesão de conhecimentos.