Trabalho Prático 3

Autorização de Operações ao nível do Sistema de Ficheiros

Universidade do Minho – Departamento de Informática

Mestrado em Engenharia Informática – Segurança de Sistemas Informáticos

Grupo 15:

- Pedro Afonso Rodrigues Santos – PG42847

1. Introdução

Este trabalho enquadra-se na UC de Segurança de Sistemas Informáticos sendo o 3º e o último Trabalho Prático a ser realizado.

Ao utilizar um sistema de ficheiros baseado em *libfuse*, a finalidade deste trabalho é criar um mecanismo adicional de autorização de operações de abertura de ficheiros que complementa os mecanismos de controlo de acesso de um sistema de ficheiros tradicional do sistema operativo Linux.
Um sistema de Arquivos *FUSE* [1] é implementado como uma aplicação independente que está ligado ao libfuse. A biblioteca fornece funções para montar o sistema de arquivos, desmontá-lo, ler solicitações do kernel e enviar respostas de volta. Existem dois tipos de API's: uma API síncrona de High-Level; uma API assíncrona de Low-Level. Com a API de High-Level, os retornos da chamada podem funcionar com nomes de arquivos e caminhos em vez de inodes, e o processamento de uma solicitação termina quando a função de retorno da chamada regressa. Com a API de Low-Level, os retornos da chamada devem funcionar com inodes e as respostas devem ser enviadas explicitamente ao utilizar um conjunto separado de funções de API.
O mecanismo desenvolvido deverá ser concretizado com estas etapas:

- autorizar a operação de abertura apenas depois de inserir um código de segurança enviado ao utilizador, via SMS ou email;
- guardar o registo de todos os utilizadores que poderão aceder ao sistema de ficheiros;
- deverá retornar com sucesso quando o código de segurança é enviado ao utilizador; ou com insucesso quando ultrapassar o limite de tempo de 30 segundos;

2. Arquitetura

Em relação á arquitetura e á estrutura do mecanismo desenvolvido, quando um utilizador solicita uma operação de abertura de um ficheiro, o programa solicita ao utilizador o seu username, e após submeter as suas credenciais, o programa lê o ficheiro *users.txt* para confirmar se esse username existe. Existem dois ficheiros chamados users.txt, onde fica guardado o username e o email do utilizador destinatário; e *credentials.txt* que contêm o email e a password do utilizador que envia o suposto email que contêm uma "chave". Caso o utilizador insira o username correto com um email existente, o programa vai confirmar que enviou o email, caso contrário, o programa indica que não foi possível enviar.
O programa ao gerar uma chave, cria um número aleatório de 6 dígitos, no qual é convertido numa string. Neste trabalho, foi utilizado o Python [2] para desenvolver o programa, e para que o programa pudesse enviar o email com a chave, foi preciso utilizar a biblioteca smtplib [3] para cumprir esse requerimento.
Após o envio do email, o utilizador só tem 30 segundos para autenticar com a chave. Se o utilizador não conseguir escrever a chave na janela a tempo, a chave expira e programa avisa que não será possível aceder ao ficheiro. Para cumprir esse requerimento, foram utilizadas bibliotecas Python (fusepy). Caso o utilizador consiga submeter a chave, o programa dá permissão de acesso. Caso a chave esteja incorreta, o programa irá negar o acesso.
É preciso ter atenção á segurança dos ficheiros users.txt e credentials.txt, pois se não estiverem bem protegidas, poderão estar expostos a ataques maliciosos no qual utilizadores maliciosos serão capazes de roubar os usernames, emails e passwords, par enviar mensagens maliciosas.
De forma a simplificar a Arquitetura e a Estrutura do mecanismo, a figura seguinte mostra um diagrama flowchart entre o Utilizador e o Sistema de Ficheiros.

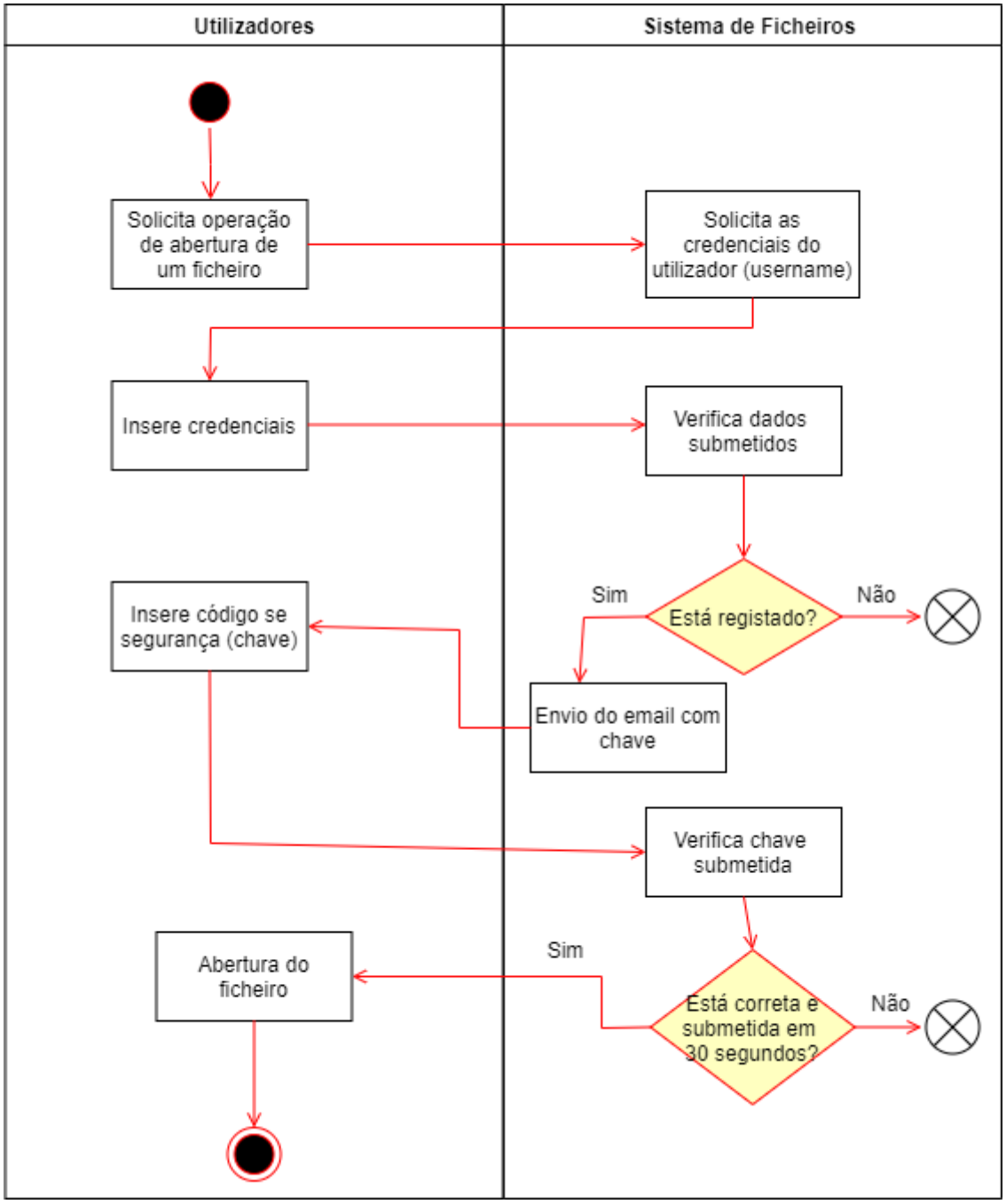

Fig. 1 – Diagrama entre Utilizador e Sistema de Ficheiros

3. Dependências e Instalação

Antes de explicar como foram feitas as instalações do programa, é necessário construir e instalar o FUSE com o *Meson* e o *Ninja.* Os detalhes sobre a instalação estão disponíveis no *GitHub.*
Para começar, utilize o comando **`sudo apt-get update`** para fazer download de pacotes de todas as fontes configuradas.
Para começar, é necessário instalar o Python com o **`pip install python`**, em que **`pip`** é um package manager para pacotes do Python, no entanto se não funcionar, utilize o **`pip3 install python`** para Python3.
Para exportar um sistema de ficheiros do libfuse para o Kernel Linux, precisamos de utilizar o comando **`pip3 install fuse-python`**.
O *fusepy* é um módulo Python que fornece uma interface simples para FUSE. Para instalar utilizamos o comando **`pip3 install fusepy`**.
Utilizar o módulo para definir um objeto de sessão de cliente SMTP que pode ser usado para enviar email a qualquer máquina da Internet com um daemon de escuta SMTP ou ESMTP.
É necessário importar a classe *MIMEText* para criar objetos MIME de texto de tipo principal, *MIMEMultipart* sendo uma classe base intermediária para mensagens MIME multiparte, *EmailMessage* sendo a classe base para o modelo de objeto de email.
Extrair o ficheiro libfuse tarball e criar uma pasta temporária chamada *build* e dentro dessa pasta, uma pasta chamada *exemple* onde criamos e executamos o ficheiro *passthrough.py*.
Na pasta exemple, precisamos de criar um ficheiro chamado users.txt que deve conter um username, um separador, e um email/receiverEmail (exemplo: UrandName/email.exemplo@hotmail.com); e credentials.txt que deve ter na primeira linha um email (senderEmail) e na segunda linha a password (senderPassword).
Criar uma pasta *teste*, onde fica guardado um ficheiro de projeto (exemplo: Makefile) para gerar programas executáveis; e criar uma pasta chamada *mount* (essa pasta fica vazia).
Utilizar o comando **`python3 passthrough.py teste / mount/`**, abrir um segundo terminar e abrir o mesmo caminho posto anteriormente (libfuse-fuse-3.10.1/build/example), aceder á pasta mount com **`cd mount`** e escrever o comando **`cat <ficheirodeprojeto>`** para executar o programa.

4. Vulnerabilidades e Fraquezas

O FUSE, tal como outros sistemas, fica exposto a certos tipos de vulnerabilidades (CVE) conhecidas.

Em relação á CVE [4] existem vários exemplos quando se trata do sistema de ficheiros FUSE:

- **CVE-2019-20794:** No Linux Kernel, nas versões 4.18 a 5.6.11, um utilizador pode criar o seu próprio namespace PID e montar um sistema de ficheiros FUSE, que pode resultar em esgotamento de recursos. Podem roubar os dados dos ficheiros users.txt;
- **CVE-2011-0543:** Certas funcionalidades em fusermount nas versões 2.8.5 e anteriores, quando util-linux não suporta a opção --no-canonicalize, permite que os utilizadores locais contornem as restrições de acesso pretendidas e desmontem diretórios arbitrários através de um ataque de link simbólico;
- **CVE-2011-0542:** fusermount na versão 2.8.5 não executa um chdir para/antes de executar uma montagem ou umount, o que permite aos utilizadores locais desmontar diretórios arbitrários por meio de vetores não especificados;
- **CVE-2011-0541:** versão 2.8.5 não controla adequadamente quando /etc/mtab não pode ser atualizado, o que permite que utilizadores locais desmontem diretórios arbitrários por meio de um ataque de link simbólico;
- **CVE-2010-3879:** O FUSE permite que utilizadores locais criem entradas mtab com pathnames arbitrários e, consequentemente, desmontem qualquer sistema de ficheiros, por meio de um ataque de link simbólico na diretoria pai do ponto de montagem de um sistema de ficheiros FUSE;
- **CVE-2010-0789:** O fusermount no FUSE antes de 2.7.5 e 2.8.x antes de 2.8.2, permite que utilizadores locais desmontem um compartilhamento de sistema de ficheiros FUSE arbitrário por meio de um ataque de link simbólico num ponto de montagem;
- **CVE-2005-1858:** O FUSE 2.x anterior a 2.3.0 não limpa adequadamente a memória usada de páginas não preenchidas quando o sistema de ficheiros retorna uma contagem curta de bytes para uma solicitação de leitura, o que pode permitir que utilizadores locais obtenham informações confidenciais.

Em relação á CWE existem vários exemplos quando se trata do sistema de ficheiros FUSE:

- **CWE-264:** relacionado com o gerenciamento de permissões, privilégios e outros recursos de segurança que são usados para executar o controlo de acesso;
- **CWE-59:** O software tenta aceder um ficheiro com base no nome, mas não impede adequadamente que identifique um link ou atalho que pode resultar num recurso indesejado.

5. Conclusões

Com este trabalho, fomos capazes de executar o FUSE como pretendido nos enunciados dados na unidade curricular, e levou a conhecer melhor a sua funcionalidade sendo um bom mecanismo para exportar sistemas de ficheiros para o Kernel Linux.

Em relação á linguagem utilizada, o Python facilitou a execução e a implementação do código passthrough.py, permitindo o envio de mensagens para vários utilizadores, mantendo alguns métodos de segurança para a prevenção de uso de dados pessoais.

Resumindo, este projeto deu importância de procurar novos conhecimentos em relação aos Sistemas de Ficheiros, novos tipos de programações e como manter a programação segura.

6. Bibliografia

[1] libfuse (GitHub): https://github.com/libfuse/libfuse

[2] Python: https://www.python.org/

[3] SMTP Protocol Client: https://docs.python.org/3/library/smtplib.html

[4] FUSE Securty Vulnerabilities: https://www.cvedetails.com/vulnerability-list/vendor_id-3090/Fuse.html