Escola de Engenharia

Universidade do Minho

Universidade do Minho
Mestrado em Engenharia Informática
Tecnologia de Segurança
Trabalho Prático 3
Deteção de Modificações Não-Autorizadas no Sistema Operativo

Grupo 5

PG47124 Daniel Filipe Santos Sousa,

PG47317 João Manuel Silva Amorim

INTRODUÇÃO

Para este trabalho foi-nos pedido para criar uma componente de software capaz de deter modificações não autorizadas num conjunto de ficheiros, para isso podíamos escolher uma opção entre a criação de um serviço de monitorização de modificações ou um sistema de ficheiros virtual idêntico ao que funciona no sistema Linux.
O nosso grupo escolheu a opção do sistema de ficheiros, realizando a sua implementação em python com a ajuda do módulo fusepy que integra a libfuse.

ARQUITETURA E ESTRUTURA

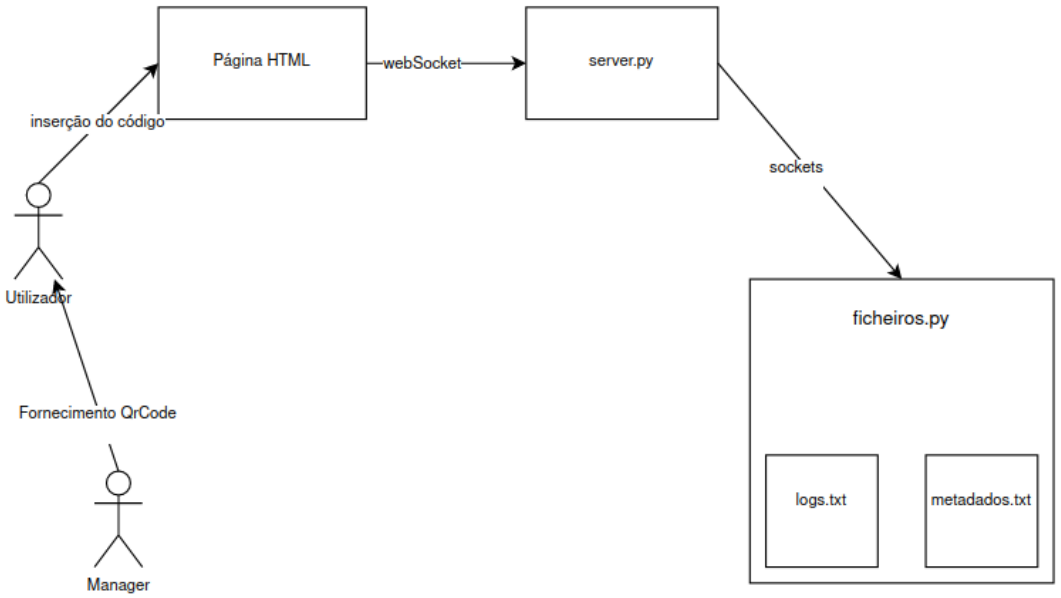

Figura 1- Representação do sistema

Para a implementação do sistema de ficheiros baseamo-nos nos exemplos fornecidos pela libfuse, tendo só de ver o correspondente em python.
Para isso, começamos pela criação da classe Passthrough que engloba os métodos todos que interagem com o sistema de ficheiros, tanto com os ficheiros bem como com as diretorias. Entre estes métodos está o de criar e remover diretorias, o de ler a diretoria, o de abrir, escrever e ler ficheiros, entre outros. Basicamente, todas as interações que se pode realizar com o sistema de ficheiros na nossa máquina está implementado na nossa classe.

Os métodos modificados foram:

- open
- read
- write
- chmod

No caso da modificação dos métodos do open, read e write foi para configurar as permissões de acesso ao ficheiro tendo em conta as permissões que estes têm no sistema. Assim sendo, é feita uma verificação se o utilizador é o dono do ficheiro, se pertence ao grupo ou se pertence à categoria de outros utilizadores. Após esta verificação observa-se as permissões que o ficheiro possui e atribui-se o acesso se assim for possível. É importante referir que caso o utilizador não tenha permissão é aberta uma página html, para a inserção de um código gerado através do google authenticator, e se o código for o correto, o utilizador passa a ter permissão para a ação desejada. No caso do chmod, foi alterado para proteger o nosso sistema de ficheiros de forma apenas a ser possível a alteração de permissões ao dono do ficheiro.
Posteriormente, usando o método "FUSE" da biblioteca fuse criamos um sistema de ficheiros a partir de outro já existente. Por outras palavras, realizamos uma cópia de uma diretoria existente, para uma nova criada usando o módulo fuse. A diretoria criada irá então obedecer aos métodos definidos na classe referida no parágrafo anterior. É importante referir que esta nova diretoria foi criada com as seguintes flags:

- **nothreads=True**, diz que o programa pode ser executado só com uma thread.
- **foreground=True**, não permite que o programa seja executado em segundo plano.
- **allow_other=True**, permite que outros utilizadores para além do criador da diretoria acedam a esta.

Na nossa implementação também foi considerado um ficheiro de metadados que contém, o nome, as permissões, o nome do utilizador dono do ficheiro, o nome do grupo que o utilizador dono pertence e, no caso de ser ficheiro, a hash do ficheiro.
A criação do ficheiro é realizada antes da diretoria ser copiada, tendo o seguinte formato:
"nome do ficheiro"-"dono do ficheiro"-"grupo do dono do ficheiro"-"permissões" -"hash do ficheiro"

Este ficheiro de metadados também possui o contacto do dono do sistema de ficheiro para possível obtenção do QRCode para obter o código otp.

No caso de ser diretora é igual menos no caso da hash do ficheiro. Esta hash serve para verificar posteriormente se o ficheiro não foi modificado.

O ficheiro de logs também faz parte do nosso programa. Este ficheiro contém logs sobre quando um utilizador acede a um determinado ficheiro ou diretoria. Estes logs também avisam quando um utilizador tenta realizar alguma ação a um ficheiro sem ter permissão para tal. Também avisa quando são garantidas permissões de acesso a um determinado ficheiro a um dado utilizador.
Um exemplo desses logs é o seguinte:

2022-06-08 16:00:42 INFO O utilizador facahd abriu o ficheiro agent.py
2022-06-10 21:30:54 INFO Foi concedida permissão ao utilizador Daniel para abrir o ficheiro testes.py

Onde é apresentada a data , o tipo de log e a mensagem que retrata a ação do utilizador.

Para a parte de ser possível permitir a um utilizador sem permissões o acesso a um determinado ficheiro foi criada uma classe servidor, que é um websocket, que através de sockets envia ao programa o código inserido pelo utilizador numa página html gerada, quando este executa alguma ação para a qual não tem permissão.Por outras palavras, um utilizador quando tenta a aceder a um ficheiro que não tem permissão é gerada uma página html.O utilizador insere o código gerado pelo google authenticator e se bater certo o sistema permite realizar a ação do utilizador, caso contrário não.
Este websocket funciona em paralelo com o sistema de ficheiros, tendo de ser inicializado antes do ficheiro que contém as instruções para o sistema de ficheiros.
Outra função implementada foi a criação de uma espécie de cache para não obrigar o utilizador a inserir o código otp cada vez que queira aceder ao ficheiro e desse modo só tem de o colocar uma vez.

SEGURANÇA

No desenvolvimento deste projeto o nosso grupo teve em conta alguns CWE's, e dessa forma implementamos o código de maneira a mitigá-las. O CWE-280 é uma fraqueza associada ao mau tratamento de permissões ou privilégios insuficientes, para combatê-la o nosso grupo verifica se o utilizador tem as permissões necessárias para executar a ação que pretende. O CWE-276, que diz respeito à definição incorreta de permissões default durante a instalação, também foi tido em conta e para o mitigar quando é criado o novo sistema de ficheiros através do nosso

programa, as permissões dos ficheiros são mantidas, evitando assim que um utilizador que não tenha acesso a um ficheiro, tenha no novo sistema de ficheiros.
Para além disso o sistema de logs implementado, ajuda o gerente do sistema de ficheiros a ter uma perceção bastante concisa do que está a acontecer. Uma vez que este gera informação em quase todas as ações do utilizador e avisos, quando o utilizador tenta realizar uma ação não permitida.
Ainda para garantir a segurança do ficheiro de metadados foi atribuída apenas uma permissão de leitura aos outros utilizadores enquanto que no ficheiro de logs apenas o dono do sistema é que tem acesso.

MODO DE EXECUÇÃO

Para executar o nosso programa é necessário fazer unzip do projeto e de seguida correr os seguintes comandos:

1. **“python3 server.py”** - Este comando cria um servidor para ser possível a utilização dos otp’s.
2. **“python3 ficheiros.py ‘diretoria a copiar’ ‘ponto de montagem’ “ -** Este comando cria o novo sistema de ficheiros, nele é necessário indicar a diretoria que se quer copiar para o novo sistema de ficheiros e o seu ponto de montagem, este representa uma pasta que tem de ser previamente criada e onde vai estar o nosso novo sistema de ficheiros.

CONCLUSÃO

Concluindo, o nosso grupo pensa ter cumprido os objetivos do trabalho, uma vez que conseguimos implementar um sistema de ficheiros virtual que utiliza a biblioteca do libfuse e ainda conseguimos implementar o objetivo opcional de enviar um otp para um dado utilizador de forma a este ter acesso a um ficheiro que anteriormente não tinha. Este trabalho permitiu-nos aprofundar os nossos conhecimentos sobre os sistemas de ficheiros, bem como a interceptar system calls através do libfuse. As maiores dificuldades que sentimos foram no início quando ainda estamos a aprender a usar a biblioteca e relativamente ao uso do vim para escrever num ficheiro, uma vez que ao abrir um ficheiro usando um editor de texto, este gera vários ficheiros temporários que interferem no nosso modo de controlo de permissões.

Referências:

1. https://thepythoncorner.com/posts/2017-02-28-writing-a-fuse-filesystem-in-python/
2. https://github.com/fusepy/fusepy/blob/master/examples/context.py