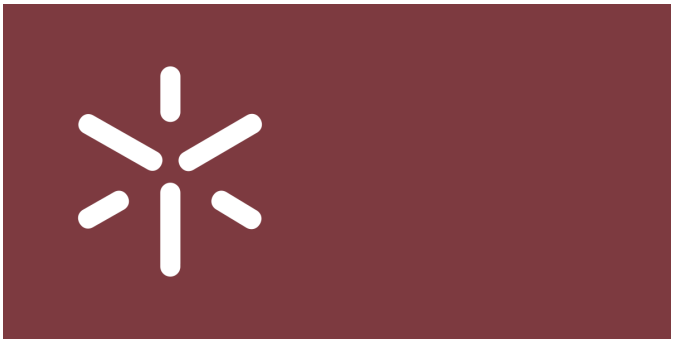

Universidade do Minho

Mestrado em Engenharia Informática

Tecnologias de Segurança

Djumbai Serviço Local de Troca de Mensagens - Grupo 11

A74806 - João Amorim

10 de Maio de 2024

Índice

Lista de Figuras

1 Introducão

1.1 Contextualização

A necessidade de um serviço de conversação robusto e seguro em sistemas Linux é evidente, dado o aumento da digitalização nas interacções. O serviço proporcionará comunicação assíncrona e síncrona entre Utilizadores locais, bem como a funcionalidade de grupos privados. O projecto deve assegurar a confidencialidade, integridade e disponibilidade das mensagens, integrando-se nas práticas de segurança do sistema operativo Linux.

1.2 Objetivos

O objectivo é criar um serviço de conversação que abranja comunicação assíncrona e síncrona, com gestão de Utilizadores e grupos privados. O foco será desenvolver mecanismos de segurança robustos para garantir a confidencialidade e integridade das mensagens. Além disso, funcionalidades como criptografia e integração com sistemas de gestão de eventos serão exploradas para melhorar a segurança e eficácia do serviço.

2 Introdução - Decisões Arquiteturais

Para o desenvolvimento do Djumbai, foi preciso fazer um planeamento inicial que descrevesse a divisão das camadas de serviço, em que cada uma corresponde à implementação dos mecanismos e funcionalidades necessárias para atingir os objetivos propostos no enunciado.

Nesta secção, será feita uma apresentação dessas camadas de serviço, descrevendo os intervenientes bem como os objetivos de cada uma, de maneira a que seja possível obter uma percepção daquilo que é o sistema Djumbai e daquilo que o mesmo oferece aos Utilizadores.

Aspetos da arquitetura como diretorias criadas/existentes, permissões específicas implementadas e para que diretorias/ficheiros, algoritmos de cifra utilizados, entre outros, serão abordados na secção seguinte, onde irá ser feita uma abordagem mais detalhada de cada camada do serviço.

2.1 Gestão de Utilizadores

2.1.1 Utilizadores do Sistema

No que diz respeito aos Utilizadores do sistema, foi escolhida a implementação de dois tipos de Utilizadores, nomeadamente um "superuser", responsável pela gestão de todos os outros Utilizadores e grupos no sistema Djumbai, e os Utilizadores do próprio sistema Linux onde o Djumbai vai correr, que consegue fazer uso das funcionalidades principais do sistema, como enviar e ler mensagens enviadas para outros Utilizadores ou grupos de conversação.

2.1.2 Preparação de Utilizadores

Para o correto funcionamento do sistema e tendo em conta que os Utilizadores do Djumbai serão de facto os Utilizadores registados no sistema Linux onde a aplicação está a ser executada, era necessário efetuar uma preparação para que os Utilizadores conseguissem fazer uso de todas as funcionalidades, tais como enviar e receber mensagens. Esta preparação passa por aquilo a que foi dado o nome de ativação e desativação de Utilizadores.

Estes dois processos de ativação e desativação consistem na criação e remoção dos caminhos onde serão guardadas as mensagens de cada Utilizador, em que estes podem ser executados pelo superuser, permitindo que o mesmo execute as operações para qualquer

user registado no sistem Linux, ou então, apenas para o caso da ativação, esta será feita automaticamente pelo sistema Djumbai caso o Utilizador que se tenha conectado á aplicação ainda não estivesse ativado.

2.2 Gestão de Grupos

A gestão de grupos, em parte semalhante à gestão de Utilizadores, é feita exclusivamente pelo "superuser", podendo este criar e remover um grupo, bem como adicionar e remover Utilizadores aos mesmos. A criação e remoção de grupos, tal como acontece com os Utilizadores, faz toda a gestão de diretorias e ficheiros necessária de modo a que todas as funcionalidades sejam executadas corretamente, de forma segura e consistente.

2.3 Gestão de Mensagens

2.3.1 Envio e Recebimento de Mensagens

Em termos de funcionalidades relacionadas com o envio e recebimento de mensagens, era preciso implementar o seguinte:

- Envio de Mensagens síncrono e assíncrona de um Utilizador para outro Utilizador
- Envio de Mensagens síncrono e assíncrona de um Utilizador para um Grupo de Conversação
- Leitura de Mensagens respetivas ao Utilizador
- Leitura de Mensagens dos Grupos a que um Utilizador pertence

2.3.2 Ficheiros de Mensagens

De maneira a dar resposta às funcionalidades descritas em cima, foi escolhida a opção de criação de ficheiros para guardar todas as mensagens referentes aos Utilizadores e aos grupos de conversação.

Cada Utilizador tem o seu respetivo ficheiro "messages.txt", criado no momento da sua ativação, e, da mesma maneira, cada grupo de conversação tem também um ficheiro com o mesmo nome.

2.4 Arquitetura Cliente-Servidor

Antes de falar das permissões dadas a cada uma das componentes do sistema, fará sentido falar da arquitetura Cliente-Servidor criada. O sistema Djumbai divide-se em duas partes, a parte com a qual o Cliente interage, que mostra a interface ao Utilizador, envia os pedidos ao Servidor e recebe as respetivas respostas e que trata dos pedidos de leitura das mensagens dos Utilizadores ou grupos de conversação.

Do outro lado, temos o Servidor que está constantemente à espera de novas conexões por parte de Utilizadores do sistema, fazendo toda a gestão de Utilizadores, criação de ficheiros, chamada de scripts para ativação, desativação e gestão de grupos, envio de mensagens e finalmente implementação das medidas de segurança. Cada um dos aspetos referentes a estes componentes será visto mais a fundo nas secções que se seguem.

2.5 Permissões do Sistema

O conceito das permissões neste trabalho é sem dúvida um dos mais importantes e aquele que deu o maior trabalho. Tendo em conta que foi escolhido o caminho da criação de diretorias e ficheiros para guardar, foi necessário garantir que todas as permissões dessas mesmas diretorias e ficheiros estavam definidas de maneira a garantir a intergridade e confidencialidade dos dados.

A definição destas permissões partiu da arquitetura base que já estava definida, ou seja, da decisão de quem iria fazer a gestão de Utilizadores e de grupos, o tratamento de envio de mensagens e finalmente da leitura das mensagens. Sendo a gestão de Utilizadores e grupos efetuada pelo Servidor, em que esta gestão corresponde apenas à criação ou remoção de diretorias e ficheiros, era necessário atribuir permissões de escrita ao Servidor e garantir que apenas ele conseguiria escrever nos mesmos. Do mesmo modo, e no que diz respeito ao envio de mensagens, também este feito pelo Servidor, era necessário que, no momento de envio de uma mensagem por um Utilizador, o Servidor conseguisse escrever e guardar a mensagem enviada no ficheiro do Utilizador ou Grupo destino.

Por outro lado, a leitura das mensagens é feita sem recorrer a funcionalidades do Servidor, sendo feita diretamente pela aplicação do Cliente, pelo que foi necessário definir permissões de leitura para os ficheiros referentes ao próprio Utilizador ou aos grupos dos quais esse mesmo Utilizador fizesse parte.

Em suma, as permissões definidas em cima e qualquer outro ficheiro, tais como scripts ou próprios ficheiros onde a implementação do Djumbai foi feita, pode ser resumida como:

- Proteção de Dados: As mensagens são protegidas contra acesso não autorizado, garantindo que apenas os destinatários pretendidos possam lê-las.
- Integridade do Sistema: Apenas o Servidor pode modificar a estrutura das diretorias e os ficheiros críticos do sistema, prevenindo alterações maliciosas ou acidentais.
- Privacidade do Utilizador: Cada Utilizador tem controle exclusivo sobre seu próprio diretório e suas mensagens, assegurando a privacidade dos dados pessoais.
- Segurança de Grupos: As mensagens de grupos são acessíveis apenas aos membros do grupo, garantindo a confidencialidade das comunicações internas do grupo.

2.6 Mecanismos de Segurança

Como mecanismos de segurança implementados, temos o seguinte:

- Permissões do Sistema - Tal como vimos, implementação correta das permissões impede o acesso a partes críticas do sistema.
- Criptografia das Mensagens guardadas nos ficheiros - A criptografia das mensagens é implementada utilizando a biblioteca OpenSSL. O Servidor gera uma chave AES-256, que é armazenada de forma segura. As mensagens são criptografadas antes de serem armazenadas e descriptografadas quando lidas pelos Clientes.

3 Arquitetura Funcional

3.1 Client.c

Tal como já vimos na secção anterior, o sistema Djumbai consiste em dois programas principais: client.c e server.c. Ambos têm funções específicas que garantem a operação do serviço de mensagens.

- activate_user(int sock, const char *username): Envia um comando ao Servidor para ativar um Utilizador.
- deactivate_user(int sock, const char *username): Envia um comando ao Servidor para desativar um Utilizador.
- send_message(int sock, const char *from_username, const char *to_username, const char *message): Envia uma mensagem de um Utilizador para outro.
- read_messages(const char *username): Lê as mensagens criptogradas e recebidas pelo Utilizador, descriptografando as mesmas. Não envia pedido ao Servidor uma vez que efetua diretamente a leitura do ficheiro de mensagens atribuído ao Utilizador.
- send_group_message(int sock, const char *from_username, const char *groupname, const char *message): Envia uma mensagem para um grupo.
- read_group_messages(const char *groupname): Lê as mensagens criptogradas de um grupo, descriptografando as mesmas. Não envia pedido ao Servidor uma vez que efetua diretamente a leitura do ficheiro das mensagens atribuído ao grupo que o Utilizador introduziu.
- manage_group(int sock, const char *action, const char *groupname, const char *username): Efetua a gestão dos grupos (criar e remover grupo, adicionar/remover Utilizadores). Esta é uma funcionalidade a que apenas o superuser do sistema tem acesso.
- list_activated_users(): Função para auxiliar superuser na ativação e desativação de Utilizadores, listando na consola todos os Utilizadores ativados no sistema linux, ou seja, que possuam a diretoria e ficheiro de mensagens criados.
- list_user_groups(const char *username): Apresenta ao Utilizador todos os grupos dos quais faz parte.

- menu(int sock, const char *username): Apresenta o menu principal para o Utilizador normal do sistema Linux.
- admin_menu(int sock): Apresenta o menu de administrador ao superuser que executa a aplicação do Cliente como root.
- main(): Efetua a conexão ao Servidor e fornece uma interface de linha de comando para que o Utilizador interaja com o sistema. Como vimos, esta interface está divida no menu, para o Utilizador normal e no admin_menu para o menu que é apresentado ao superuser que executa o Cliente como root.

3.2 Server.c

- activate_user(int client_sock, const char *username): Trata o pedido de ativação vindo do Cliente e ativa um Utilizador ao criar a diretoria para o seu ficheiro de mensagens, bem como o próprio ficheiro.
- deactivate_user(int client_sock, const char *username): Trata o pedido de desativação vindo do Cliente e desativa um Utilizador removendo a diretoria para o seu ficheiro de mensagens, bem como o próprio ficheiro.
- send_message(int client_sock, const char *from_username, const char *to_username, const char *message): Trata do pedido de envio de uma mensagem para o Utilizador passado como argumento. Guarda a mensagem no ficheiro de mensagens desse mesmo Utilizador.
- send_group_message(int client_sock, const char *from_username, const char *groupname, const char *message): Trata do pedido de envio de uma mensagem para o Grupo passado como argumento. Guarda a mensagem no ficheiro de mensagens desse mesmo Grupo.
- manage_group(int client_sock, const char *action, const char *groupname, const char *username): Efetua a gestão dos grupos no sistema, criando e removendo grupos ao criar e remover as diretorias e ficheiros de mensagens dos grupos passados como argumentos, e adiciona e remove Utilizadores de Grupos. Esta faz uso da criação de processos filho para a execução das ações necessárias na criação e remoção de grupos, ou no caso da adição e remoção de Utilizadores, as funções chamadas são executadas da mesma maneira.
- add_user_to_group(const char *username, const char *groupname) e remove_user_from_group (const char *username, const char *groupname): Estas funções auxiliares são responsáveis pela gestão de associações de Utilizadores a grupos no sistema Linux. Utilizam a criação de processos filhos para executar comandos administrativos com privilégios elevados, uma vez que modificar grupos de Utilizador requer permissões de superuser.

3.3 Diretorias e Ficheiros

Tendo em conta o que irá ser abordado nas próximas secções, fará sentido apresentar as diretorias e ficheiros criados e as decisões tomadas em relação às mesmas.

Uma vez que o Djumbai é uma aplicação local para o envio e recebimento de mensagens entre Utilizadores do sistema Linux ou de grupos do mesmo, era necessário garantir que qualquer Utilizador do sistema teria acesso ao mesmo, ou seja, que o sistema estaria numa diretoria acessível a todos os Utilizadores locais. Para isto, o Djumbai está localizado na diretoria "/usr/local/Djumbai". Esta diretoria é depois divida em:

```
|--------------------------|
| /usr/local/djumbai/      |
|  - bin/                  |
|    - client              |
|    - server              |
|  - scripts               |
|     - activate_user.sh   |
|     - deactivate_user.sh |
|     - manage_group.sh    |
|     - aes_key            |
|  - src                   |
|    - client.c            |
|    - server.c            |
|  - Makefile              |
|--------------------------|
```

A pasta "bin"contém os ficheiros executáveis do Client e do Server, a pasta "scripts"contém os scripts chamados pelo Servidor para a ativação e desativação de Utilizadores e a gestão de grupos, a pasta "src"contém os ficheiros .c que definem a implementação do Cliente e Servidor e finalmente a Makefile para a compilação e criação de algumas das diretorias com respetivas permissões.

Não menos importante, as diretorias e ficheiros criados para guardar as mensagens dos Utilizadores e dos Grupos, estão também guardados em diretorias em que a leitura das mensagens contidas nos ficheiros referentes aos seus donos seja possível sem que seja feito uso do Servidor. Para isto, foi escolhida a diretoria "/var/djumbai/"que é depois dividida nas seguintes subdiretorias:

```
|----------------------|
| /var/djumbai/        |
|  - users/            |
|    - <username>/     |
|      - messages.txt  |
|  - groups/           |
|    - <groupname>/    |
|      - messages.txt  |
|----------------------|
```

Aqui temos uma divisão bastante simples, em que uma das pastas, a de "users"contém todos os Utilizadores ativos no sistema com o respetivo ficheiro "messages.txt"onde estão incluídas todas as mensagens do Utilizador. Do outro lado e do mesmo modo, temos a pasta "groups"que contém todos os groups criados no sistema, novamente com o respetivo ficheiro de mensagens.

3.4 Interoperabilidade entre Componentes

Uma vez apresentados todos os componentes do sistema, o seu funcionamento e a sua estrutura, à exceção dos componentes de segurança, vamos agora apresentar o ciclo de vida da aplicação para os diferentes tipos de Utilizadores, descrevendo assim todo o funcionamento da aplicação. Em primeiro lugar, são apresentadas de seguida as imagens das Interfaces para o "superuser"e para o Utilizador normal, bem como o Servidor em funcionamento após a conexão de um Utilizador.

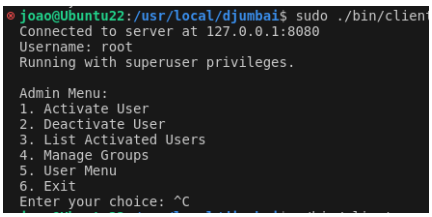

Figura 3.1: Interface do Superuser

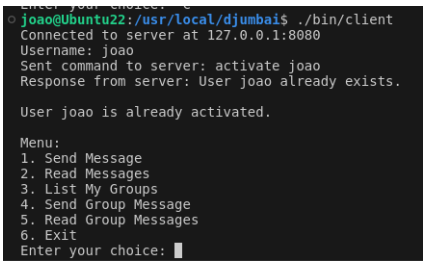

Figura 3.2: Interface do Utilizador normal

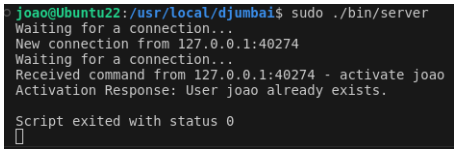

Figura 3.3: Funcionamento do Servidor

3.4.1 Funcionalidades do Superuser

De seguida, vai ser apresentado o funcionamento detalhado e por etapas das funcionalidades do Superuser:

- Execução da aplicação com privilégios de "Superuser"(root)
- Apresentação da Interface do "Superuser"ao Utilizador
- Escolha da opção "Activate User"
  - Inserção do <username> do Utilizador que se pretende ativar
  - Pedido é enviado para o Servidor que executa o script activate_user.sh.
  - Script de ativação cria a diretoria /var/djumbai/users/<username>/messages.txt
- Escolha da opção "Deactivate User"
  - Inserção do username do Utilizador que se pretende desativar
  - Pedido é enviado para o Servidor que executa o script deactivate_user.sh.
  - Script de desativação cria remove diretoria /var/djumbai/users/<username> e o ficheiro messages.txt dentro da pasta <username>
- Escolha da opção "List Activated Users". Aplicação do Cliente apresenta na consola o nome de todas as pastas existentes em /var/djumbai/users/, que no fundo correspondem aos Utilizadores ativados.
- Escolha da opção "Manage Groups", para criação ou remoção de um grupo ou para a adição ou remoção de um membro a um grupo.
  - Inserção do comando "create"seguido do nome do grupo a ser criado.
  - Inserção do comando "remove"seguido do nome do grupo a ser removido.
  - Inserção do comando "add_user"seguido do nome do grupo ao qual o Utilizador será adicionado e o <username> do mesmo Utilizador.
  - Inserção do comando "remove_user"seguido do nome do grupo do qual o Utilizador será removido e o <username> do mesmo Utilizador.
  - Esta gestão é feita através da criação de processos filhos para executar comandos administrativos com privilégios elevados, uma vez que modificar grupos de Utilizador requer permissões de superuser.

- Escolha da opção "User Menu"que apresenta o menu do Utilizador Normal apresentado de seguida.
- Escolha da opção "Exit"termina o funcionamento da aplicação.

3.4.2 Funcionalidades do Utilizador Normal

- Execução da aplicação com privilégios de "Superuser"(root)
- Apresentação da Interface de Utilizador normal ao Utilizador
- Escolha da opção "Send Message"
  - Inserção do <username> do Utilizador para onde enviar a mensagem, seguido do conteúdo da mensagem.
  - Pedido de envio de mensagem enviado para o Servidor que possuí as permissões necessárias para editar o ficheiro "messages.txt"do Utilizador para quem a mensagem é enviada.
  - Mensagens são guardadas no formato "Sender: <sender_username>; Message: <encrypted_message>"
  - Servidor envia de volta a resposta que, em caso de Utilizador inexistente, apresenta o erro, caso contrário indica que a mensagem foi enviada com sucesso.
- Escolha da opção "Read Messages"para leitura das mensagens do Utilizador que executou a aplicação. Aplicação do Cliente apresenta na consola todas as mensagens que foram enviadas para o Utilizador por outros Utilizadores do sistema Linux através do Djumbai, sem qualquer envio de pedido ao Servidor.
- Escolha da opção "List My Groups". Aplicação do Cliente apresenta na consola todos os Grupos dos quais o Utilizador faz parte no sistema.
- Escolha da opção "Send Group Message"
  - Inserção do <username> do Grupo para onde enviar a mensagem, seguido do conteúdo da mensagem.
  - Pedido de envio de mensagem enviado para o Servidor que possuí as permissões necessárias para editar o ficheiro "messages.txt"do Utilizador para quem a mensagem é enviada.
  - Mensagens são guardadas no formato "Group: <groupname>; Sender: <sender_username>; Message: <encrypted_message>"
  - Servidor envia de volta a resposta que, em caso do Utilizador não pertencer ao Grupo para o qual enviou a mensagem, apresenta o erro, caso contrário indica que a mensagem foi enviada com sucesso.

- Escolha da opção "Read Group Messages"para leitura das mensagens do Grupo do qual o Utilizador que executou a aplicação faz parte. Aplicação do Cliente apresenta na consola todas as mensagens que foram enviadas para o Grupo por qualquer Utilizador pertencente ao mesmo, sem qualquer envio de pedido ao Servidor.
- Escolha da opção "Exit"termina o funcionamento da aplicação.

4 Segurança do Sistema

4.1 Permissões de Ficheiros

As permissões são cruciais neste sistema de mensagens para garantir a segurança, privacidade e integridade dos dados dos Utilizadores. Elas impedem acessos não autorizados, protegendo a confidencialidade das mensagens e assegurando que apenas Utilizadores apropriados possam ler ou modificar determinados ficheiros e diretorias. Permissões adequadas também garantem que apenas o root possa alterar scripts e binários críticos, mantendo uma segura operação do sistema. Além disso, elas facilitam a conformidade com políticas de segurança e permitem melhor auditoria, assegurando que os dados dos Utilizadores sejam acedidos e manipulados apenas por aqueles que têm autorização para isso.

De modo a garantir isto, foram criadas as seguintes permissões:

4.1.1 Permissões: drwxr-xr-x (root:root)

Diretorias

- /var/djumbai/
- /var/djumbai/users/
- /var/djumbai/groups/
- /usr/local/djumbai/
- /usr/local/djumbai/bin/

Descrição

Estas diretorias são responsáveis por armazenar dados, binários, scripts, e código-fonte relacionados com o sistema de mensagens.

Estas permissões permitem que todos os utilizadores possam navegar até às suas respetivas subdiretorias, mas garantem que apenas o root possa fazer modificações para manter a integridade e segurança do sistema.

4.1.2 Permissões: drwxr-xr– (root:root)

Diretorias

- /usr/local/djumbai/scripts

Descrição

Diretoria que contém scripts necessários para a configuração e operação do sistema e ficheiro com a chave aes para encriptação.

Permitir que qualquer utilizador possa ler os scripts, mas apenas o root pode modificá-los para garantir a segurança e integridade do sistema.

4.1.3 Permissões: drwx—— (root:root)

Diretorias

- /usr/local/djumbai/src

Descrição

Diretoria que contém os ficheiros de código-fonte.

Apenas o root pode ler, escrever e executar os ficheiros nesta diretoria, garantindo a segurança do código-fonte.

4.1.4 Permissões: rwxr-xr-x (root:root)

Diretorias

- /usr/local/djumbai/bin/client

Descrição

Binário do cliente.

Permitir que qualquer utilizador execute o cliente, mas garantir que apenas o root possa modificar o binário.

4.1.5 Permissões: rwx—— (root:root)

Diretorias

- Permissões: rwx—— (root:root)

Descrição

Binário do servidor.

Apenas o root pode ler, escrever e executar o binário do servidor para garantir a sua integridade e segurança.

4.1.6 Permissões: rw——- (root:root)

Diretorias

- /usr/local/djumbai/Makefile
- /usr/local/djumbai/src/client.c
- /usr/local/djumbai/src/server.c

Descrição

Código-fonte do cliente e servidor, e ficheiro Makefile para compilação e configuração do sistema.

Garantir que apenas o root possa ler e modificá-los para manter a integridade do código-fonte.

4.1.7 Permissões: r——— (<username>:<username>)

Diretorias

- /var/djumbai/users/<username>/messages.txt

Descrição

Ficheiro onde são armazenadas as mensagens do utilizador.

Permitir que o utilizador leia e escreva no ficheiro, enquanto outros utilizadores não têm acesso, garantindo a privacidade das mensagens.

4.1.8 Permissões: dr-xr-xr-x (<username>:<username>)

Diretorias

- /var/djumbai/users/<username>

Descrição

Diretoria pessoal de cada utilizador, contendo o ficheiro messages.txt.

Permitir que o utilizador e outros possam navegar até à diretoria, mas garantir que apenas o utilizador dono possa alterar os conteúdos da diretoria, mantendo a privacidade e segurança das mensagens.

4.1.9 Permissões: drwxr-x— (root:<groupname>)

Diretorias

- /var/djumbai/groups/<groupname>

Descrição

Diretoria específica para cada grupo, contendo o ficheiro messages.txt.

Apenas membros do grupo e o root podem aceder e ler os conteúdos da diretoria, garantindo a privacidade e segurança das mensagens do grupo.

4.1.10 Permissões: rw-r—— (root:<groupname>)

Diretorias

- /var/djumbai/groups/<groupname>/messages.txt

Descrição

Ficheiro onde são armazenadas as mensagens do grupo.

Apenas membros do grupo podem ler o ficheiro, e apenas o root pode escrever nele, garantindo a integridade e segurança das mensagens do grupo.

4.1.11 Permissões: rw———- (root:root)

Diretorias

- /usr/local/djumbai/scripts/aes_key

Descrição

Ficheiro contendo a chave AES para criptografia.

Apenas o root pode ler e escrever neste ficheiro para garantir a segurança da chave criptográfica, impedindo acesso não autorizado.

4.2 Criptografia das Mensagens

A criptografia de mensagens é uma parte crucial da segurança da aplicação, garantindo que as mensagens armazenadas no sistema não possam ser lidas por utilizadores não autorizados, mesmo que obtenham acesso físico aos ficheiros de mensagens. Para implementar a criptografia em repouso, utilizamos a criptografia AES (Advanced Encryption Standard) com a biblioteca OpenSSL.

4.2.1 Implementação da Criptografia

Geração e Armazenamento da Chave AES:

Foi gerada uma chave AES segura utilizando OpenSSL. A chave foi armazenada no arquivo /usr/local/djumbai/scripts/aes_key.

Para garantir a segurança, o arquivo aes_key foi configurado com permissões rw———- (read/write apenas para o root). Este ficheiro é gerado automaticamente na Makefile

Criptografar Dados ao Escrever

No servidor (server.c), as funções send_message e send_group_message foram modificadas para criptografar as mensagens antes de armazená-las nos arquivos messages.txt.

A função encrypt foi implementada para realizar a criptografia utilizando a chave AES.

```
int encrypt(unsigned char *plaintext, int plaintext_len, unsigned char *key,
        unsigned char *iv, unsigned char *ciphertext);
```

Esta função utiliza a biblioteca OpenSSL para realizar a criptografia com o algoritmo AES-256-CBC.

Descriptografar Dados ao Ler:

No cliente (client.c), as funções read_messages e read_group_messages foram modificadas para descriptografar as mensagens ao lê-las dos arquivos messages.txt.

A função decrypt foi implementada para realizar a descriptografia utilizando a chave AES.

```
int decrypt(unsigned char *ciphertext, int ciphertext_len, unsigned char *key,
        unsigned char *iv, unsigned char *plaintext);
```

Esta função também utiliza a biblioteca OpenSSL para realizar a descriptografia com o algoritmo AES-256-CBC.

4.2.2 Desafios na Implementação

Durante a implementação, enfrentamos desafios relacionados à configuração de permissões para garantir que os usuários possam acessar o arquivo aes_key para descriptografar as mensagens. Idealmente, isso envolveria a criação de um grupo no sistema Linux ao qual todos os usuários ativados pertencem. Devido à limitação de tempo, esta parte da implementação não foi concluída. Este passo é essencial para garantir que os usuários possam descriptografar as mensagens de forma segura e eficiente.

5 Funcionalidades em Falta e Trabalho Futuro

Em primeiro lugar, é necessário abordar aquilo que foi mencionado ao longo do relatório e que não foi de facto implementado completamente, nomeadamente a criptografia das mensagens. É utilizada a palavra completamente porque na realidade o código para a encriptar as mensagens encontra-se de facto nos ficheiros do Servidor e do Cliente. No entanto, por falta de tempo, era necessária a tal criação do Grupo Djumbai que contivesse todos os Utilizadores ativos na aplicação, uma vez que estes teriam que aceder ao ficheiro aes_key para descriptografar as suas mensagens, quer as mensagens contidas no seu próprio ficheiro como nos ficheiros de mensagens dos grupos a que os mesmos pertençam.

A outra funcionalidade de grande importância mencionada no enunciado era a comunicação síncrona, algo que acabei por tentar implementar mas que, tendo em conta o grande atraso na entrega e o ter que acabar as outras funcionalidades, acabou por não se feito. A ideia era ter a criação de duas threads do lado do Cliente, uma thread principal que fizesse a gestão da interação do Utilizador que inicia a aplicação, para o envio e leitura de mensagens, e uma outra thread que fosse responsável por ficar à escuta de mensagens vindas do Servidor. Do lado do Servidor, era criada uma thread para lidar com a comunicação de cada Utilizador. Assim que um Utilizador enviasse uma mensagem para outro, em que este também estivesse conectado à Aplicação do Cliente, a mensagem seria enviada através da thread criada na conexão inicial efetuada com o Utilizador a quem a mensagem se destina, sendo esta mensagem tratada no lado do Utilizador e apresentada na consola.

Fora isto, penso que o principal seria utilizar algum tempo para rever a arquitetura do sistema, uma vez que esta foi sendo alterada ao longo do tempo, não só pela falta de entendimento que tive do enunciado no início da realização deste trabalho, como do pouco tempo que acabei por ter para pensar e desenhar o sistema de início a fim e resolver alguns bugs que acabaram por ficar na aplicação.

6 Conclusão

O desenvolvimento deste sistema de mensagens local apresentou um conjunto de desafios e aprendizagens significativas, tanto em termos de funcionalidades quanto de segurança. A solução implementada seguiu uma arquitetura Cliente-Servidor que permite a comunicação eficiente entre Utilizadores e grupos, garantindo a integridade e confidencialidade das mensagens através de criptografia.

Pontos Fundamentais das Decisões Tomadas

- Arquitetura Cliente-Servidor - A escolha desta arquitetura permitiu uma clara separação de responsabilidades, facilitando a escalabilidade e manutenção do sistema.
- Criptografia: Implementar a criptografia AES para proteger as mensagens armazenadas foi crucial para garantir a confidencialidade dos dados, mesmo em caso de acesso indevido aos arquivos.
- Gestão de Permissões: Definimos permissões rigorosas para diretórios e arquivos, assegurando que apenas os Utilizadores autorizados possam aceder e modificar os dados.
-

Limitações e Trabalho Futuro

- Comunicação Síncrona - A implementação da leitura das mensagens recebidas em tempo real apresentou desafios significativos e foi removida na fase final do projeto.
- Implementação finalizada da Criptografia - Apesar da implementação da estrutura base da criptografia, ficou a faltar mais algum desenvolvimento para finalizar esta parte.

Este projeto demonstrou a importância de um design cuidadoso e da implementação de práticas de segurança rigorosas no desenvolvimento de sistemas de comunicação, especialmente no que diz respeito à gestão de permissões no desenvolvimento de aplicações para o sistema Linux. Apesar das limitações e desafios enfrentados, a solução desenvolvida atende à maioria dos requisitos funcionais e de segurança inicialmente propostos, oferecendo uma base sólida para futuras expansões e melhorias.