
TRABALHO PRÁTICO 3 *Tecnologias de Segurança*

TECHNICAL REPORT

① **Ivo Miguel Alves Ribeiro**
Mestrado em Engenharia Informática
Universidade do Minho
Pg53886
pg53886@alunos.uminho.pt

② **Henrique Ribeiro Fernandes**
Mestrado Integrado em Engenharia Informática
Universidade do Minho
A95323
a95323@alunos.uminho.pt

12 de maio de 2024

ABSTRACT

Este projeto tem como objetivo a conceção e análise de requisitos para o desenvolvimento de um serviço de conversação entre utilizadores locais de um sistema *Linux*, proporcionando aos utilizadores uma forma confiável de trocar mensagens com garantias de autenticidade, integridade e fiabilidade do sistema. Sendo um projeto desenvolvido no âmbito da cadeira de Tecnologias de Segurança, iremos ao máximo focar a nossa implementação nas métricas até aqui discutidas e abordadas nas aulas teóricas e teórico-práticas, como usufruir das vantagens da utilização de grupos *Linux* prioridades de diretorias perante grupos e utilizadores entre outros. Iremos ainda usufruir de algoritmos de encriptação e desencriptação de mensagens, técnicas de comunicação seguras como *TLS* e boas práticas também mencionadas nesta e noutras UCs.

Keywords análise de requisitos · serviço de conversação entre utilizadores locais · garantias de autenticidade, integridade e fiabilidade do sistema

1 Introdução

O projeto prático pode ser decomposto em duas partes, sendo uma a implementação de uma solução para um serviço de troca de mensagens seguras entre utilizadores *Linux*, e uma segunda fase de análise a essa mesma solução por nós desenvolvida.

Como solução prática iremos desenvolver um sistema Cliente/Servidor, onde a **aplicação cliente** permitirá aos utilizadores interagir com o sistema através de uma interface gráfica simples, via *bash*, que permitirá enviar mensagens a outros clientes, ver mensagens para ele enviadas, entrar em grupos e ainda proporcionar uma conversação em tempo real com um outro utilizador de sistema.

Já o **servidor** será responsável por gerir as solicitações dos utilizadores, manter o estado da aplicação e gerir toda a informação bem como a forma e local onde essa informação será guardada. Estas duas entidades irão comunicar sob um protocolo de mensagens seguras e cifradas que iremos implementar para garantir a conformidade das mensagens trocadas.

Implementamos ainda um serviço que nomeamos de **MasterUser** que é um serviço responsável pelas ações de *root* necessárias para criar/remover utilizadores de sistema, grupos *Linux* e ainda conceber permissões a certas diretorias.

2 Notas introdutórias

Devido ao curto período de tempo disponível para a realização deste trabalho prático decidimos optar por reutilizar um projeto já elaborado para uma outra unidade curricular, que tinha um objetivo semelhante, no caso, um serviço de troca de mensagens seguras entre clientes. A princípio achamos uma boa ideia pois o serviço em si parecia semelhante, mas com o decorrer da elaboração deste projeto prático percebemos que talvez uma linguagem como *C* ou *C++*, linguagens mais voltadas para sistema *linux* e com bibliotecas já preparadas para leitura e escrita em ficheiros com permissões de acesso pré-definidas, porem seguimos com o *python*.

Contudo destacamos que para um sistema destes ser colocado em ambiente de produção o indicado seria mesmo um serviço desenvolvido em *C* ou *C++* uma ofereçam um alto desempenho e controlo preciso, porem seria necessário mais tempo para evitar erros de segurança, como estouro de *buffer* e vazamentos de memória.

Para além disso houve um pequeno erro de leitura do enunciado, só percebido a uns dias da entrega, onde não nos apercebemos que era assumido que os utilizadores já existissem e então perdemos um bocado mais de tempo e recursos para implementar um serviço de criação de utilizadores de sistema incluído como funcionalidade do serviço.

3 Descrição do Problema

O projeto consistirá no desenvolvimento de um sistema Cliente/Servidor, em que a **aplicação cliente** será executada por cada utilizador para aceder à funcionalidade de enviar, iniciar uma conversação síncrona e receber mensagens de forma segura, conversação assíncrona, tipo email.

O **servidor** será responsável por responder às solicitações dos utilizadores como receber uma mensagem e direcionada ao seu legítimo recetor, bem como estabelecer comunicações com o **MasterUser** para garantir a execução das operações em modo *root*, visto que este servidor será também um utilizador do sistema de modo a garantir uma confiabilidade extra aos utilizadores deste serviço.

O **MasterUser** como referido anteriormente é responsável por receber pedidos do servidor e executar funções de *root* bem como, criar utilizadores de sistema, grupos, e conceber permissões de acesso a determinada diretoria.

O sistema como um todo deve respeitar questões como confidencialidade, autenticidade e integridade de todas as comunicações, e ainda garantir que o servidor não consegue manipular as mensagens trocadas pelos clientes. Para além das conversações síncronas e assíncronas o serviço deverá também suportar a gestão de utilizadores do serviço e ainda deverá oferecer mecanismos para criação de grupos, remoção de grupos e de gestão dos seus membros.

4 Requisitos de Segurança Pretendidos

O foco deste projeto reside na garantia da confidencialidade, integridade e disponibilidade das mensagens trocadas entre os usuários, bem como na implementação de mecanismos de controle de acesso inspirados no modelo de segurança do sistema operacional *Linux*. Assim, optamos por desenvolver um serviço cliente/servidor, onde os utilizadores enviam mensagens, com um outro utilizador como destinatário, para o servidor. O servidor, por sua vez, é também um utilizador de sistema com permissões especiais e, principalmente não é um utilizador que pertence ao grupo a *root*, logo não tem permissões para exercer funções de super Utilizador.

Isto garante um sistema bem mais confiável e seguro uma vez que garantimos algum controlo adicional sobre estas execuções que podem apenas ser realizadas por pedidos ao *MasterUser*. Assim o Servidor é responsável por não só comunicar ao *MasterUser* sempre que alguma operação de super Utilizador necessita de acontecer, assim como armazena as mensagens recebidas em um arquivos localizados nas diretorias específicas e protegido, acessível apenas ao servidor e ao utilizador destinatário da mensagem.

Quanto às mensagens de grupo e mensagens síncronas a base destes serviços é muito semelhante, as mensagens são na mesma enviadas para o servidor e este guarda-as em ficheiros em diretorias específicas e com permissões específicas. Concebendo acesso apenas a membros de grupos no caso dos grupos e aos dois utilizadores em conversação síncrona no caso dos *LiveChats*.

A nossa abordagem foi inspirada na arquitetura modular e segura do *qmail*, porem bem mais simplista e não tão confiável para ambiente de produção. Ao adotar estas estratégias, garantimos que apenas o servidor e os legítimos recetores das mensagens tenham acesso à diretoria de armazenamento dessas mesmas mensagens, protegendo assim a confidencialidade das comunicações. Além disso, as permissões de arquivo foram configuradas de forma a limitar o acesso a outros usuários do sistema, fortalecendo ainda mais a segurança da solução.

5 Descrição das entidades

5.1 Aplicação ClienteStarter

A aplicação *ClienteStarter* foi projetada para atender aos requisitos de criar utilizadores de sistema *linux* capazes de usufruir do nosso serviço. Cumprindo com requisitos de segurança, incluindo a utilização de criptografia assimétrica, autenticação de clientes e servidor.

Podemos destacar as seguintes funcionalidades principais:

- **Registo de Utilizador:** Esta aplicação permite que novos usuários sejam registados no sistema fornecendo um nome de utilizador e uma senha. Durante o registo, os dados do usuário são criptografados e assinados antes de serem enviados ao servidor, garantindo a confidencialidade e integridade das informações.
- **Login de Utilizador:** Utilizadores registados podem fazer login no sistema fornecendo seu nome de usuário e senha. As credenciais do usuário são enviadas de forma segura ao servidor para autenticação.

Os dados enviados no registo são armazenados no servidor, garantindo que só o utilizador que sabe a sua palavra-passe pode entrar na aplicação e tirar partido do serviço. Caso a operação de login ou registo seja válida o utilizador é executada a aplicação cliente com na *bash* do utilizador respetivo.

A aplicação cliente-servidor descrita neste relatório oferece uma solução robusta e segura para comunicações entre usuários em um ambiente Linux. Ao adotar técnicas avançadas de criptografia, autenticação e controle de acesso, a aplicação garante a proteção das informações sensíveis e a confiabilidade do sistema como um todo.

5.2 Aplicação Cliente

A aplicação cliente permite aos utilizadores do nosso sistema uma troca de mensagens seguras, utilizando criptografia assimétrica e comunicação segura via sockets *SSL/TLS*.

O cliente comunica com um servidor centralizado para enviar mensagens de forma segura, visualizar mensagens para ele destinadas bem como mensagens nos grupos a que pertence e ainda participem também de *chats* ao vivo.

A segurança é garantida através de técnicas de criptografia e autenticação robustas em todas as mensagens enviadas para o servidor sob protocolo *TLS* com garantia de autenticidade das mensagens.

Podemos destacar as seguintes funcionalidades principais:

- **Registo e Autenticação:** Os utilizadores podem se registar na aplicação com um nome de utilizador e senha. A autenticação é realizada usando certificados digitais, garantindo a identidade do utilizador.
- **Envio de Mensagens:** Os utilizadores podem enviar mensagens para outros utilizadores de forma segura. As mensagens são criptografadas e assinadas digitalmente para garantir a confidencialidade e autenticidade. São enviadas para o servidor onde serão direcionadas e armazenadas.
- **Caixa de Entrada Segura:** Os utilizadores podem verificar sua caixa de entrada para novas mensagens. Todas estas mensagens armazenadas em diretorias nas quais o utilizador tem acesso e permissão de leitura. As mensagens são descriptografadas apenas pelo destinatário autorizado, garantindo a confidencialidade dos dados.
- **Chats ao Vivo:** Os utilizadores podem participar de *chats* ao vivo de forma segura. A comunicação é criptografada no envio para o servidor, garantindo a confidencialidade das conversas uma vez que apenas os utilizadores que acordaram o *livechat* podem abrir o ficheiro onde o servidor esta a armazenar o conteúdo recebido.
- **Gerenciamento de Grupos:** Os utilizadores podem criar e/ou entrar em grupos de mensagens. As mensagens de grupo são tratadas com segurança semelhante às mensagens individuais.

Todas as mensagens são criptografadas usando algoritmos robustos, garantindo que apenas o destinatário pretendido possa ler o conteúdo. A autenticidade dos utilizadores é verificada usando certificados digitais, evitando qualquer tipo de falsificação ou *spoofing*.

Quanto ao gerenciamento de chaves publicas, neste momento as chaves são introduzidas ao iniciar a aplicação, porem para um ambiente de produção é aconselhável um gerenciamento de chaves sob a autoria de um autoridade certificadora externa que tornaria todas as trocas de mensagens mais seguras bem como garantia uma confiabilidade extra e para evitar fugas de informação ou uso não autorizado.

São de realçar as técnicas de segurança como *Encriptação Assimétrica*, onde as mensagens são cifradas com a chave pública do destinatário, que garantem que apenas o destinatário possa ler a mensagem; como *Assinaturas Digitais*, onde as mensagens são assinadas com a chave privada do remetente, que garantem a autenticidade e integridade da mensagem, e ainda *Comunicação Segura*, onde o cliente se conecta ao servidor por uma conexão *SSL/TLS*, o que garante que todas as comunicações estejam mais seguras e protegidas.

Assim, achamos que a aplicação cliente desenvolvida oferece uma solução robusta e segura para a troca de mensagens entre utilizadores do sistema, garantindo confidencialidade, autenticidade e integridade das comunicações.

5.3 Aplicação Servidor

A aplicação servidor é responsável por gerir as conexões dos clientes, receber mensagens de forma segura, gerir/armazenar as mensagens recebidas. Esta opera em um modelo centralizado, onde os clientes se conectam para enviar mensagens, foi construída com foco na segurança, usando criptografia para proteger as comunicações e garantir a integridade dos dados.

A aplicação do servidor seguro oferece uma plataforma para registo de utilizadores, criação e gerenciamento de grupos, troca de mensagens seguras e recursos avançados, como bate-papo ao vivo entre utilizadores. Ela opera em um ambiente distribuído, onde os clientes se conectam ao servidor para interagir e realizar diversas operações.

Assim como na aplicação cliente, todas as trocas de mensagens são seguras e tiram partido de criptografia assimétrica e comunicação segura via sockets *SSL/TLS*.

Podemos destacar as seguintes funcionalidades principais:

- **Registo de utilizadores:** Os utilizadores podem se registar na aplicação, fornecendo credenciais seguras. Caso não esteja registado o servidor pode registar ele próprio faz um pedido ao *MasterUser* para a criação de um novo utilizador de sistema e adiciona-lo ao serviço.
- **Registo de Grupos:** A aplicação permite responder a pedidos de criação e gestão de grupos para facilitar a comunicação entre os membros. Todas as ações de criação e inserção em grupos são retransmitidas do servidor para o *MasterUser*.
- **Comunicação Segura:** A aplicação utiliza criptografia para garantir a segurança das comunicações entre clientes e servidor. Isso inclui a troca de mensagens, autenticação de utilizadores e proteção contra ataques de intermediários.
- **Controle de Permissões:** O servidor controla as permissões de acesso dos utilizadores e grupos a recursos específicos. Para cada utilizador é criado um grupo de sistema que é responsável por-lhe conceber acesso aos recursos a ele destinados. Quanto aos grupos e às conversações assíncronas o serviço é muito semelhante concebendo acesso a esses recursos aos utilizadores que a esses grupos pertencem. Isso garante que apenas utilizadores autorizados possam acessar determinadas dados.
- **Live-Chat ao Vivo:** A aplicação suporta conversação assíncrona em tempo real entre utilizadores, permitindo que eles se comuniquem de forma instantânea e segura. Dois utilizadores realizam um acordo mutuo de que querem fazer um *LiveChat* e o servidor pede para criar um grupo próprio e conceber acesso de leitura aos dois utilizadores e assim se estabelece esta comunicação em direto.

Assim como na aplicação cliente, tiramos partido de técnicas como *Encriptação Assimétrica* das mensagens e *Assinaturas Digitais* para garantir uma autenticidade e integridade dos conteúdos trocados. Para além disso, beneficiamos da *Autenticação de Clientes*, onde o servidor verifica a autenticidade das mensagens recebidas, garantindo que apenas clientes legítimos possam interagir com o sistema e ainda *Proteção das Mensagens trocadas*, onde as mensagens enviadas são guardadas em diretorias que apenas o servidor e o legítimo recetor tem permissões de acesso, protegendo-as contra acesso e manipulação não autorizado.

Relativamente ao armazenamento de dados, como referido acima, as mensagens são armazenadas em diretorias específicas para cada utilizador, nas quais apenas o servidor e o legítimo utilizador de sistema tem permissões de acesso. Para além disso, os metadados das mesmas são armazenados num ficheiro de *Log*, permitindo auditoria e mapeamento das atividades e dos dados do sistema.

Assim, achamos que servidor desenvolvido oferece uma solução sólida e segura para comunicação entre clientes, garantindo confidencialidade, autenticidade e integridade das mensagens trocadas.

5.4 Aplicação MasterUser

MasterUser é um aplicação de controle centralizado, projetada para gerir utilizadores e grupos em um ambiente Linux. A aplicação foi desenvolvida para automatizar tarefas relacionadas à criação, modificação e remoção de utilizadores e grupos, oferecendo um controle simplificado e centralizado sobre as permissões e acessos do sistema.

A aplicação MasterUser opera como um servidor central que recebe comandos do servidor e executa ações específicas com base nesses comandos. Ela oferece funcionalidades para criar utilizadores e grupos de sistema, adicionar utilizadores a grupos, definir permissões de acesso a diretorias e arquivos, e remover utilizadores e grupos do sistema. A aplicação utiliza comunicação via sockets para receber e processar os comandos do servidor de forma eficiente, porem não muito segura.

Podemos destacar as seguintes funcionalidades principais:

- Criação de utilizadores: A aplicação permite a criação de novos utilizadores no sistema Linux. Isso é feito de forma automatizada, garantindo que os novos utilizadores tenham acesso aos recursos necessários e sejam adicionados aos grupos apropriados.
- Criação de grupos: A aplicação permite a criação de novos grupos no sistema Linux. Normalmente seguido da execução de definir permissões de acesso a uma determinada diretoria para este grupo em específico.
- Gerir Acesso: A aplicação oferece recursos para adicionar utilizadores a grupos específicos, definir permissões de acesso a diretorias e arquivos para grupos e/ou utilizadores individuais. Esta fase permite garantir praticas seguras contra a manipulação dos dados no sistema uma vez que o acesso é restringido aos legítimos recetores dos dados.
- Remoção de Utilizadores e Grupos: A aplicação permite remover utilizadores e grupos do sistema conforme necessário. Isso simplifica o processo de gerir acessos e garante a segurança e integridade do sistema.
- Comunicação Servidor-MasterUser: A comunicação entre servidor e o servidor *MasterUser* é estabelecida por meio de sockets *TCP/IP*. Isso permite que os clientes enviem comandos para o servidor de forma remota e recebam *feedback* sobre o status das operações executadas.

A segurança é uma consideração importante na implementação da aplicação *MasterUser*, no entanto antes de colocar este serviço em ambiente de produção medidas de segurança extra devem ser tomadas. Medidas como garantir que apenas utilizadores autorizados (servidor no nosso caso) possam executar operações que afetam a segurança do sistema, promovendo técnicas de garantia de autenticidade das mensagens trocadas entre servidor e *MasterUser*, nomeadamente assinaturas digitais e validação de certificados com um autoridade certificadora externa, um controlo das atividades deste servidor, de momento existe as ações nele executados promovem uma mensagem na *bash* que executa o mesmo mas de futuro um ficheiro de *logs* com identificação das ações e de quem efetuou o pedido das mesma seria o mais indicado.

A aplicação *MasterUser* oferece uma solução eficaz e conveniente para gerir as operações de *root* necessárias para o bom funcionamento do nosso serviço. A sua interface simplificada e recursos abrangentes tornam o processo de administração do sistema mais eficiente.

5.5 Protocolo de Comunicação

O sistema em questão requer uma robusta estrutura de comunicação, onde as mensagens são transmitidas e recebidas através de conexões serializadas. Essa serialização e futura desserialização é essencial para garantir a integridade, autenticidade e confidencialidade das informações transmitidas entre as diferentes componentes do sistema.

O mecanismo de serialização consiste em transformar mensagens em formato *JSON* numa cifra, pronta para transmissão através da rede, enquanto o de desserialização consiste na reconstrução das mensagens à sua forma original, para que possam ser interpretadas e utilizadas pelos destinatários, a partir dos dados cifrados recebidos. Tanto a serialização como a desserialização são processos críticos que envolvem várias etapas, desde a assinatura do conteúdo até a cifragem e encapsulamento da mensagem.

O Processo de Serialização consiste em:

- Assinatura do Conteúdo: O primeiro passo é assinar o conteúdo da mensagem realizado pela função *Sign*, que utiliza a chave privada do emissor para gerar uma assinatura única do conteúdo.

- **Encriptação do Conteúdo:** Após a assinatura, o conteúdo da mensagem é cifrado utilizando a chave pública do recetor da mensagem em questão. Esta encriptação é usada como garantia extra de que o conteúdo está de alguma maneira mais seguro do que o resto da mensagem, uma vez que é nele que circula a informação secreta.
- **Validação de Injeção de JSON:** Antes de prosseguir, é realizada uma validação para garantir que a mensagem não contém injeções de JSON maliciosas ou inválidas. Essa etapa é uma implementação adicional ao problema do enunciado, mas achamos crucial para evitar vulnerabilidades de segurança, visto que este assunto já foi abordado nas aulas.
- **Conversão da mensagem para JSON:** A estrutura da mensagem é montada com todas as informações necessárias, serializada para o formato JSON que, por fim, é então codificada em *Base64* para garantir uma representação segura e compatível com a nossa função de encriptação.
- **Encriptação da Mensagem serializada:** O resultado da conversão é então cifrado novamente utilizando a chave pública do recetor, o que garante uma camada adicional de segurança durante a transmissão.
- **Assinatura da Cifra:** Finalmente, a mensagem encriptada é assinada utilizando a chave privada do emissor. Esta assinatura é concatenada com a cifra da mensagem, formando o pacote final a ser transmitido, que na sua receção pode ser validado rapidamente.

Relativamente ao Processo de Desserialização, este consiste em:

- **Divisão da Cifra e Assinatura:** A primeira etapa consiste em separar a assinatura da mensagem e a própria cifra da mensagem. Estes dois argumentos serão usados no final do processo juntamente com a chave pública do emissor para uma validação da mesma.
- **Desencriptação da Mensagem:** O *ciphertext* é decifrado utilizando a chave privada do destinatário, o que devolve a mensagem serializada à sua forma original.
- **Descodificação *Base64* e Conversão para JSON:** A sequência de bytes decifrada é descodificada a partir de *Base64*, recuperando assim a representação JSON original da mensagem.
- **Desencriptação do Conteúdo da Mensagem:** O conteúdo da mensagem, que foi cifrado durante a serialização, é decifrado utilizando novamente a chave privada do destinatário. Caso a mensagem tenha como destinatário o próprio recetor, o conteúdo ficará claro.
- **Validação de Injeção de JSON:** Antes de concluir o processo de desserialização, é realizada uma verificação adicional para garantir que a mensagem não contém injeções de JSON maliciosas ou inválidas.
- **Obtenção da Chave Pública do Remetente:** A chave pública do remetente da mensagem é carregada a partir da representação PEM fornecida no pacote da mensagem como certificado do emissor.
- **Verificação da Assinatura:** Finalmente, a função *Verify* é utilizada para verificar a autenticidade da mensagem, com recurso à assinatura recebida, à cifra da mensagem e à chave pública do remetente, garantindo que a mensagem não tenha sido alterada durante a transmissão e que realmente tenha sido enviada pelo remetente alegado.

Ao seguir uma abordagem sistemática, que inclui assinatura, encriptação e validação, assegura-se que as mensagens sejam transmitidas de forma confiável e segura entre os componentes do sistema.

5.6 Estratégia de Encriptação

A segurança das comunicações é um aspeto crítico em sistemas distribuídos, especialmente quando se trata da troca de informações sensíveis.

Assim como suscitado no enunciado, desenvolvemos mecanismos de encriptação para as mensagens partilhadas pelo nosso sistema, que associamos ao nosso protocolo de comunicação. Porém, decidimos destacar e descrever o funcionamento das mesmas para uma maior familiarização com o sistema.

Atendendo ao fornecido no enunciado, os certificados permitiam definir uma chave pública e uma chave privada, ambas RSA, a cada utilizador do sistema, porém estas chaves associadas a mecanismos de encriptação também RSA apresentam algumas limitações práticas para grandes blocos de dados que queiramos que sejam cifrados. Como decidimos no protocolo de comunicação que temos uma dimensão razoável para cada mensagem, atendendo aos campos obrigatórios da mesma, decidimos implementar uma técnica de criptografia híbrida, já conhecida, onde usamos RSA para cifrar a chave simétrica (AES), e usamos essa chave para encriptar os dados reais de forma mais eficiente.

Função de Encriptação:

A função *encrypt* recebe o conteúdo a ser cifrado e a chave pública *RSA* do destinatário.

- Criação de chave *AES*: É gerada uma chave *AES* (*Advanced Encryption Standard*) aleatória com 256 bits de comprimento e um vetor de inicialização (*iv*) com 128 bits de comprimento.
- Encriptação *AES*: De seguida, com os argumentos gerados, cria um objeto de cifra com o modo de operação *GCM* (*Galois/Counter Mode*), que é usado para construir a cifra *AES* a partir do conteúdo original.
- Encriptação *RSA*: Posteriormente, ciframos a chave *AES* segundo o algoritmo *OAEP* (*Optimal Asymmetric Encryption Padding*) com recurso à chave pública *RSA* do destinatário do conteúdo.
- Codificação dos componentes: Por fim, são combinados todos os componentes (chave *AES*, *IV*, tag de autenticação e texto cifrado) numa mensagem cifrada, que é codificada em *Base64* para facilitar a transmissão segura pela rede, retornando a mensagem cifrada pronta para ser transmitida ao destinatário.

Função de Desencriptação:

A função *decrypt* recebe a cifra completa na forma de uma mensagem e a chave privada *RSA* do destinatário.

- Descodificação da mensagem cifrada: Descodifica a mensagem cifrada de *Base64* para recuperar cada um dos componentes. Posteriormente, extrai a chave *AES* criptografada, o *IV*, o tag de autenticação e o texto cifrado da mensagem.
- Desencriptação da chave *AES*: Utiliza a chave privada *RSA* do destinatário para Desencriptar a chave *AES* utilizando o algoritmo *OAEP*.
- Construção do objeto de cifra: De seguida, utiliza a chave *AES*, o *IV* e a tag de autenticação para criar um objeto de capaz de desencriptar a cifra em modo de operação *GCM*.
- Desencriptação *AES*: Por fim, desencripta o texto cifrado com recurso ao objeto criado e retorna o conteúdo original.

Ao utilizar algoritmos robustos e práticas de segurança adequadas, como a criptografia assimétrica e a criptografia *AES* com modos de operação seguros, é possível proteger efetivamente as comunicações contra ameaças de manipulação de dados.

6 Descrição das atividades

6.1 Arranque da aplicação Servidor

Ao iniciar a aplicação cliente dados como chave privada, chave publica e certificado são carregados, nesta fase manualmente porem em ambiente de produção o recomendado seria que fossem enviados e validados por uma autoridade certificadora, é ainda estabelecida uma ligação ao servidor *MasterUser* para que no futuro possam ser enviados pedidos de execução de funções de sistema, bem como é aberto um canal de comunicação ao qual os clientes se podem conectar para usufruir do serviço.

De seguida é enviado um pedido ao *MasterUser* de criação do grupo com o nome do servidor, seguido de um pedido de inserção do utilizador no grupo com o seu nome.

NOTA: O servidor deve ser um utilizador de sistema já criado, que pode ser criado com a aplicação *ClienteStarter* por exemplo e com o nome de 'server' e password 'root' para o nosso caso.

Após a criação do grupo é então criada a diretoria 'DataBase' que é a diretoria responsável por organizar todas as pastas relativas aos clientes do sistema, seguida da criação da diretoria com o nome do servidor à qual é pedido que seja vinculada ao grupo com o nome do servidor e permissões de leitura, escrita e execução para o dono (o server) e ainda leitura para o grupo com o nome do servidor, este passo permite que mais tarde um outro utilizar como um gestor de tráfego por exemplo, tenha acesso a leitura dos dados presentes nesta pasta para uma melhor monitorização do serviço.

Por fim o servidor fica a aguardar conexões com clientes e a cada conexão inicia um *thread* para lidar com o mesmo, assim garantimos a escalabilidade da nossa solução.

6.2 Registo Utilizador

A ação de registo de utilizador esta numerada no nosso sistema como "ação 0", e tem como principal função garantir que o cliente que entrou numa comunicação com o servidor é um utilizador de sistema válido e inserido no ambiente do sistema. Oferecemos esta garantia uma vez que um cliente necessita passar pela ação de registo, ou caso já seja um cliente registado necessita de passar por um *login*, para tirar partido das demais funcionalidades do sistema.

Caso se pretendido criar um utilizador de sistema capaz de tirar partido do nosso serviço recomendamos a execução da aplicação *ClienteStarter* uma vez que existe nela a opção de criar um novo utilizador. Esta opção de criar um novo utilizador consiste no envio de um pedido ao servidor de registo, este verifica se na base de dados existe alguma diretoria com o nome deste cliente, caso exista é retornada ao cliente essa informação, indicando que é necessário o envio das credenciais de acesso para legitima validação do utilizador. Caso não exista é enviado um pedido ao *MasterUser* para que ele crie um novo utilizador de sistema *Linux*, de seguida é enviado um outro pedido ao *MasterUser* para criação de um grupo com o nome deste mesmo cliente que se segue de um outro pedido, o de inserção deste utilizador recém criado ao grupo com o seu nome.

NOTA: A ação de criar um novo utilizador de sistema no *MasterUser* é acompanhada da inserção do mesmo num grupo denominado **code** que é o grupo responsável por conceber permissões de leitura, escrita e execução na diretoria onde está o código das aplicações do sistema. Este grupo é criado no arranque da aplicação *MasterUser* para garantir que todos os novos utilizadores de sistema tenham acesso às aplicações. Num ambiente de produção medidas como permissões de acesso apenas às aplicações clientes para os clientes e apenas acesso às aplicações servidor pelos servidores seriam indicadas.

Após a criação do utilizador e do grupo, o servidor criada a diretoria com o nome do cliente e ainda um ficheiro *csv* nela presente para auxiliar na identificação de mensagens lidas ao não lidas.

Por fim, mas não menos importante, é enviado um pedido ao *MasterUser* para definir o grupo do cliente como o grupo sob diretoria como seu nome, bem como permissões de leitura, escrita e execução para o servidor, e apenas leitura e execução para o cliente, finalmente a password é registado no sistema e armazenada para futura verificação *login*.

Estas permissões foram assim definidas numa primeira análise ao sistema, porem existem aspetos a realçar. O servidor apresenta-se com mais permissões sobre a diretoria de um utilizador do que o proprio utilizador, isto dá-se porque a ideia do grupo foi o servidor é que gere o ficheiro de *csv* com os metadados das mensagens (como mensagens lidas ou não lidas), e o cliente apenas com permissões de leitura e execução pois apenas iria ler o conteúdo dos ficheiros que corresponderia a mensagens a ele destinadas.

Contudo é necessário perceber que adotamos uma postura de servidor não malicioso, que para um ambiente de produção não é de todo o mais indicado. Medidas como restringir ao máximo o acesso à manipulação dos dados pelo servidor e ainda garantir o mesmo da parte dos clientes, seria o mais indicado para um sistema como este em ambiente de produção, destacando uma vez mais que o *python* não é recomendado uma vez que erros de *permission denied* foram comuns quando não concebemos permissões de execução a ambos os intervenientes.

6.3 Criar e ingressar em Grupo

A nossa solução divide estes dois processos em duas ações, impedindo a inserção de um utilizador em um grupo que não exista e permitindo que um utilizador possa sempre que pretender criar um grupo.

O processo de criação de um grupo consiste em um pedido ao servidor *MasterUser* para a criação de um grupo de sistema *Linux* que caso tenha um nome válido retorna sucesso da operação e então enviado um novo pedido o de inserção do utilizador que ordenou a sua criação no grupo. Já o processo de ingressar em um grupo consiste na verificação da existência desse grupo, que caso exista é então enviado um pedido ao servidor *MasterUser* para a inserção deste mesmo utilizador no grupo.

A nossa solução consiste em um ambiente de comunidades onde um cliente pode criar uma comunidade para partilhar e discutir sobre um assunto e um outro qualquer utilizador pode ingressar e comentar sobre nesse grupo, se assim o pretender.

Para um ambiente de produção mais funcional seria importante definir alternativas de gestão de grupos como grupos privados onde só seria possível entrar com uma password ou ainda determinar um utilizador com administrador do grupo e capaz de remover e adicionar utilizadores ao mesmo.

6.4 Envio de mensagens (Cliente-Servidor)

O processo de envio de mensagens é de longe o processo mais seguro presente na nossa solução. Este processo é iniciado no cliente este indica o destinatário da mensagem, no caso pode ser um outro cliente ou um grupo do nosso sistema, de seguida o assunto da mensagem e por fim o conteúdo da mesma.

Antes de ser enviada o conteúdo da mensagem é cifrado e assinada com a chave publica do recetor da mesma garantindo que apenas o legítimo recetor consegue decifrar e visualizar o mesmo, não satisfeitos decidimos cifrar e assinar a mensagem como um todo com a chave publica do servidor e só depois é enviada segundo protocolo de comunicação *TLS* com autenticação segundo certificados digitais.

Assim que recebida no servidor a mensagem é então validada e desserializada e decifrada com recurso à chave privada do servidor. São consultados então os metadados da mesma que foram decifrados e é passado então a totalidade da mensagem para um novo ficheiro na diretoria da base dados relativa ao recetor da mensagem.

Este processo não é de todo a implementação perfeita e existe a probabilidade de existir problemas de concorrência e segurança se colocado em um ambiente de produção, daí deixarmos um alerta para essa fator. A nossa alternativa apenas incrementa o numero anterior de mensagem de nomeia o novo ficheiro com esse numero mas para o caso de duas mensagens estarem a ser escritas ao mesmo tempo pode levantar alguns problemas. Para tal alguma solução como nomear o ficheiro com uma combinação única recorrendo a dados como *timestamp*, *id da thread* e *id* do emissor da mensagem seria a melhor opção em ambiente de produção.

6.5 Visualização de Mensagens

O processo de visualização de mensagens consiste na iteração das diretorias à qual o utilizador tem permissões de acesso para leitura, percorrendo todos os ficheiros nele presentes que não estejam no ficheiro *csv* dessa diretoria como lidos. Assim que um desses ficheiros seja aberto por parte de um cliente essa informação é enviada ao servidor de modo a que este coloque/altere essa informação no ficheiro *csv*.

Este processo funciona para ambos os casos, mensagens a ele direcionadas e grupos de conversa, contudo as mensagens a ele destinadas necessitam de um descriptação e validado do conteúdo com a própria chave privada para que o conteúdo original seja obtido, garantindo assim um fator de segurança extra nesta troca de mensagens.

6.6 Modo LiveChat

O processo de conversação síncrona é realizado em etapas, a princípio dois utilizadores seleccionam a opção de *LiveChat*, o primeiro obviamente vai receber a notificação do servidor de que não existem utilizadores disponíveis para conversas em tempo real, porém o segundo já irá receber a informação que o primeiro se encontra disponível e então irá mandar um pedido ao mesmo dizendo que pretende iniciar uma conversa. O outro utilizador receberá então essa mensagem e pode então aceitar essa comunicação, quando aceite o servidor cria uma nova diretoria com um ficheiro capaz de armazenar todas as mensagens. Com a criação desta diretoria dá-se um pedido ao *MasterUser* para que seja criado um grupo de sistema com o mesmo nome e que ambos os utilizadores sejam adicionados a esse grupo. Por fim indica ambos os utilizadores que iniciou uma conversa em tempo real permitindo que ambos após um refresh da página possam começar a conversa.

Assim que um utilizador deixa esta conversa o servidor trata de avisar o outro desse acontecimento e termina este processo ao enviar ao *MasterUser* um pedido de remover o grupo criado para esta conversa bem como a diretoria e todos os ficheiros nela presentes.

Em ambiente de produção pode ser benéfico podermos voltar e reestabelecer o conteúdo desta conversa, mas na nossa solução não optamos por esse trabalho acrescido.

6.7 Remoção de Utilizador e Grupo

O processo de remoção de utilizador e grupo é bastante simples e pode ser realizado como um pedido do cliente ao servidor para remoção dele próprio do sistema como usado pelo servidor para gestão de recursos noutras atividades como na conversação síncrona. Este processo consiste num pedido ao servidor *MasterUser* para que um utilizador ou grupo sejam eliminados do sistema. Um pedido de remoção de utilizador segue-se de um pedido de eliminar o grupo com o nome do mesmo bem como a diretoria na base de dados relativa.

7 Análise de Risco

A análise de risco é uma parte crucial do processo de segurança da informação, pois ajuda a identificar e avaliar as ameaças potenciais que podem afetar um sistema ou aplicação. Neste contexto, utilizamos o método STRIDE para analisar diferentes aspectos de segurança relacionados com a aplicação cliente e servidor.

7.1 Aplicação Cliente

7.1.1 Modelação de Ameaças Orientado ao Software

7.1.1.1 Spoofing

Fraqueza: Sistema linux deixa admins fazerem se passar por outros utilizadores.

Ameaça: A ameaça associada a essa fraqueza é a possibilidade de um invasor realizar um ataque de spoofing e se passar por outro usuário para acessar suas mensagens. Ao comprometer a identidade de um usuário legítimo, o invasor pode obter acesso às informações confidenciais contidas nas mensagens, comprometendo assim a privacidade e a segurança dos dados do usuário.

Forma de Mitigação: Para mitigar os riscos relacionados com o spfing sobre utilizadores , a principal medida é implementar a cifragem de todas as suas informações. Isto pode ser feito utilizando algoritmos de cifragem robustos para proteger os arquivos

Análise de risco: Esta fraqueza está prevista nos requisitos e para ter acesso é preciso ter permissões de superutilizador , pelo que apresenta um risco baixo. Quanto ao seu impacto, este é alto, visto que os ficheiros de mensagem podem ser reveladas informações sensíveis .

- RISCO: 1
- IMPACTO: 3
- CRITICIDADE: $3 + 1 = 4$

7.1.1.2 Tampering

Fraqueza: Ficheiros de mensagens de grupo não cifrados.

Ameaças: A principal ameaça relacionada a arquivos não cifrados é a violação de privacidade e segurança. Quando os arquivos não são cifrados, estão vulneráveis a acessos não autorizados. Isto significa que qualquer pessoa com acesso ao sistema onde os arquivos estão armazenados pode visualizá-los, copiá-los ou modificá-los sem restrições.

Forma de mitigação: Para mitigar os riscos relacionados com arquivos não cifrados, a principal medida é implementar a cifragem de arquivos. Isto pode ser feito utilizando algoritmos de cifragem robustos para proteger os arquivos.

Análise de risco: Esta fraqueza está prevista nos requisitos e para ter acesso é preciso ter permissões de superutilizador , pelo que apresenta um risco baixo. Quanto ao seu impacto, este é alto, visto que os ficheiros de mensagem que sejam modificados pode revelar informações sensíveis .

- RISCO: 2
- IMPACTO: 5
- CRITICIDADE: $2 + 5 = 7$

7.1.1.3 Repudiation

Com recurso a assinatura digital, a protocolos como *GCM* do *AES* e a *logs* que so o servidor consegue modificar que contem a identificação dos emissores das mensagens e alocação de um *timestamp*, o não repúdio não é viável nesta aplicação.

7.1.1.4 Information Disclosure

Fraqueza: Ficheiros de mensagens de grupo não cifrados.

Ameaças: A principal ameaça relacionada a arquivos não cifrados é a violação de privacidade e segurança. Quando os arquivos não são cifrados, estão vulneráveis a acessos não autorizados. Isto significa que qualquer pessoa com acesso ao sistema onde os arquivos estão armazenados pode visualizá-los, copiá-los ou modificá-los sem restrições.

Forma de mitigação: Para mitigar os riscos relacionados com arquivos não cifrados, a principal medida é implementar a cifragem de arquivos. Isto pode ser feito utilizando algoritmos de cifragem robustos para proteger os arquivos.

Análise de risco: Esta fraqueza está prevista nos requisitos e para ter acesso é preciso ter permissões de superutilizador, pelo que apresenta um risco baixo. Quanto ao seu impacto, este é alto, visto que os ficheiros de mensagens podem conter informações sensíveis.

- RISCO: 2
- IMPACTO: 5
- CRITICIDADE: $2 + 5 = 6$

7.1.1.5 DoS

Os ataques de *Denial of Service* são realizados de forma a tornar os recursos de um sistema indisponíveis para os seus utilizadores. Normalmente os alvos típicos deste tipo de ataques são servidores, pois há necessidade de estes estarem sempre disponíveis para que os utilizadores consigam aceder a informação nele contida. No caso da *aplicação do cliente*, este tipo de ataques não seria preocupante pois uma aplicação não é geralmente alvo destes ataques visto que não há necessidade da *aplicação do cliente* estar sempre disponível para obtenção de informação. Estes ataques são geralmente direcionados para entidades como o *aplicação server*, pois há necessidade de este estar sempre disponível para comunicar com as entidades do sistema. Assim, estes ataques não são aplicados à entidade em questão.

7.1.1.6 Elevation of Privilege

Fraqueza: Cliente tentar aceder a uma pasta de um grupo que não tem permissões.

Ameaça: A ameaça associada a essa fraqueza é a tentativa de acesso não autorizado por parte de um utilizador à uma pasta de outro grupo. Ao tentar acessar a pasta sem as permissões adequadas, o usuário pode ser tentado a utilizar métodos não autorizados para obter acesso, comprometendo a integridade e a segurança do sistema..

Forma de Mitigação: Uma medida eficaz de segurança seria cifrar e assinar as mensagens contidas na pasta. Ao utilizar criptografia, as mensagens se tornam ilegíveis para os utilizadores não autorizados, mesmo que consigam acessar os arquivos. Além disso, a assinatura digital garante a autenticidade e integridade das mensagens, permitindo que apenas os destinatários autorizados possam decifrar e verificar a autenticidade das mensagens.

Análise de Risco: Esta fraqueza está prevista nos requisitos e para ter acesso é preciso ter permissões de superutilizador, pelo que apresenta um risco baixo. Quanto ao seu impacto, este é muito alto, visto que os ficheiros de mensagens podem conter informações sensíveis e este pode tanto vê-lo e modifica-lo.

- RISCO: 1
- IMPACTO: 3
- CRITICIDADE: $3 + 1 = 4$

7.2 Aplicação Servidor

7.2.1 Modelação de Ameaças Orientado ao Software

7.2.1.1 Spoofing

No contexto atual, não foram identificados problemas de spoofing no servidor em questão. Este servidor mantém uma postura robusta de segurança e não compartilha informações com terceiros nem tem nenhuma funcionalidade que consiga interferir com os outros utilizadores ou sistema. Portanto, não há preocupações imediatas relacionadas ao spoofing.

7.2.1.2 Tampering

Fraqueza: Modificar ficheiros de logs.

Ameaças: A ameaça associada a essa fraqueza é a tentativa de manipulação maliciosa dos arquivos de logs por parte de usuários não autorizados. Ao modificar os registros de atividades, um invasor pode ocultar evidências de suas ações, dificultando a detecção de atividades maliciosas e comprometendo a capacidade de resposta a incidentes de segurança.

Forma de mitigação: Para mitigar essa vulnerabilidade, é fundamental implementar medidas de segurança robustas para proteger os arquivos de logs. Isso pode incluir restrições de acesso adequadas aos arquivos de logs, garantindo que apenas usuários autorizados tenham permissão para visualizar, modificar ou excluir os registros. Além disso, é importante utilizar técnicas de criptografia e assinatura digital para proteger a integridade dos arquivos de logs. Ao cifrar os registros de atividades e assiná-los digitalmente, é possível garantir que qualquer tentativa de manipulação seja detectada e impedida.

Análise de risco: Esta fraqueza está prevista nos requisitos e para ter acesso é preciso ter permissões de superutilizador, pelo que apresenta um risco baixo. Quanto ao seu impacto, este é medeio, visto que os ficheiros que de logs permitem o não repúdio mas não tem informações sensíveis.

- RISCO: 1
- IMPACTO: 3
- CRITICIDADE: $1 + 3 = 4$

7.2.1.3 Repudiation

O problema de repúdio é mitigado pela existência de logs detalhados em nosso sistema. Os logs funcionam como um registro de todas as atividades realizadas pelos usuários. Isso significa que qualquer ação executada dentro do sistema deixa um rastro digital inequívoco, tornando difícil para os usuários negarem sua participação ou responsabilidade em uma determinada atividade. Portanto, podemos confiar nos logs como uma fonte confiável de evidências para resolver questões relacionadas ao repúdio.

7.2.1.4 Information disclosure

Não há preocupações com divulgação de informações, uma vez que o servidor não transmite as mensagens para os clientes. Em vez disso, o servidor quando recebe uma mensagem guarda na pasta do destinatário. Dessa forma, o servidor age como um ponto de controle seguro, garantindo que as informações sejam transmitidas apenas para os destinatários apropriados, sem o risco de divulgação não autorizada para outros clientes.

7.2.1.5 DoS

Fraqueza: Falta de controlo da quantidade de pedidos de dados.

Ameaças: Um adversário pode inundar o sistema com uma quantidade excessiva de solicitações, sobrecarregando o servidor e impedindo-o de atender às necessidades dos demais utilizadores. Além disso, a chegada dessas solicitações em grande quantidade pode aumentar significativamente o consumo de recursos de rede, resultando numa redução drástica no fluxo de comunicação com outras entidades e, conseqüentemente, diminuindo a disponibilidade do servidor.

Forma de Mitigação: Limitar a quantidade de pedidos sobre os quais o servidor está a realizar operações.

Análise de Risco: Esta fraqueza está prevista nos requisitos, pelo que apresenta um risco baixo. Quanto ao seu impacto, este é alto visto que retira a disponibilidade do sistema.

- RISCO: 1
- IMPACTO: 5
- CRITICIDADE: $1 + 5 = 6$

7.2.1.6 Elevation of privilege

Não há preocupações com elevação de privilégios, pois o servidor já possui permissões sobre todas as pastas relevantes. Isso significa que o servidor tem acesso adequado e autorizado para ler, gravar as pastas conforme necessário para suas operações. Portanto, não há risco dele conseguir acesso não autorizado a recursos adicionais através de técnicas de elevação de privilégios, já que o servidor já detém as permissões adequadas desde o início.

7.3 Aplicação Master

7.3.1 Modelação de Ameaças Orientado ao Software

7.3.1.1 Spoofing

No contexto atual, não foram identificados problemas de spoofing no master em questão. Este mantém uma postura robusta de segurança e não compartilha informações com terceiros nem tem nenhuma funcionalidade que consiga interferir com os outros utilizadores ou sistema. Portanto, não há preocupações imediatas relacionadas ao spoofing.

7.3.1.2 Tampering

Fraqueza: Transferência de dados não cifrados.

Ameaças: A ameaça neste contexto é a possibilidade de que os dados transferidos entre as entidades do sistema possam ser alterados ou manipulados por terceiros não autorizados. Isto pode levar a consequências graves, como a violação da integridade dos dados.

Forma de Mitigação: Para mitigar a vulnerabilidade da transferência de dados não cifrados entre entidades do sistema, é essencial implementar medidas de segurança robustas. Isto inclui o uso de criptografia para proteger a confidencialidade das informações e a adoção de certificados digitais para autenticar as partes envolvidas. Além disso, tecnologias seguras de comunicação, como SSL/TLS, devem ser utilizadas para estabelecer canais seguros. Ainda, mecanismos de validação de integridade, como *hash's*, garantem a integridade dos dados transmitidos.

Análise de Risco: Esta fraqueza está prevista nos requisitos pelo que apresenta um risco baixo. Quanto ao seu impacto este é alto pelo facto que podem dar permissões a utilizadores a grupos indevidos o que pode levar a roubo de informações sensíveis.

- RISCO: 1
- IMPACTO: 7
- CRITICIDADE: $1 + 7 = 7$

7.3.1.3 Repudiation

O problema de repúdio é mitigado pela existência de logs detalhados em nosso sistema. Os logs funcionam como um registro de todas as atividades realizadas pelo server. Isso significa que qualquer ação executada dentro do sistema deixa um rastro digital inequívoco, tornando difícil para os atacantes negarem sua participação ou responsabilidade em uma determinada atividade. Portanto, podemos confiar nos logs como uma fonte confiável de evidências para resolver questões relacionadas ao repúdio.

7.3.1.4 Information disclosure

O papel do master está limitado a criar grupos e pastas, bem como conceder permissões. Dessa forma, não há informações sensíveis ou confidenciais em posse do master para serem divulgadas. Sua função é estritamente administrativa, focada na organização e gestão das estruturas de grupos e pastas, e na atribuição apropriada de permissões aos usuários e aos grupos. Portanto, não há risco de divulgação de informações confidenciais por parte do master, pois ele não tem acesso a dados sensíveis além das configurações de grupos e permissões.

7.3.1.5 DoS

Fraqueza: Falta de controlo da quantidade de pedidos de dados.

Ameaças: Um adversário pode inundar o sistema com uma quantidade excessiva de solicitações, sobrecarregando o servidor e impedindo-o de atender às necessidades dos demais utilizadores. Além disso, a chegada dessas solicitações em grande quantidade pode aumentar significativamente o consumo de recursos de rede, resultando numa redução drástica no fluxo de comunicação com outras entidades e, consequentemente, diminuindo a disponibilidade do servidor.

Forma de Mitigação: Limitar a quantidade de pedidos sobre os quais o servidor está a realizar operações.

Análise de Risco: Esta fraqueza está prevista nos requisitos, pelo que apresenta um risco baixo. Quanto ao seu impacto, este é alto visto que retira a disponibilidade do sistema.

- RISCO: 1
- IMPACTO: 5
- CRITICIDADE: $1 + 5 = 6$

7.3.1.6 Elevation of privilege

Não há preocupações com elevação de privilégios, pois o servidor já possui permissões sobre todas as pastas relevantes. Isso significa que o servidor tem acesso adequado e autorizado para ler, gravar as pastas conforme necessário para suas operações. Portanto, não há risco dele conseguir acesso não autorizado a recursos adicionais através de técnicas de elevação de privilégios, já que o servidor já detém as permissões adequadas desde o início.

8 Pontos Críticos a Melhorar

Apesar da tentativa do grupo da elaboração de um serviço o mais seguro possível, tentando sempre assemelhar a nossa solução com a do *gmail* estamos cientes que o nosso sistema pode relevar falhas de segurança e alguns comprometimentos se implementado em ambiente de produção. Alguns desses problemas, assim como formas de os mitigar já foram abordados ao longo deste relatório porém deixamos aqui uns apontamentos para alguns outros que necessitam de uma revisão se efetivamente quisermos lançar este projeto para produção.

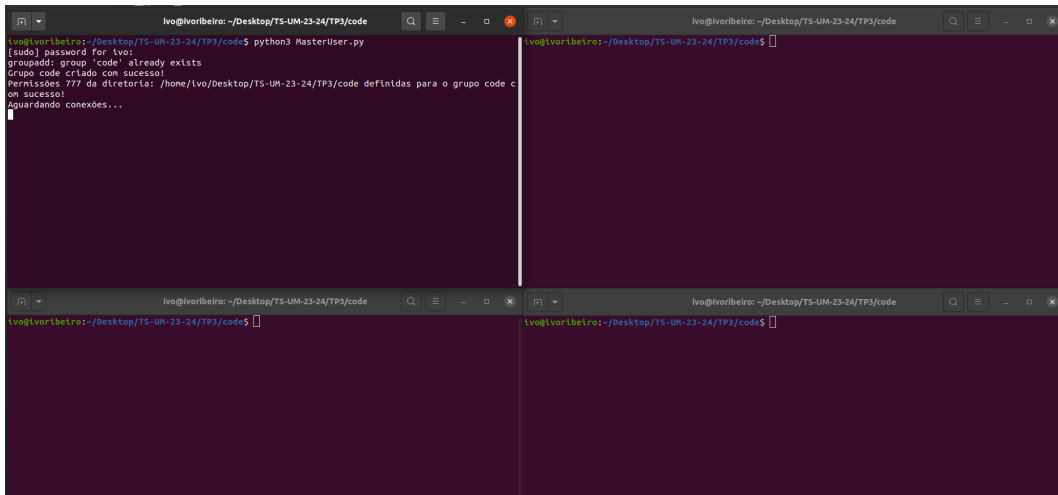
- **Autoridade certificadora:** Implementar um serviço externo responsável por gerir e validar os certificados e chaves em circulação no nosso sistema. Adicionalmente mecanismos de controlo de certificados expirados.
- **Controlo no MasterUser:** Deve-se garantir um melhor controlo sobre as operações e quem as pode pedir. Adicionalmente gerir um ficheiro de *logs* que garanta a autenticidade e não repúdio das ações.
- **Controlo nas diretorias:** O servidor deve ter menos permissões sob as diretorias dos utilizadores, prevenindo assim o sistema contra servidores maliciosos. Idealmente os utilizadores teriam permissão de leitura e o servidor permissão de escrita. Quando à informação adicional como saber que mensagens já foram lidas, deveríamos implementar mecanismos em ambas as partes que o garantam.
- **Mensagens de Grupo não cifradas:** Como não tínhamos uma autoridade certificadora capaz de produzir certificados e chaves novas para cada grupo criado não implementamos esse serviço nas mensagens de grupos nem nas conversações síncronas. Para um bom ambiente de produção essa questão deve ser resolvida e criar e dar acesso aos legítimos intervenientes essas chaves garantindo problemas como repúdio, confiabilidade e autenticidade também para mensagens trocas nestes casos.
- **Acessibilidade aos Grupos:** Já abordado a cima, mas devemos implementar um serviço capaz de tornar os grupos privados e só conceber acesso a utilizadores específicos e não a todos os que pretendam entrar.
- **Gerenciamento de sessões:** Implementar um sistema de gestão de sessões baseada em *tokens* para evitar o sequestro de sessões. Uma forma de fazer isso seria utilizar *tokens* de sessão seguros que são verificados a cada solicitação do utilizador.
- **Implementar um sistema de reconhecimento e monitorização de ataques *DoS*,** essencial para manter a aplicação em pleno funcionamento.

9 Conclusão

O projeto desenvolvido provou ser uma excelente aplicação dos conhecimentos teóricos e práticos adquiridos na Unidade Curricular. A criação de um serviço de mensagens seguro, utilizando um sistema Cliente/Servidor, em ambiente *Linux*, usufruindo das capacidades do mesmo de gestão de utilizadores, grupos e permissões sobre diretorias, que permitiu garantir autenticidade, integridade e confidencialidade nas comunicações entre clientes deste serviço.

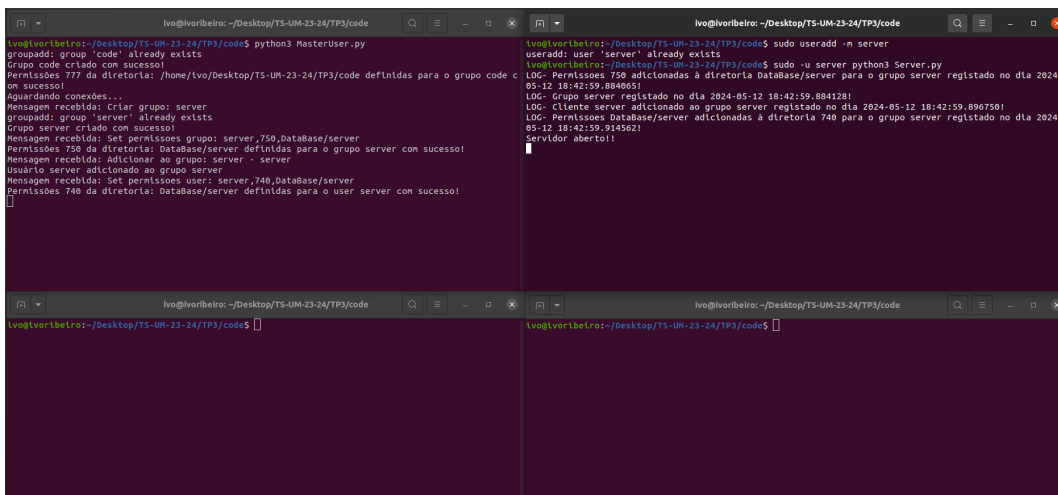
Entretanto, as análises à nossa própria solução, revelaram necessidades de melhorias, como uma melhor definição das permissões sobre certos dados, um melhor monitoramento das atividades que ocorrerem no sistema de modo a garantir o não repúdio e autenticidade das ações e uma melhor garantia de controlo de concorrência no servidor. Concluindo, o projeto não apenas atendeu aos requisitos iniciais, mas também forneceu uma plataforma para explorar profundamente a aplicabilidade das técnicas de tecnologias de segurança.

10 Testes ao serviço



The figure shows four terminal windows arranged in a 2x2 grid. The top-left window shows the execution of `python3 MasterUser.py`, which creates a group named 'code' and sets permissions. The top-right window shows the prompt `lvo@lvoirbelro: ~/Desktop/TS-UM-23-24/TP3/code$`. The bottom-left window shows the prompt `lvo@lvoirbelro: ~/Desktop/TS-UM-23-24/TP3/code$`. The bottom-right window shows the prompt `lvo@lvoirbelro: ~/Desktop/TS-UM-23-24/TP3/code$`.

Figura 1: Inicialização do servidor MasterUser



The figure shows four terminal windows arranged in a 2x2 grid. The top-left window shows the execution of `python3 MasterUser.py`, which creates a group named 'server' and sets permissions. The top-right window shows the execution of `sudo useradd -m server`, which creates a system user named 'server'. The bottom-left window shows the prompt `lvo@lvoirbelro: ~/Desktop/TS-UM-23-24/TP3/code$`. The bottom-right window shows the prompt `lvo@lvoirbelro: ~/Desktop/TS-UM-23-24/TP3/code$`.

Figura 2: Inicialização do servidorNota: criação do utilizador de sistema com o nome 'server'

```

lvo@lvoirbeiro: ~/Desktop/TS-UM-23-24/TP3/code
lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
om sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
useradd: warning: the home directory /home/CLI1 already exists.
useradd: Not copying any file from skel directory into it.
Usuário CLI1 criado com sucesso!
Usuário CLI1 adicionado ao grupo code
groupadd: group 'CLI1' already exists
Grupo CLI1 criado com sucesso!
Usuário CLI1 adicionado ao grupo CLI1
Permissões 750 da diretoria: Database/CLI1 definidas para o grupo CLI1 com sucesso!

lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 18:47:47.742316!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 18:47:47.754683!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-
05-12 18:47:47.771389!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 34886)
LOG- Cliente CLI1 registrado no dia 2024-05-12 18:48:14.258848!
LOG- Grupo CLI1 registrado no dia 2024-05-12 18:48:14.269841!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo CLI1 registrado no dia 2024-05-1
2 18:48:14.287010!
LOG- Permissões 750 adicionadas à diretoria Database/CLI1 para o grupo CLI1 registrado no dia 2024-05-1
2 18:48:14.305468!

lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 ClienteStarter.py
Nome de usuário: CLI1
Password: 123
1- Register!
2- Login!
1
User registrado com sucesso!
[sudo] password for lvo:

lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
om sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
useradd: warning: the home directory /home/CLI2 already exists.
useradd: Not copying any file from skel directory into it.
Usuário CLI2 criado com sucesso!
Usuário CLI2 adicionado ao grupo code
groupadd: group 'CLI2' already exists
Grupo CLI2 criado com sucesso!
Usuário CLI2 adicionado ao grupo CLI2
Permissões 750 da diretoria: Database/CLI2 definidas para o grupo CLI2 com sucesso!

lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 18:52:36.771689!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 18:52:36.793667!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-
05-12 18:52:36.823007!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 34186)
LOG- Cliente CLI2 registrado no dia 2024-05-12 18:52:43.955331!
LOG- Grupo CLI2 registrado no dia 2024-05-12 18:52:43.964742!
LOG- Cliente CLI2 adicionado ao grupo CLI2 registrado no dia 2024-05-12 18:52:43.982753!
LOG- Permissões 750 adicionadas à diretoria Database/CLI2 para o grupo CLI2 registrado no dia 2024-05-1
2 18:52:43.999671!
Conexão estabelecida com ('127.0.0.1', 34188)

lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$
1- Check MailBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!

```

Figura 3: Registo do CLI1 no nosso sistema

Nota: O pedido de password para execução como *superUser* dá-se por o *ClienteStarter* pedir a execução da aplicação cliente como o novo utilizador de sistema (*sudo -u CLI1 python3 Cliente.py CLI1 123*)

```

lvo@lvoirbeiro: ~/Desktop/TS-UM-23-24/TP3/code
lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
om sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
useradd: warning: the home directory /home/CLI2 already exists.
useradd: Not copying any file from skel directory into it.
Usuário CLI2 criado com sucesso!
Usuário CLI2 adicionado ao grupo code
groupadd: group 'CLI2' already exists
Grupo CLI2 criado com sucesso!
Usuário CLI2 adicionado ao grupo CLI2
Permissões 750 da diretoria: Database/CLI2 definidas para o grupo CLI2 com sucesso!

lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 18:52:36.771689!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 18:52:36.793667!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-
05-12 18:52:36.823007!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 34186)
LOG- Cliente CLI2 registrado no dia 2024-05-12 18:52:43.955331!
LOG- Grupo CLI2 registrado no dia 2024-05-12 18:52:43.964742!
LOG- Cliente CLI2 adicionado ao grupo CLI2 registrado no dia 2024-05-12 18:52:43.982753!
LOG- Permissões 750 adicionadas à diretoria Database/CLI2 para o grupo CLI2 registrado no dia 2024-05-1
2 18:52:43.999671!
Conexão estabelecida com ('127.0.0.1', 34188)

lvo@lvoirbeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 Cliente.py CLI1 123
1- Check MailBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!

```

Figura 4: À esquerda o CLI2 aberto e à direita a inicialização do CLI1 ambos como utilizador CLI2 e CLI1 de sistema, respetivamente

The image shows two terminal windows side-by-side, both running on a system with the user 'lvo' and directory '/Desktop/TS-UM-23-24/TP3/code'.

Left Terminal (MasterUser.py):

```
lvo@lvoribeiro: ~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
on sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
useradd: warning: the home directory /home/CLI2 already exists.
useradd: Not copying any file from skel directory into it.
Usuário CLI2 criado com sucesso!
Grupo CLI2 adicionado ao grupo code
groupadd: group 'CLI2' already exists
Grupo CLI2 criado com sucesso!
Usuário CLI2 adicionado ao grupo CLI2
Permissões 750 da diretoria: Database/CLI2 definidas para o grupo CLI2 com sucesso!
[]
```

Right Terminal (Server.py):

```
lvo@lvoribeiro: ~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 18:52:36.711689!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 18:52:36.793867!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-05-12 18:52:36.821007!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 34180)
LOG- Cliente CLI2 registrado no dia 2024-05-12 18:52:43.955535!
LOG- Grupo CLI2 registrado no dia 2024-05-12 18:52:43.964742!
LOG- Cliente CLI2 adicionado ao grupo CLI2 registrado no dia 2024-05-12 18:52:43.982753!
LOG- Permissões 750 adicionadas à diretoria Database/CLI2 para o grupo CLI2 registrado no dia 2024-05-12 18:52:43.999671!
Conexão estabelecida com ('127.0.0.1', 34180)
Conexão estabelecida com ('127.0.0.1', 56222)
LOG- Mensagem recebida do utilizador CLI2 para o utilizador CLI1.
LOG- Atualização da leitura de mensagens do utilizador CLI1.
[]
```

Bottom Left Terminal (Message sent):

```
#####
Destinatário (Receiver): CLI1
Assunto (Subject): Nova conversa
Mensagem (Content): Ola amigo.Tudo bem?
#####
Mensagem enviada(Message sent)
#####
1- Check MailBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!
#####
```

Bottom Right Terminal (Message received):

```
#####
Message number:1
Subject: Nova conversa
Content: Ola amigo.Tudo bem?
#####
1- Check MailBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!
#####
```

Figura 5: Envio e respetiva receção de uma mensagem pelos dois clientes acima logados

The image shows four terminal windows arranged in a 2x2 grid, illustrating the process of establishing a synchronous conversation.

Top Left (MasterUser.py):

```
lvo@lvoribeiro: ~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
on sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
[]
```

Top Right (Server.py):

```
lvo@lvoribeiro: ~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 19:00:01.991582!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 19:00:02.003317!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-05-12 19:00:02.019310!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 40656)
Conexão estabelecida com ('127.0.0.1', 40632)
LOG- User CLI1 em modo livechat!
LOG- User CLI2 em modo livechat!
[]
```

Bottom Left (Live Chat Model):

```
Live Chat Model
ask-(user) to ask for a LiveChat!
!acpt-(user) to accept a LiveChat!
Press ENTER to refresh!
!exit to exit
[]
```

Bottom Right (Live Chat Model):

```
Live Chat Model
ask-(user) to ask for a LiveChat!
!acpt-(user) to accept a LiveChat!
Press ENTER to refresh!
!exit to exit
CLI1 está disponível para livechat!
[]
```

Bottom Left (Message.py):

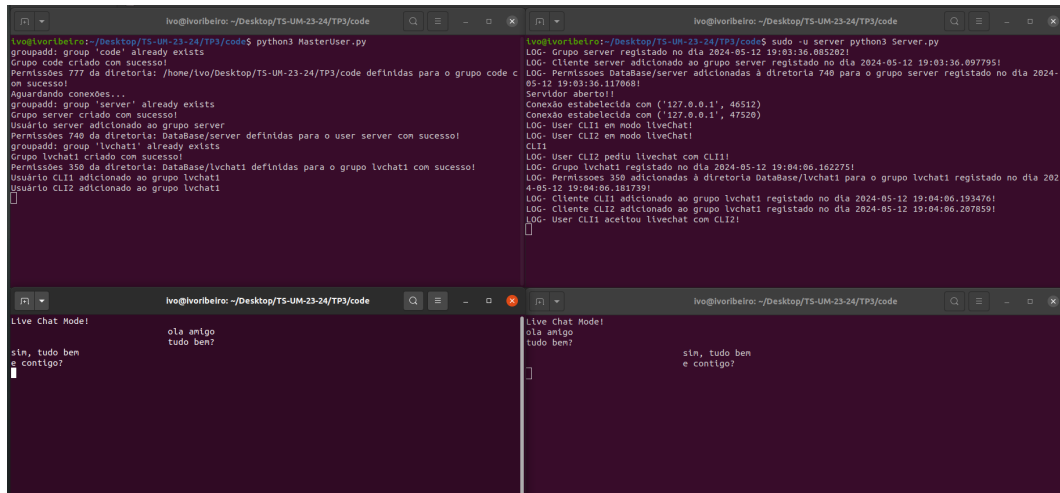
```
lvo@lvoribeiro: ~/Desktop/TS-UM-23-24/TP3/code$ python3 Message.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
on sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
Grupo livechat criado com sucesso!
Permissões 370 da diretoria: Database/livechat definidas para o grupo livechat com sucesso!
Usuário CLI1 adicionado ao grupo livechat
Usuário CLI2 adicionado ao grupo livechat
[]
```

Bottom Right (Message.py):

```
lvo@lvoribeiro: ~/Desktop/TS-UM-23-24/TP3/code$ python3 Message.py
LOG- Grupo server registrado no dia 2024-05-12 19:00:01.991582!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 19:00:02.003317!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-05-12 19:00:02.019310!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 40656)
Conexão estabelecida com ('127.0.0.1', 40632)
LOG- User CLI1 em modo livechat!
LOG- User CLI2 em modo livechat!
CLI1
LOG- User CLI2 pediu livechat com CLI1!
LOG- Grupo livechat registrado no dia 2024-05-12 19:01:04.030002!
LOG- Permissões 370 adicionadas à diretoria Database/livechat para o grupo livechat registrado no dia 2024-05-12 19:01:04.046495!
LOG- Cliente CLI1 adicionado ao grupo livechat registrado no dia 2024-05-12 19:01:04.064886!
LOG- Cliente CLI2 adicionado ao grupo livechat registrado no dia 2024-05-12 19:01:04.082664!
LOG- User CLI1 aceita livechat com CLI2!
[]
```

Figura 6: Processo de estabelecer uma conversação síncrona entre dois clientes

Nota: o primeiro a colocar-se em modo *LiveChat* recebe a informação que ninguém está disponível já o segundo obtém a informação que o outro se encontra *on-line*, faz o pedido e que de seguida é aceite



The screenshot shows two terminal windows side-by-side. The left window displays the output of a Python script that sets up a live chat system, including creating groups, adding users, and defining permissions. The right window shows the server logs, which include timestamps, IP addresses, and messages from the clients. Below the terminal windows, a chat interface is visible with a text input field and a 'Live Chat Mode!' button. The chat history shows a conversation between two users.

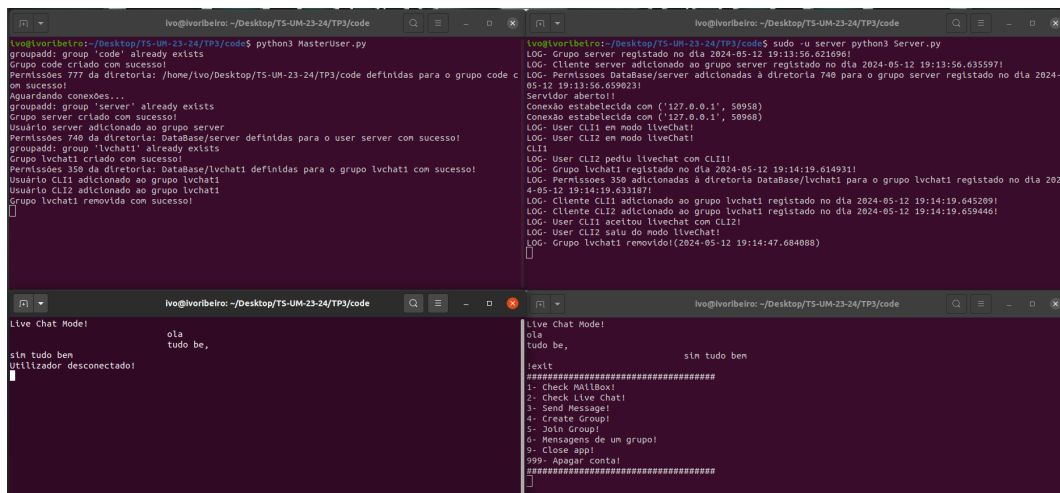
```
lvo@lvoribeiro: ~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
om sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
groupadd: group 'lvchat1' already exists
Grupo lvchat1 criado com sucesso!
Permissões 350 da diretoria: Database/lvchat1 definidas para o grupo lvchat1 com sucesso!
Usuário CLI1 adicionado ao grupo lvchat1
Usuário CLI2 adicionado ao grupo lvchat1
[]

lvo@lvoribeiro:~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 19:03:36.085202!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 19:03:36.097795!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-05-12 19:03:36.117668!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 46512)
Conexão estabelecida com ('127.0.0.1', 47520)
LOG- User CLI1 em modo livechat!
LOG- User CLI2 em modo livechat!
CLI1
LOG- User CLI2 pediu livechat com CLI1!
LOG- Grupo lvchat1 registrado no dia 2024-05-12 19:04:06.162275!
LOG- Permissões 350 adicionadas à diretoria Database/lvchat1 para o grupo lvchat1 registrado no dia 2024-05-12 19:04:06.181791!
LOG- Cliente CLI1 adicionado ao grupo lvchat1 registrado no dia 2024-05-12 19:04:06.193476!
LOG- Cliente CLI2 adicionado ao grupo lvchat1 registrado no dia 2024-05-12 19:04:06.207859!
LOG- User CLI1 aceitou livechat com CLI2!
[]

Live Chat Mode!
ola amigo
tudo ben?

sin, tudo ben
e contigo?
[]
```

Figura 7: Conversação síncrona entre dois clientes



The screenshot shows the same two terminal windows as Figure 7, but now the chat session is being terminated. The left window shows the output of the Python script, which includes a message indicating that the group 'lvchat1' has been removed. The right window shows the server logs, which include a message indicating that the group 'lvchat1' has been removed. Below the terminal windows, the chat interface shows a message from the user 'sin' saying 'tudo be,' and a message from the user 'ola' saying 'tudo be, sin tudo ben Utilizador desconectado!'. The chat history also shows a list of menu items.

```
lvo@lvoribeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/lvo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
om sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
groupadd: group 'lvchat1' already exists
Grupo lvchat1 criado com sucesso!
Permissões 350 da diretoria: Database/lvchat1 definidas para o grupo lvchat1 com sucesso!
Usuário CLI1 adicionado ao grupo lvchat1
Usuário CLI2 adicionado ao grupo lvchat1
Grupo lvchat1 removido com sucesso!
[]

lvo@lvoribeiro:~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 19:13:56.621696!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 19:13:56.635597!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-05-12 19:13:56.659023!
Servidor aberto!!
Conexão estabelecida com ('127.0.0.1', 50958)
Conexão estabelecida com ('127.0.0.1', 50968)
LOG- User CLI1 em modo livechat!
LOG- User CLI2 em modo livechat!
CLI1
LOG- User CLI2 pediu livechat com CLI1!
LOG- Grupo lvchat1 registrado no dia 2024-05-12 19:14:19.614931!
LOG- Permissões 350 adicionadas à diretoria Database/lvchat1 para o grupo lvchat1 registrado no dia 2024-05-12 19:14:19.633187!
LOG- Cliente CLI1 adicionado ao grupo lvchat1 registrado no dia 2024-05-12 19:14:19.645209!
LOG- Cliente CLI2 adicionado ao grupo lvchat1 registrado no dia 2024-05-12 19:14:19.659446!
LOG- User CLI1 aceitou livechat com CLI2!
LOG- User CLI2 saiu do modo livechat!
LOG- grupo lvchat1 removido(2024-05-12 19:14:47.684088)
[]

Live Chat Mode!
ola
tudo be,

sin tudo ben
Utilizador desconectado!
[]

Live Chat Mode!
ola
tudo be,

sin tudo ben

exit
#####
1- Check WallBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!
#####
[]
```

Figura 8: Termina de uma sessão de conversação síncrona

```
ivo@ivoribeiro: ~/Desktop/TS-UM-23-24/TP3/code
ivo@ivoribeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/Ivo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
om sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
[]

ivo@ivoribeiro:~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 19:20:15.334794!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 19:20:15.346673!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-
05-12 19:20:15.364380!
Servidor aberto!
Conexão estabelecida com ('127.0.0.1', 54376)
conexão estabelecida com ('127.0.0.1', 52008)
[]

ivo@ivoribeiro:~/Desktop/TS-UM-23-24/TP3/code
Name of the group: TS
#####
1- Check MailBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!
#####
[]

ivo@ivoribeiro:~/Desktop/TS-UM-23-24/TP3/code$ python3 MasterUser.py
groupadd: group 'code' already exists
Grupo code criado com sucesso!
Permissões 777 da diretoria: /home/Ivo/Desktop/TS-UM-23-24/TP3/code definidas para o grupo code c
om sucesso!
Aguardando conexões...
groupadd: group 'server' already exists
Grupo server criado com sucesso!
Usuário server adicionado ao grupo server
Permissões 740 da diretoria: Database/server definidas para o user server com sucesso!
Grupo TS criado com sucesso!
Permissões 750 da diretoria: Database/CLI1 definidas para o grupo CLI1 com sucesso!
Usuário CLI1 adicionado ao grupo TS
Usuário CLI2 adicionado ao grupo TS
[]

ivo@ivoribeiro:~/Desktop/TS-UM-23-24/TP3/code$ sudo -u server python3 Server.py
LOG- Grupo server registrado no dia 2024-05-12 19:20:15.334794!
LOG- Cliente server adicionado ao grupo server registrado no dia 2024-05-12 19:20:15.346673!
LOG- Permissões Database/server adicionadas à diretoria 740 para o grupo server registrado no dia 2024-
05-12 19:20:15.364380!
Servidor aberto!
Conexão estabelecida com ('127.0.0.1', 54376)
Conexão estabelecida com ('127.0.0.1', 52008)
LOG- Grupo TS registrado no dia 2024-05-12 19:20:55.143436!
LOG- Permissões 750 adicionadas à diretoria Database/CLI1 para o grupo CLI1 registrado no dia 2024-05-1
2 19:20:55.163152!
LOG- Cliente CLI1 adicionado ao grupo TS registrado no dia 2024-05-12 19:20:55.184350!
LOG- Cliente CLI2 adicionado ao grupo TS registrado no dia 2024-05-12 19:21:06.742349!
[]

ivo@ivoribeiro:~/Desktop/TS-UM-23-24/TP3/code
Name of the group: TS
#####
1- Check MailBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!
#####
[]

ivo@ivoribeiro:~/Desktop/TS-UM-23-24/TP3/code
Name of the group: TS
#####
1- Check MailBox!
2- Check Live Chat!
3- Send Message!
4- Create Group!
5- Join Group!
6- Mensagens de um grupo!
9- Close app!
999- Apagar conta!
#####
[]
```

Figura 9: Criação de grupo e pedido de inserção em um grupo

Nota: neste exemplo mostramos duas ações possíveis sendo um cliente a criar o grupo e outro a pedir para entrar nesse grupo recém criado



Figura 10: Envio e respetiva receção de uma mensagem de grupo pelos dois clientes, ambos tem acesso à mesma mensagem pois pertencem ao mesmo grupo

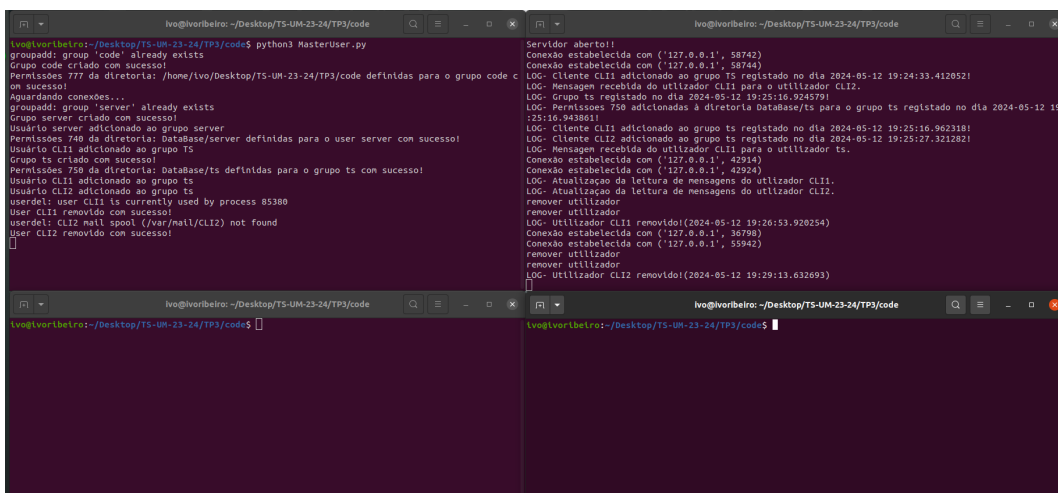


Figura 11: Opção de sair do serviço onde ambos os clientes assim como as informações a eles destinadas durante o funcionamento do serviço foram eliminados, bem como os próprios utilizadores de sistema e ainda os grupos com o nome do próprio utilizador.