



PROJETO DE DESENVOLVIMENTO 1 *Engenharia de Segurança*

TECHNICAL REPORT

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

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

 **Paulo Henrique Pianissola de Cerqueira**
Mestrado Integrado em Engenharia Informática
Universidade do Minho
PG52699
PG52699@alunos.uminho.pt

28 de abril de 2024

ABSTRACT

Este projeto tem como objetivo a conceção e análise de requisitos para o desenvolvimento de um serviço de mensagens seguro, proporcionando aos membros de uma organização uma forma confiável de trocar mensagens com garantias de autenticidade, integridade e fiabilidade do sistema. Sendo um projeto desenvolvido no âmbito da cadeira de Engenharia de Segurança, iremos ao máximo focar a nossa implementação nas métricas até aqui discutidas e abordadas nas aulas teóricas e teórico-práticas, como algoritmos de encriptação e descriptação de mensagens, técnicas de comunicação seguras como *TLS* e boas praticas também mencionadas.

Keywords análise de requisitos · serviço de mensagens seguro · 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 clientes, 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, que permitirá enviar mensagens a outros clientes, ver a lista de mensagens para ele enviadas, bem como ver o conteúdo dessas mesmas mensagens. Já o **servidor** será responsável por gerir as solicitações dos utilizadores e manter o estado da aplicação. Estas duas entidades irão comunicar sob um protocolo de mensagens seguras e cifradas que iremos implementar para garantir a conformidade das mensagens trocadas.

2 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 e receber mensagens de forma segura. O **servidor** será responsável por responder às solicitações dos utilizadores e manter o estado do sistema.

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.

3 Requisitos de Segurança Pretendidos

Assim como pedido no enunciado, todas as comunicações entre a aplicação cliente e servidor devem ser protegidas contra acesso ilegítimo e/ou manipulação (incluindo de outros utilizadores do sistema). Como solução a este requisito, implementamos um serviço onde todas as mensagens são cifradas com a chave pública do recetor da mesma, isto é, o cliente cifra a mensagem com a chave pública do servidor e envia-a para o mesmo, garantindo assim que apenas o legítimo recetor e portador da chave privada as consegue decifrar.

Quanto ao requisito de confiar no servidor para efeitos da atribuição da "hora/data" associada às mensagens, garantimos que o mesmo, perante a receção de uma mensagem, escreve num ficheiro de *logs* a informação de que recebeu uma mensagem de determinado *user* com um outro *user* como destinatário, bem como o assunto da mesma e ainda uma atribuição da "hora/data" dessa receção.

Como solução relativa ao requisito de o servidor não comprometer a confidencialidade das mensagens tratadas, implementamos um mecanismo de cifragem do conteúdo da mesma com a chave pública do cliente destinatário, garantindo que apenas o proprietário da chave privada associada a essa chave pública usada seja capaz de decifrar o conteúdo da mensagem. Assim, garantimos que o servidor apenas tem acesso à informação não confidencial das mensagens e que nunca tem acesso ao conteúdo nelas presente. O servidor guarda a informação acima descrita relativa aos metadados das mensagens num ficheiro de *logs* mas armazena a totalidade da cifra recebida num ficheiro numa diretoria denominada com o nome do destinatário, o que garante que qualquer manipulação ao conteúdo da mesma seja identificada.

Os clientes, ao receberem uma mensagem, devem poder verificar que a mesma foi efetivamente enviada pelo remetente e a si dirigida. Este requisito foi solucionado não só para as mensagens de um cliente para o outro, mas também para todas as comunicações do nosso sistema. Em caso de erro, também cumprimos com o requisito de enviar para o *stderr* a indicação respetiva, nomeadamente: *'MSG SERVICE: unknown message'* no caso da mensagem não existir na lista do utilizador; *'MSG SERVICE: verification error'* no caso da verificação falhar.

4 Descrição das entidades

4.1 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 e receber mensagens, obter chaves públicas de outros utilizadores e solicitar informações sobre as mensagens. Dados de identificação do servidor, como chave pública do mesmo, são dados como adquiridos na utilização da aplicação cliente.

Podemos destacar as seguintes funcionalidades principais:

- **Registo do Utilizador:** Ao iniciar, o cliente regista o seu certificado público e identificação no servidor central. Isso permite que outros clientes o identifiquem e enviem mensagens de forma segura, onde só o próprio destinatário consegue decifrar.
- **Envio de Mensagens:** O cliente pode enviar mensagens encriptadas para outros utilizadores do sistema. Antes do envio, verifica-se o tamanho da mensagem para garantir que não excede 1000 bytes. A mensagem é cifrada usando a chave pública do destinatário e assinada com a chave privada do remetente.
- **Obtenção de Chaves Públicas:** O cliente pode solicitar a chave pública de outros usuários ao servidor central, caso não a possua. Isso é feito de forma segura, garantindo a autenticidade da chave recebida, permitindo que possa depois ser encriptado o conteúdo secreto, apenas possível de descriptar pelo legítimo recetor.
- **Consulta de Mensagens:** O cliente pode solicitar ao servidor uma lista das mensagens pendentes, tanto as não lidas, como as já lidas.
- **Receber Mensagens:** O cliente pode solicitar a visualização de mensagens a ele destinadas presentes no servidor. As mensagens são descriptadas com a chave privada do destinatário e verificadas usando a chave pública do remetente.

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.

4.2 Aplicação Servidor

A aplicação servidor é responsável por gerir as conexões dos clientes, receber e enviar mensagens de forma segura, armazenar as mensagens recebidas e gerir/armazenar as chaves públicas dos utilizadores. Esta opera em um modelo centralizado, onde os clientes se conectam para enviar e receber mensagens.

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:

- **Registar Novos Utilizadores:** O servidor permite o registo de novas entidades no sistema, atribuindo-lhes um identificador único (*UID*) e armazenando suas chaves públicas para uso futuro.
- **Receber e Armazenar Mensagens:** O servidor recebe as mensagens enviadas pelos clientes, armazena-as em diretorias específicas para cada usuário e regista os metadados das mesmas em um ficheiro de *log*, que mapeia a informação armazenada e é usado em alternativa a dados em memória para prevenir perdas de informação aquando uma negação do serviço do servidor.
- **Consulta de Mensagens Pendentes:** Os clientes podem solicitar ao servidor uma lista das mensagens pendentes, tanto as não lidas quanto as já lidas.
- **Solicitação de Chaves Públicas:** Os clientes podem solicitar ao servidor as chaves públicas de outros utilizadores, caso não as tenham. O servidor responde com as chaves públicas solicitadas de forma segura.

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 no formato em que foram recebidas (no formato de cifras), protegendo-as contra acesso e manipulação não autorizado.

Relativamente ao armazenamento de dados, como referido acima, as mensagens são armazenadas em diretórios específicos para cada utilizador, o que permite uma mais fácil gestão e a recuperação das mensagens. 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.

4.3 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.

4.4 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.

5 Infraestrutura e Comunicações

De modo a descrever melhor o processo de comunicação do nosso sistema apresentamos dois fluxogramas que demonstram o completo ciclo de comunicação entre as nossas entidades.

No primeiro fluxograma, usamos como exemplo um pedido de observação da lista de mensagens de um cliente. Este exemplo consegue demonstrar o comportamento do nosso sistema em comunicações a solo, como uma mensagem de um cliente onde o destino é o servidor e uma resposta do servidor onde o destinatário é o recetor da mensagem. A ação de pedido de lista de mensagens que identificamos é útil para perceber todos os processos presentes no sistema e comuns a todos os tipos de mensagens trocadas.

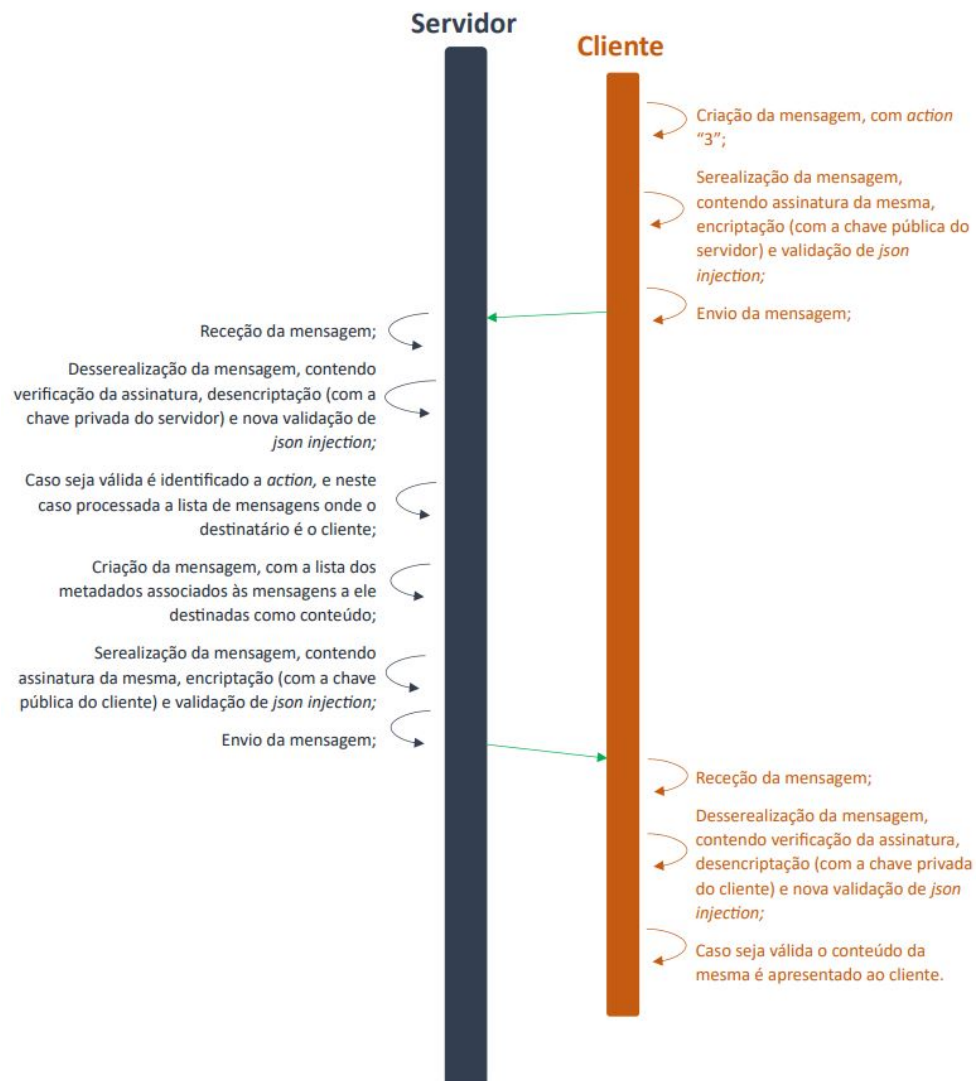


Figura 1: Fluxograma de um pedido de lista de mensagens por um cliente.

No segundo fluxograma, apesar da semelhança com o primeiro, pois como referimos os métodos são comuns a todo o tipo de mensagens, decidimos destacar e realçar a importância que demos à dupla encriptação do conteúdo. Este processo tem como objetivo impedir que o servidor tenha conhecimento sobre o conteúdo trocado entre os clientes, permitindo apenas que acesse aos metadados das mesmas quando por ele recebidas. Isto garante uma confidencialidade extra e uma autenticidade do conteúdo, pois impede que qualquer outro utilizador que possa ter acesso a uma mensagem não a ele destinada consiga obter o conteúdo, uma vez que este apenas é decodificado com a chave privada do legítimo destinatário.

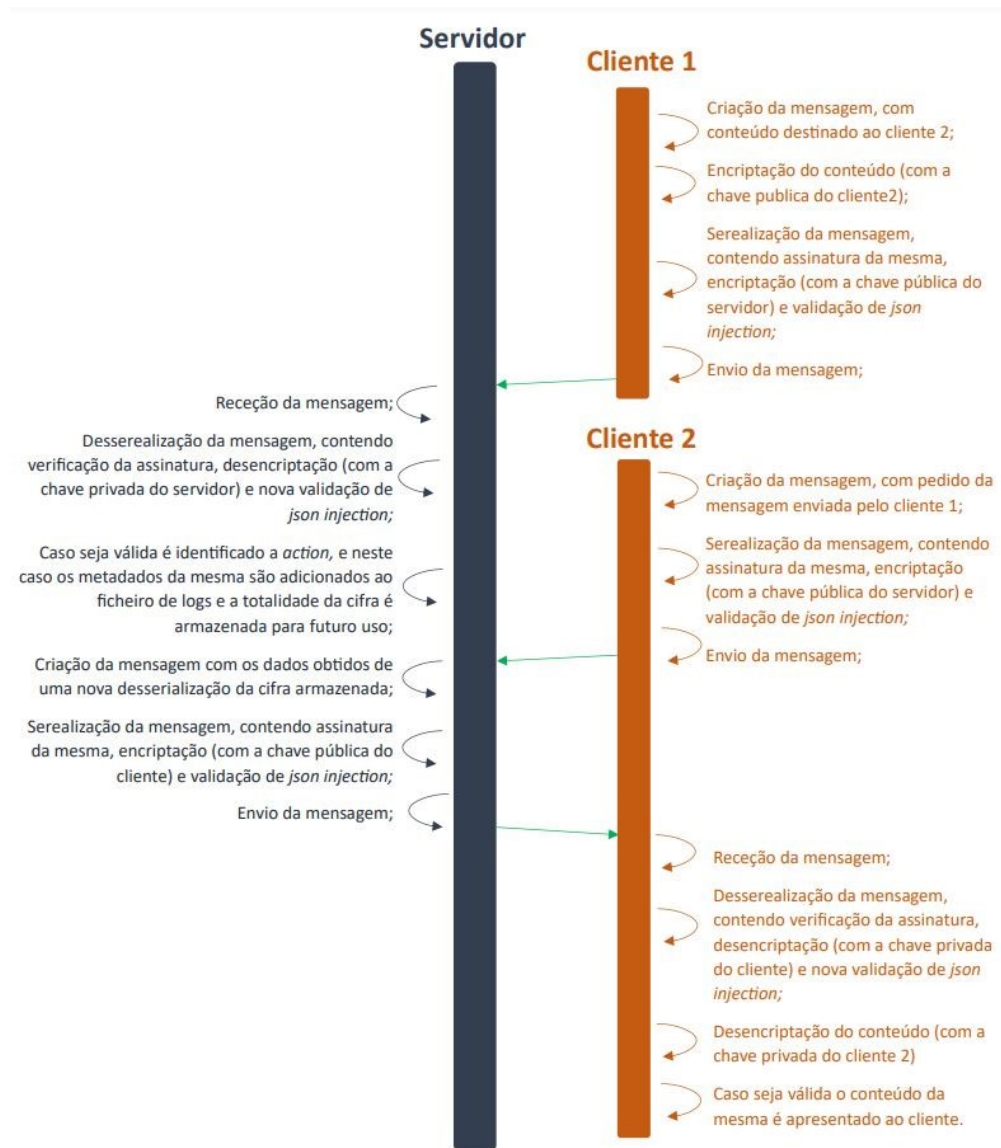


Figura 2: Fluxograma do percurso de uma mensagem cliente_para_cliente.

Atendendo aos requisitos do enunciado e à metodologia por nós definida cumprimos com:

Comunicação Segura: Toda a comunicação entre a aplicação cliente e o servidor será protegida contra acesso ilegítimo e manipulação, garantindo a confidencialidade e integridade das mensagens.

Confiança no Servidor: Confia-se no servidor para atribuir a hora/data de recebimento das mensagens, sem comprometer sua confidencialidade. No entanto, o servidor não deve ser capaz de manipular o conteúdo ou o destinatário das mensagens.

Verificação de Mensagens: Os clientes devem ser capazes de verificar a autenticidade das mensagens recebidas, garantindo que foram enviadas pelo remetente especificado e destinadas ao utilizador correto. Em caso de falha na verificação, serão emitidas mensagens de erro apropriadas.

Encriptação da mensagem: As mensagens são duplamente cifradas e assinadas, primeiramente, é cifrado o conteúdo da mensagem com a chave pública do destinatário. Esta cifragem é feita através de uma combinação de *RSA* e *AES*, método que é chamado de "*cifragem híbrida*".

6 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.

6.1 Aplicação Cliente

6.1.1 Modelação de Ameaças Orientado ao Software

6.1.1.1 Spoofing

A autenticação do cliente baseia-se num certificado emitido por uma entidade confiável e chaves privadas e públicas, porém, não há validação explícita da identidade do servidor, visto que o código foi configurado desta maneira para o cliente não precisar de instalar o certificado do servidor no seu sistema, sendo assim, abre-se espaço para ataques *spoofing*.

Ameaça: Sem a identificação da identidade do servidor, é possível que um atacante se tente passar pelo servidor da aplicação.

Forma de Mitigação: Instalar em todos os clientes o certificado de identidade do servidor, garantindo assim que a aplicação do cliente possa autenticar o servidor.

Análise de Risco: Com todas estas medidas de segurança, como uso do *TLS*, a encriptação do conteúdo com a chave pública do destinatário, nova encriptação e assinatura deste conteúdo com a chave pública do servidor e, por fim, a encriptação da mensagem com a chave pública do servidor, o risco torna-se baixo para fuga de informações e médio para impacto geral no sistema.

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

6.1.1.2 Tampering

Devido a cifragem dupla da mensagem, utilizando tecnologias híbridas com AES e RSA e também as assinaturas das mensagens, não há possibilidade de adulterar a mensagem, mesmo com ataques *man in the middle*, sem que o sistema identifique a adulteração.

6.1.1.3 Repudiation

Com recurso a assinatura digital, a protocolos como *GCM* do *AES* e a *logs* no servidor com identificação dos emissores das mensagens e alocação de um *timestamp*, o não repúdio não é viável nesta aplicação.

6.1.1.4 Information Disclosure

O cliente utiliza tecnologias de SSL e TLS para comunicação com servidor, portanto, não há possibilidade de vazamento de informações, devido a estas tecnologias cifrarem por elas próprias o conteúdo que passa por este canal, para além da nossa própria cifragem.

6.1.1.5 DoS

Apesar de haver um limite máximo do tamanho da mensagem que pode ser enviada, não há limites para a quantidade de mensagens que podem ser enviadas simultaneamente.

Ameaças: Um adversário pode inundar o sistema com uma quantidade excessiva de mensagens, sobrecarregando o sistema e impedindo-o de atender os demais utilizadores. Além disso, a chegada dessas solicitações em grande quantidade pode aumentar significativamente o consumo de recursos de rede.

Forma de Mitigação: Limitar a quantidade de mensagens por um determinado período de tempo.

Análise de Risco: Esta fraqueza pode ser considerada de alto impacto e risco médio.

- RISCO: 3
- IMPACTO: 5
- CRITICIDADE: $3 + 5 = 8$

6.1.1.6 Elevation of Privilege

No contexto da aplicação do cliente, todos têm o tratamento igualitário perante a aplicação, não havendo um utilizador administrador, portanto, não há a possibilidade da elevação de privilégios.

6.2 Aplicação Servidor

6.2.1 Modelação de Ameaças Orientado ao Software

6.2.1.1 Spoofing

Não há preocupações com *spoofing* neste contexto. A autoridade certificadora é uma entidade confiável e bem estabelecida, externa ao sistema em desenvolvimento. Ela segue métodos robustos, como o *Certificate Signing Request (CSR)*, para emitir certificados. Portanto, não há risco de falsificação de identidade ou ataques de *spoofing* na emissão ou associação de certificados.

6.2.1.2 Tampering

Fraqueza: Ficheiros 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, pelo que apresenta um risco baixo. Quanto ao seu impacto, este é alto, visto que os ficheiros que a modificação de ficheiros de chave pública e de logs tem grande impacto no sistema.

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

6.2.1.3 Repudiation

Fraqueza: A falta de registos de auditoria torna possível o repúdio, dificultando a responsabilização e rastreamento de ações maliciosas.

Ameaças: Sem o armazenamento de *logs* num sistema de registo de eventos, torna-se impossível auditar as operações que ocorreram entre as entidades. Como resultado, em caso de ataque, não será possível identificar o autor, pois não haverá registos das atividades realizadas no sistema.

Forma de Mitigação: Para mitigar essa vulnerabilidade, é essencial implementar um sistema de registo e auditoria robusto que registre todas as interações com o servidor. Esses registos devem ser armazenados de forma segura e acessível para verificação posterior, o que fornecerá evidências sólidas em caso de disputas de repúdio, ajudando a garantir a confiabilidade e a transparência do sistema.

Análise de Risco: Não está previsto no enunciado a mitigação desse problema, pois não é expressa nos requisitos a necessidade de utilizar uma autenticação robusta. Desta forma, o risco associado a esta fraqueza é médio-alto, já o

impacto desta fraqueza é baixo, visto que os problemas de não repúdio não trazem problemas significativos ao sistema. Desta forma, temos:

- RISCO: 4
- IMPACTO: 1
- CRITICIDADE: $4 + 1 = 5$

6.2.1.4 Information disclosure

Fraqueza: Ficheiros não cifrados.

Ameaças: A principal ameaça relacionada com 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, pelo que apresenta um risco baixo. Quanto ao seu impacto, este é baixo, visto que os ficheiros que têm informações confidenciais estão cifrados, só os de chave pública e os de *logs* é que não.

- RISCO: 1
- IMPACTO: 1
- CRITICIDADE: $1 + 1 = 2$

6.2.1.5 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 não está prevista nos requisitos, pelo que apresenta um risco médio-alto. Quanto ao seu impacto, este é alto pelo facto de impossibilitar o funcionamento do sistema.

- RISCO: 4
- IMPACTO: 5
- CRITICIDADE: $4 + 5 = 9$

6.2.1.6 Elevation of privilege

A presença de mecanismos adequados de controlo de acesso e autenticação no servidor oferece uma camada eficaz de proteção contra problemas de *elevation of privilege*. Esses mecanismos garantem que apenas utilizadores autorizados tenham acesso aos recursos e funcionalidades do servidor, reduzindo assim significativamente o risco de elevação de privilégios não autorizado. Ao implementar estes mecanismos, a aplicação é capaz de validar a identidade dos utilizadores e garantir que tenham apenas os privilégios necessários para realizar suas tarefas.

7 Análise Estática de Código

No seguimento do Projeto de Análise feito anteriormente, decidimos avaliar o nosso próprio sistema segundo a aplicação que descrevemos nesse mesmo projeto. Sendo o *SonarQube* uma ferramenta de análise estática de código e ter como objetivo encontrar vulnerabilidades e os demais problemas no código proposto para revisão, mostra-se útil usufruir desta ferramenta para o nosso caso prático. No caso, é possível encontrar falhas de segurança, bugs e código duplicado automaticamente e com alto grau de confiança.

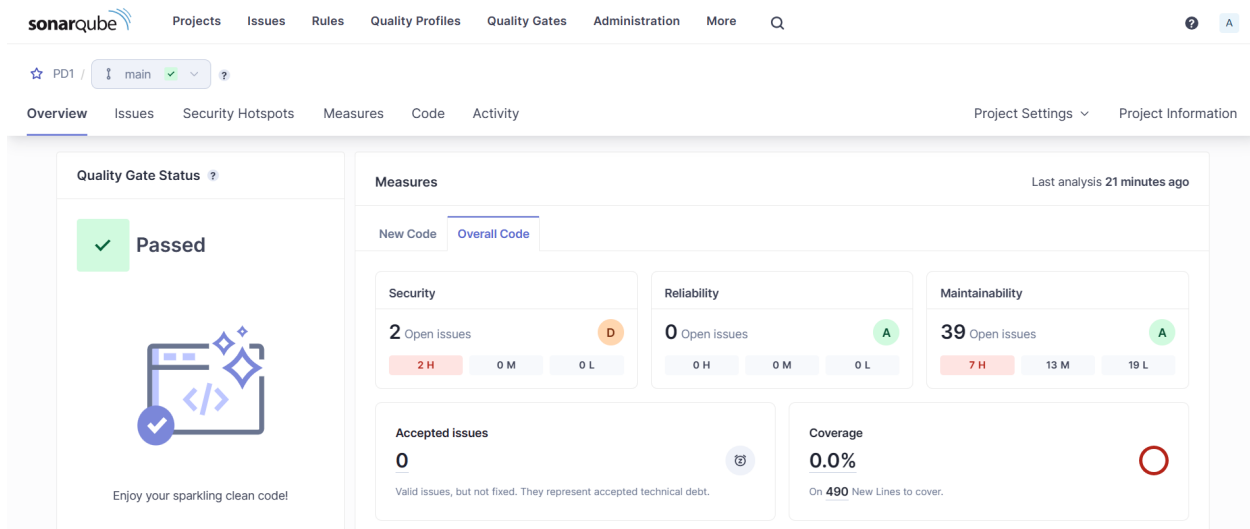


Figura 3: Resumo da análise do SonarQube.

Após a análise do *SonarQube*, os resultados foram bastante positivos em relação à aplicação desenvolvida pelo grupo, conforme a imagem acima. Foram encontrados 39 problemas de manutenibilidade, onde destes somente 7 são considerados de alta criticidade, o que faz com que o código receba a nota **A** neste quesito. Outro ponto que faz parte do conjunto de análise do *SonarQube*, é a confiabilidade da aplicação, onde não foram identificados problemas e, portanto, também recebeu a nota **A**.

Por fim, é analisado a questão de segurança da aplicação, onde, conforme a imagem abaixo, foram encontrados os dois problemas já mencionados pelo grupo neste documento. Ambos são relativos ao código *Cliente.py*, onde é pedido para a aplicação não verificar a validação do certificado e *hostname* do servidor. Isto acontece devido à necessidade de instalação do certificado do servidor nas máquinas dos clientes para que estas reconheçam a confiabilidade do servidor.

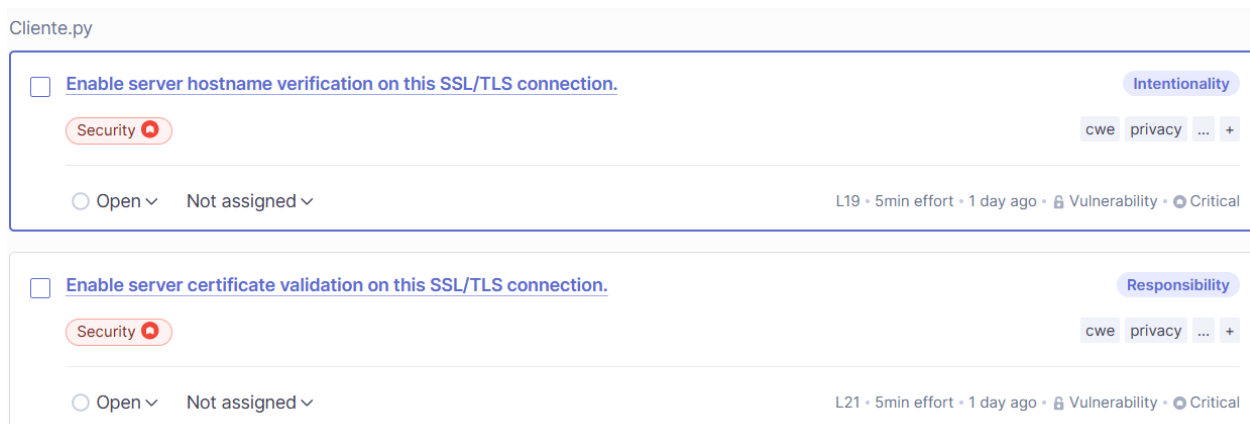


Figura 4: Análise de Segurança do SonarQube.

Em conclusão à análise do *SonarQube*, foram analisadas 655 linhas de código e encontradas somente 2 vulnerabilidades, já conhecidas pelo grupo, e 39 problemas de manutenibilidade, o que garante que a aplicação seja aprovada pelo *Quality Gate* do *SonarQube*, que diz que a aplicação está pronta para entrar no ambiente de produção.

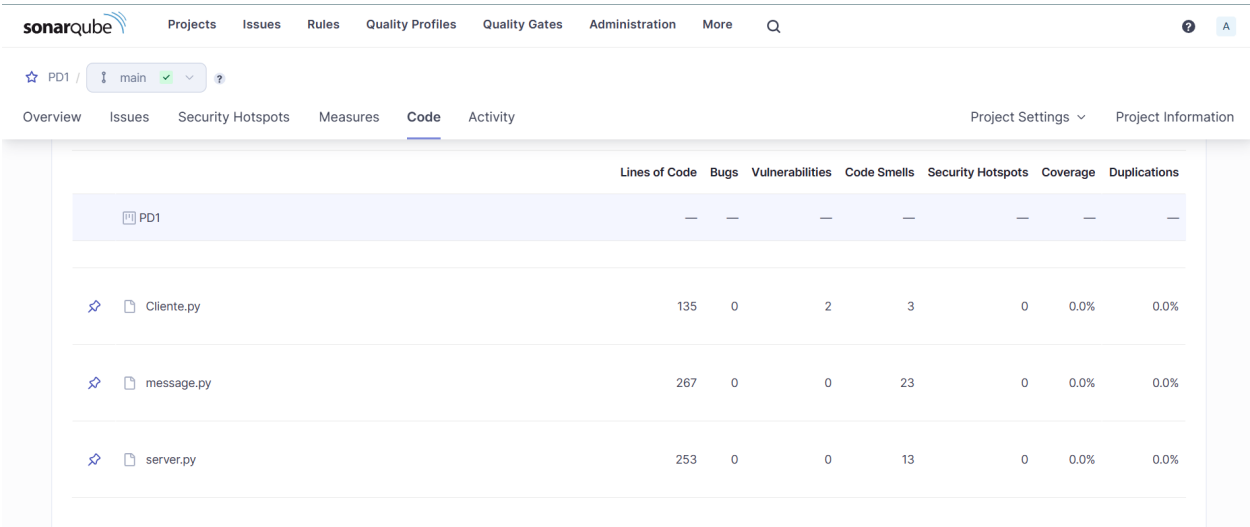


Figura 5: Resumo da análise.

8 Pontos Críticos a Melhorar

A aplicação construída pelo grupo possui um robusto sistema de segurança e atende ao solicitado, porém há pontos que podem ser melhorados, como os citados abaixo:

- 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.
- Banco de dados e *Caching*: Neste projeto a tabela de mensagens enviadas e recebidas pelo utilizador fica num arquivo ".csv", o que pode tornar-se um gargalo caso o envio de mensagens cresça exponencialmente. Para um projeto futuro o ideal seria utilizar um banco de dados baseado em SQL para salvar esta tabela de mensagens e um sistema de *cache* para recuperar os dados frequentemente utilizados.
- Numa versão futura e para a escalabilidade da aplicação seria essencial a presença de uma versão com interface gráfica para o utilizador, que seja responsiva e intuitiva para os utilizadores que não estejam familiarizados com terminais ou linhas de comando.
- Implementar um sistema de reconhecimento e monitorização de ataques *DoS*, essencial para manter a aplicação em pleno funcionamento.
- Integrar o servidor com sistemas de gestão de identidade existentes para uma autenticação mais robusta e centralizada, como por exemplo uma entidade certificadora capaz de regular a emissão de certificados bem como a identificação de certificados.
- Implementar um sistema de armazenamento seguro de chaves privadas, protegendo-as contra acesso não autorizado, bem como um sistema mais seguro de armazenamento das chaves públicas presentes tanto no servidor como nos clientes. Conclusão

9 Conclusão

O projeto desenvolvido provou ser uma excelente aplicação dos conhecimentos teóricos e práticos adquiridos no curso. A criação de um serviço de mensagens seguro, utilizando um sistema Cliente/Servidor, permitiu garantir autenticidade, integridade e confidencialidade nas comunicações entre membros de uma organização. A implementação destacou-se pela utilização dos algoritmos de criptografia *RSA* e *AES* em configuração híbrida, além da incorporação das práticas recomendadas de segurança, incluindo a verificação de assinaturas e a validação contra injeções de *JSON*.

Entretanto, as análises, incluindo a estática com *SonarQube*, revelaram necessidades de melhorias, como um gerenciamento de sessões mais robusto e a criação de uma interface gráfica mais amigável. Concluindo, o projeto não apenas atendeu aos requisitos iniciais, mas também forneceu uma plataforma para explorar profundamente a aplicabilidade das técnicas de engenharia de segurança.