universidade de aveiro

Departamento de Eletrônica, Telecomunicações e Informática

Segurança
1º Semestre 2016/2017

Secure Instant Messaging System

Relatório - Primeiro sub-projeto

Grupo: P3G6

Nuno Silva [72708]
Ricardo Filipe [72727]

Docente: André Zúquete

Conteúdos

1. Introdução e Objetivos

O trabalho proposto para o projeto da unidade curricular de segurança foi um sistema de troca de mensagens instantâneas seguro. A implementação deste sistema consiste num server central que interliga os vários clients que trocam mensagens instantâneas entre si.

Este relatório apresenta todas as fases necessárias ao desenvolvimento deste sistema, mostrando passo-a-passo as diferentes etapas da implementação e os diferentes elementos que compõem o servidor e os clientes.

Os objetivos deste trabalho são acreditar ao sistema um conjunto de componentes que o tornam seguro. A garantia de segurança é dada através de técnicas de criptografia e troca de chaves que impedem que as mensagens entre clientes sejam “escutadas”, introduzidas por terceiros, modificadas ou que sejam sujeitas a qualquer tipo de ataque.

2. Componentes do Sistema

O sistema de troca de mensagens instantâneas pode ser visto como uma rede que tem uma topologia centralizada num **servidor** que interliga os **clientes** que queiram estabelecer comunicações entre si.

2.1. Servidor

O servidor no sistema é visto como o ponto fulcral onde os clientes de se conectam e todas as mensagens de clientes são processadas ou encaminhadas para um cliente.

O servidor contém uma lista dos clientes conectados a si, assim como a respectivo conteúdo criptográfico necessário para interagir com os peers.

2.2. Cliente

O cliente são nós na rede que primeiramente se conectam ao servidor antes de estabelecer qualquer outro tipo de ligação.

Após estarem ligados ao servidor os clientes podem enviar pedidos como obter uma lista de peers ligados ao servidor ou iniciar uma comunicação com outro cliente.

3. Tipos de mensagens

Todas as mensagens entre peers e servidor e respostas do servidor aos peers são através de mensagens JSON.

3.1. Connect

"Connect" é a primeira mensagem que o cliente envia automaticamente ao servidor para estabelecer a sua ligação.

Consiste numa mensagens onde o cliente envia o seu **id**, **nome**, e **tipo de cifragem** a ser utilizado (algoritmo de cifragem usado entre o servidor e o cliente).

Sem uma mensagem "connect" inicial é impossível enviar qualquer outro tipo de mensagem seja ela qual seja.

3.2. Secure

"Secure" é usado para encapsular mensagens (além do "connect") de maneira segura.

A mensagem encapsulada vai encriptada no campo **payload** da mensagem "secure".

3.3. List

"List" é um tipo de mensagem encapsulado que o cliente usa para obter uma lista de peers conectados ao server. É útil para obter o **id** dos clientes para proceder à sua conexão.

3.4. Client-Connect

"Client-Connect" é um tipo de mensagem encapsulado para estabelecer contacto entre dois clientes. Os campos desta mensagem incluem uma **source** (src), e um **destination** (dst). Incluem também o algoritmo de **cifragem** usado entre os clientes.

3.5. Client-Com

"Client-Com" é um tipo de mensagem encapsulado para um cliente enviar a mensagem propriamente dita para outro cliente. Os campos desta mensagem incluem uma **source** (src), e um **destination** (dst). Incluem também um campo **data** onde vai o conteúdo da mensagem.

3.6. Client-Disconnect

"Client-Disconnect" é um tipo de mensagem encapsulado usado para terminar a ligação entre dois clientes. Os campos desta mensagem incluem uma **source** (src), e um **destination** (dst).

4. Implementação

Durante o desenvolvimento do sistema foram usadas libraries como cryptography, socket, json, base64, entre outras. Para simplificar algumas operações como cifragem e decifragem de conteúdos criámos um conjunto de módulos num ficheiro "CipherHelper.py" que se baseou nas primitivas da library cryptography.

Segue-se como foi implementado o sistema e como funcionam alguns processos fulcrais:

4.1. Socket Connection

Quando o servidor inicia, estabelece uma sessão TCP/IP através de sockets por onde possam ser transportadas mensagens.

4.2. Diffie-Hellman

Quando um cliente pretende conectar-se ao servidor envia uma mensagem tipo "connect". O cliente do seu lado possui um par de chaves pública e privada onde transmite ao servidor o seu valor público. O servidor recebe o valor público do cliente e com a sua chave privada gera um "shared secret". Ao enviar o seu valor público ao cliente o cliente pode gerar o "shared secret" com a sua chave privada.

Durante este "handshake", existe um acordo entre algoritmos de cifragem. O servidor dispõem de algoritmos de cifragem que pretende usar com uma ordem de preferência. Quando o cliente pretende conectar-se ao servidor propõe ao servidor os algoritmos que pretende usar.

Se houver uma discordância entre o servidor e o cliente, isto é, se os algoritmos que o cliente propôs não se encontram na lista de algoritmos que o servidor pretende usar, então o servidor oferece ao cliente um dos seus algoritmos e o cliente optará por um deles.

Após a seleção do algoritmo a conexão será re-efetuada com sucesso.

4.3. Geração de novo par de chaves

O princípio usado no sistema de troca de mensagens é não utilizar o mesmo par de chaves em mensagens diferentes. A vantagem é que se um impostor descobrir o conjunto de chaves necessário para obter uma mensagem decriptada não poderá usar o mesmo conjunto de chaves para decifrar outra mensagem.

Após o cliente estar conectado com sucesso ao servidor este cria um novo par de chaves privado e público onde o “shared secret” irá ser gerado pela nova chave privada e o valor publico do servidor.

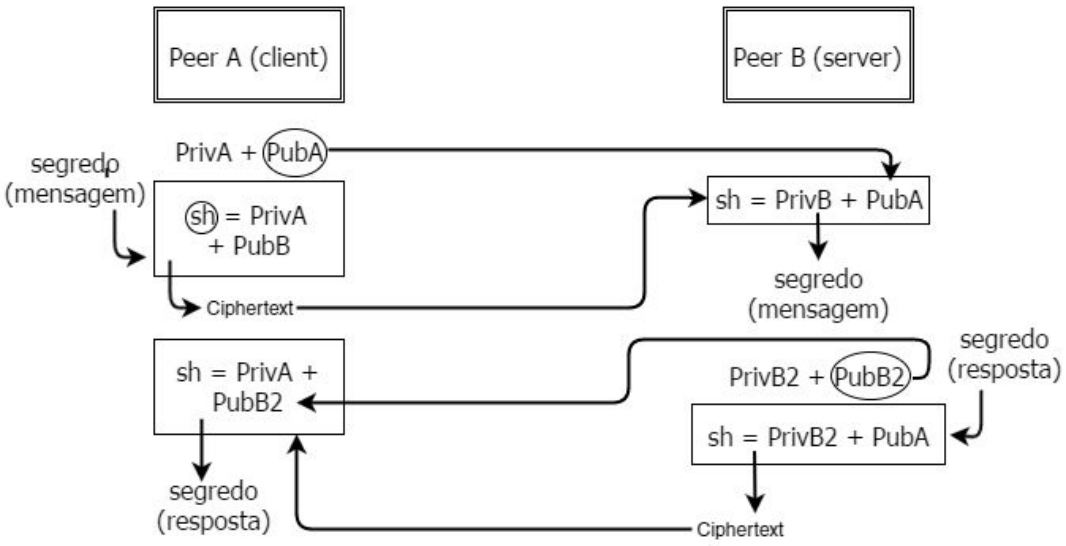
Quando o cliente envia uma mensagem este irá encriptar um segredo usando esse “shared secret” convertendo o segredo para um texto cifrado que será enviado para o servidor em conjunto com o novo valor público do cliente.

No lado do servidor este gera um novo “shared secret” com a sua chave privada e o novo valor público recebido do cliente. Com este “shared secret” o servidor consegue decriptar o texto cifrado do cliente e ler o segredo (mensagem).

Quando o servidor responde este cria um novo conjunto de chaves pública e privada e gera um novo “shared secret” com o seu novo valor privado e o novo valor publico recebido do cliente. A resposta é encriptada como este novo valor gerado e o texto cifrado segue para o cliente.

O cliente recebe o novo valor público do servidor e com este cria um novo “shared secret” em conjunto com a sua chave privada atual.

Segue-se um diagrama que sumariza o ciclo explicado:



4.4. Processo de cifragem

Os processos de cifragem ocorrem em dois instantes: cifragem de mensagens entre cliente-servidor e cifragem de mensagens entre clientes:

4.4.1. Mensagens entre Cliente-Servidor e vice-versa

Depois de se estabelecer uma conexão segura entre o cliente e servidor através do processo Diffie-Hellman, cliente e servidor podem trocar mensagens encriptadas usando o método escolhido no processo de conexão.

A cada nova mensagem a ser enviada, a entidade que está a enviar a mensagem gera um novo par de chaves, e utiliza a ultima chave pública que conhece do parceiro para gerar um novo segredo.

$$Sh = Priv_{newA} + Pub_{oldB}$$

A mensagem é então cifrada com o novo segredo e é enviada para o recetor juntamente com o chave pública gerada.

4.4.2. Mensagens entre Clientes

Para as mensagens entre clientes estas são cifradas com uma chave gerada no processo de conexão. Após ser acordado o algoritmo de cifragem a usar, quem inicia a conexão gera uma chave simétrica, e encripta-a com a chave pública RSA do cliente com o qual quer fazer a conexão. Este cliente ao receber a chave encriptada decripta-a com a sua chave privada. Assim, ambos os clientes obtêm o mesmo segredo sem que este possa ser descoberto pelo meio.

Neste tipo de mensagens a chave é sempre a mesma, pois como as mensagens ‘client-com’ são encapsuladas pelas mensagens ‘secure’, e as mensagens ‘secure’ são sempre encriptadas com chaves diferentes decidiu-se que o nivel de segurança era bastante bom para os requisitos aplicacionais.

4.5. Processo de decifragem

Os processos de decifragem ocorrem em dois instantes: decifragem de mensagens entre cliente-servidor e decifragem de mensagens entre clientes:

4.5.1. Mensagens entre Cliente-Servidor e vice-versa

Ao ser recebida uma mensagem 'secure', quem a vai decifrar (cliente ou servidor) necessita primeiro de saber com que chave é que ela foi cifrada.

Adjacente à mensagem vem uma chave pública de quem a enviou, que foi usada para gerar o texto encriptado. O recetor da mensagem aplica então a função para obter o segredo comum.

$$Sh = Priv_{oldB} + Pub_{newA}$$

Ao obter a chave com que foi cifrada a mensagem, basta decriptá-la usando o algoritmo acordado na fase de conexão.

4.5.2. Mensagens entre Clientes

O processo de decifragem nas mensagens entre clientes é bem mais simples. Como a chave é acordada no processo de conexão, não é alterada a cada mensagem, a ser recebida uma mensagem de um cliente, o receptor apenas tem de a decifrar com a chave partilhada entre eles.

4.6. Comandos

Para facilitar a utilização do programa segue em lista os comandos suportados e uma breve descrição dos mesmos:

- 'client-connect <id>' - Cria uma ligação segura com o cliente com <id>
- 'list' - Recebe uma listagem de clientes conectados ao servidor
- 'client-com <id> <message>' - Envia uma mensagem para o cliente com <id> . Apenas é permitida depois de haver uma conexão segura com o cliente.
- 'client-disconnect <id>' - Termina uma ligação segura com o cliente <id>
- 'peerlist' - Mostra todos os clientes com os quais tenho uma conexão segura.

5. Problemas

Ao implementar a aplicação, deparamo-nos com alguns problemas que não tínhamos detectado na fase de desenho. Alguns problemas foram possíveis de resolver, mas outros, dada a altura em que foram descobertos implicam uma mudança considerável na arquitetura da aplicação, pelo que a sua resolução não fica implementada neste *milestone,* mas já estão a ser tratados.

- O processo de geração de um novo segredo a cada mensagem enviada entre cliente e servidor falha caso uma mensagem seja enviada, mas não chegue ao destinatário.

- A chave usada para gerar o HMAC é a mesma usada para cifrar o conteúdo.

- A preocupação com as principais features levou a que a interface com o utilizador ficasse pouco polida.

6. Conclusão

Esta primeira fase do projeto foi desafiadora mas contudo ofereceu uma oportunidade de aprofundar o conhecimento de maneira mais concisa sobre a matéria lecionada nas aulas teóricas e práticas.

O tema em si é interessante pois aborda tecnologias que usamos todos os dias sem o utilizados comum se aperceber. Tecnologias essas que envolvem o desenvolvimento de sistemas de trocas de mensagens e de sistemas seguros.

Ao longo do desenvolvimento do projeto foram encontrados obstáculos sobre temas que já nos eram conhecidos e outros temas que desconheciamos. A aprendizagem sobre bases de certos temas que nos eram pouco familiares ajudou ao nosso crescimento como futuro engenheiros.

As dificuldades que surgiram que não foram possíveis de ultrapassar serão abordadas futuramente na segunda parte tendo em conta o seguimento do desenvolvimento do sistema.

Referências

- Criptography python library - *https://cryptography.io/*
- Segurança em Redes Informáticas, André Zúquete, 4º Ed. Aumentada, FCA - Editora de informática, Lda.