Universidade de Coimbra

Universidade de Coimbra

Segurança e Tecnologias de Informação

Trabalho Práctico III

Gonçalo Santos
2012140860 - gdsantos@student.dei.uc.pt

Ana Inês Mesquita Fidalgo
2013134819 - aimf@student.dei.uc.pt

2 de Junho de 2017

Conteúdo

Capítulo 1

Introdução

Este projecto foi realizado no âmbito da cadeira Segurança em Teconologias de Informação, STI, inserida no plano de estudos do Mestrado em Engenharia Informática da Universidade de Coimbra, lecionada pelos Professores Doutores João Paulo da Silva Machado Garcia Vilela e António Jorge da Costa Granjal, no ano lectivo de 2016/2017.

O projecto acima referido tem como objetivo a implementação de um serviço de chat seguro. Num sistema de chat tradicional não existem mecanismos de segurança, não suportando mecanismos de autenticação, nem a verificação de integridade e de autenticidade das mensagens recebidas, para além de outros aspectos.

A figura abaixo apresenta o cenário apresentado a implementar:

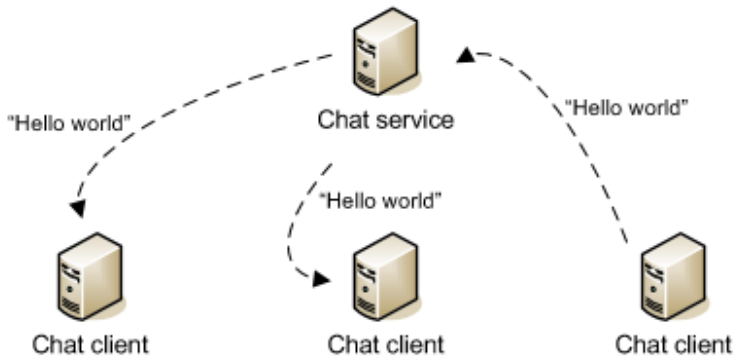

Fig. 1 Cenário a Implementar

Capítulo 2

Arquitectura

Neste projecto foram implementadas uma classe cliente e uma classe servidor e todos os métodos necessários para poderem estabelecer comunicação entre elas de maneira segura.

Foi importante ter em consideração uma implementação deste serviço de chat que permitisse a presença dos seguintes requisitos de segurança.

- **Confidencialidade** - Garantir que todas as comunicações entre os clientes e o servidor do serviço de chat estão protegidas, e para tal a utilização de uma chave simétrica, utilizando o key size considerado mais apropriado ao serviço.
- *Autenticidade* - Garantir que todas as mensagens no sistema são autenticas.
- *Integridade* - Garantir a integridade de todas as comunicações no sistema, impedindo a ocorrência de ataques *"Man in the Middle"*.
- *Não-repudio* - Garantir que todas as comunicações do sistema estão inequivocadamente associadas a uma entidade, seja cliente ou servidor.
- *Controlo de Acessos* - Recusar conneções de determinados clientes, adicionados a uma blacklist, através do seu nome.

Para além destes atributos de qualidade, tivemos também em conta os aspetos:

- *Key Management* - Um sistema que usa criptografia simétrica só é seguro se as chaves forem renovadas periódicamente.

- *Gestão Segura de Informação Confidencial* - Armazenar informação, como as private keys e passwords.

A figura 1 apresenta o cenário considerado neste trabálho práctico:

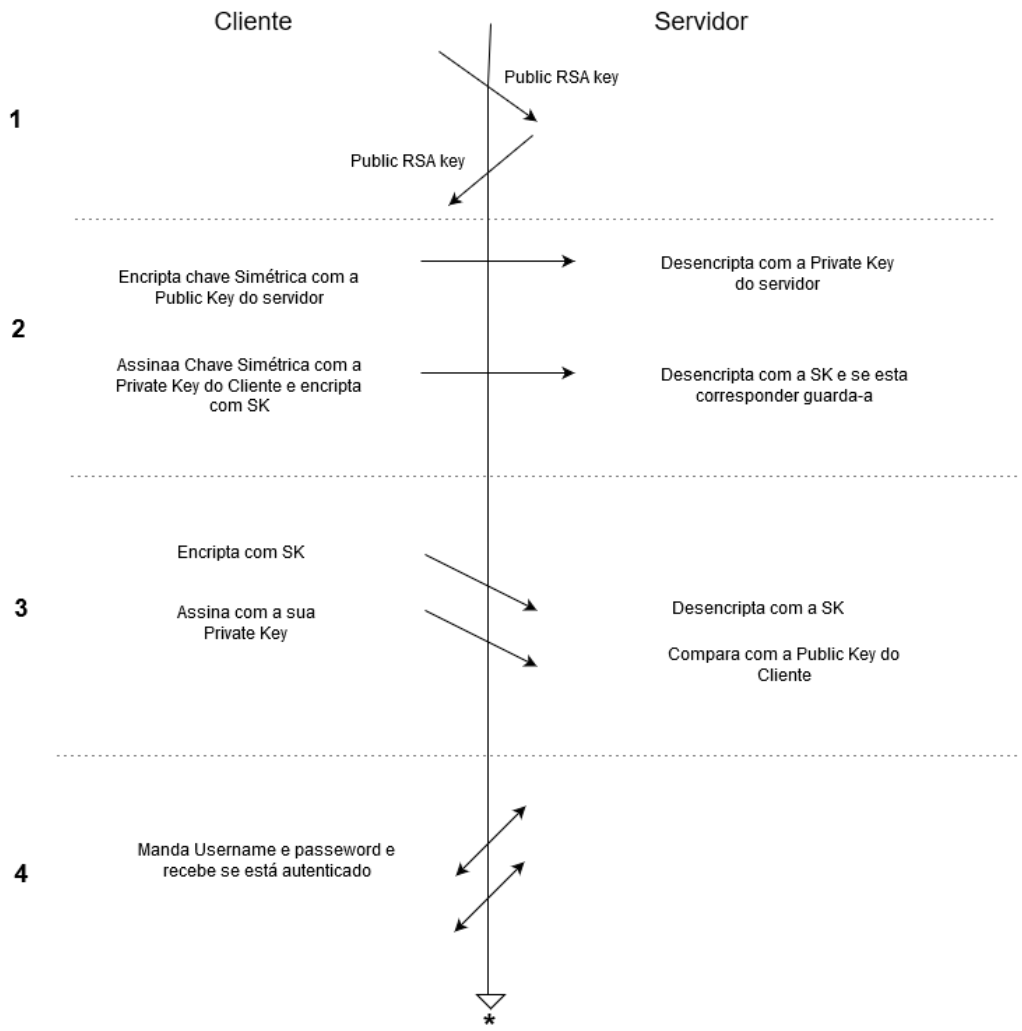

Fig. 2 Cenário Trabalho III

O estabelecimento da comunicação entre Servidor e Cliente divide-se por quatro fases:

- *1ª Fase* - Cliente e Servidor geram, enviam e recebem as Public Keys entre eles.
- *2ª Fase* - O Cliente vai encriptar a chave simétrica gerada com a public key que recebeu do servidor, enviando-a para o Servidor. O Cliente assina a chave simétrica com a sua private key e encripta-a com a SK. o Servidor, por consequência, irá desencripta-la com a sua private key.
- *3ª Fase* - O Cliente encripta a mensagem com a SK, assinamdo-a com a sua Private Key, enviando-a para o Servidor, este desencripta-a e compara-a com a Public Key do Cliente.
- *4ª Fase* - O Cliente manda as suas credenciais, username e password, ao Servidor, sendo que este verifica se este cliente se encontra "*blacklisted*", caso não esteja, o cliente recebe a informação de que está autenticado.

Estas fases são repetíveis, por exemplo ao fim do envio de dez mensagens entre um cliente e o servidor, a sua chave é renovada, tendo de se voltar a realizar as fases 2 e 3 acima descritas.

Capítulo 3

Comunicação Cliente-Servidor

Foi criada a classe *ChatClient*, onde inicialmente o servidor liga-se a um porto, no qual fica à espera do contato de um ou mais clientes. Quando esta comunicação é iniciada, o servidor inicialmente verifica se é possível realizar a ligação sem ultrapassar o seu número máximo de clientes, caso seja possível, é adicionada a thread do cliente correspondente. Num primeiro *"handshake"*(na função *firstHandShake()*), em primeiro lugar o Servidor e o Cliente geram as suas chaves. O Servidor irá receber a chave privada desse cliente e de seguida enviar a sua chave privada.

De seguida, na função *secondHandShake()*, recebe a chave simétrica encriptada com a sua public key no cliente e desencripta-a com a sua chave. Recebe a chave simétrica encriptada com a private key do cliente e encriptada com a SK, para verificar a autenticidade do cliente desencripta-a com a SK, caso seja correspondente irá guardar a chave simétrica como a chave definitiva desse utilizador, pois anteriormente tinha só uma chave temporária pois não tinha verificado a autenticidade do mesmo.

De seguida, ocorrem duas instâncias de comunicação entre esse cliente e o servidor na função *thirdHandShake()*: o Cliente encripta a mensagem com a sua Symetric Key e envia-a para o Servidor, por sua vez este desencripta-a, e numa segunda instância, o Cliente assina a mensagem com a sua Private Key. O Servidor recebe esta mensagem e compara-a com a *public key* do cliente. Na 4ª fase de comunicação, função *fourthHandShake()*, O Cliente envia o seu username e password, e o Servidor na função *handle*, para além de verificar sempre se o Cliente pediu para dar *refresh* da sua chave (através do comando ".refresh"ou pedir para terminar a comunicação (através do comando ".quit"), o Servidor também verifica se esse Cliente está *"blacklisted"*

através do seu username, (estes estão guardados num ficheiro localmente), caso esteja, irá impedi-lo de estabelecer mais qualquer contato e remove o id da *thread* correspondente desse Cliente.

O Servidor pode dar Blacklist a qualquer Cliente, guardando estas informações num ficheiro, que irá ser sempre verificado cada vez que uma autenticação for bem sucedida.

Tal como já tinha sido referido anteriormente, após dez mensagens entre o servidor e o cliente, a chave simétrica do cliente é renovada, voltando-se a suceder a função *secondhHandShake()* e *thirdHandShake()*.

Capítulo 4

Testes Realizados

Realizamos diversos testes enviando chaves erradas entre o Cliente e o Servidor, de modo a verificar se isto iria criar uma falha nos Handshakes descritos anteriormente, ou seja, caso se verifica-se que as chaves não eram correspondentes na desencriptação, se a comunicação Servidor-Cliente era interrompida, o que se pode verificar.

Para além destes testes, verificamos também que se a autenticação de um cliente não fosse bem sucedida a comunicação entre ele e o Servidor seria interrompida.

```
C:\Users\gonca\Desktop\STIP3\STITP3>java ChatClient localhost 8080
Establishing connection to server...
Connected to server: Socket[addr=localhost/127.0.0.1,port=8080,localport=29586]
1st handshake
2nd handshake
3rd handshake
4th handshake (login)
Insert your username
goncalo
Insert your password
ze
Wrong password

Handshake terminated

Error sending string to server: Stream closed
Listening error: socket closed

C:\Users\gonca\Desktop\STIP3\STITP3>
```

Fig. 2 Cenário de Autenticação Incorreta

Um outro teste realizado, foi a verificação de que se se adiciona-se um

determinado Cliente à *blacklist* do Servidor, o mesmo, mesmo que se autenticasse com sucesso iria ter a sua comunicação com o servidor interrompida.

Fig. 2 Cenário Trabalho III

Por outro lado, caso os quatro *handshakes* ocorressem sem qualquer problema e o cliente, ao autenticar-se, mandasse ao Servidor as suas credenciais corretas e não estivesse na sua *blacklist*, verificou-se que esta comunicação Cliente-Servidor poderia ser estabelecida, realizando-se testes para cenários com o Servidor e vários Clientes em simultâneo.

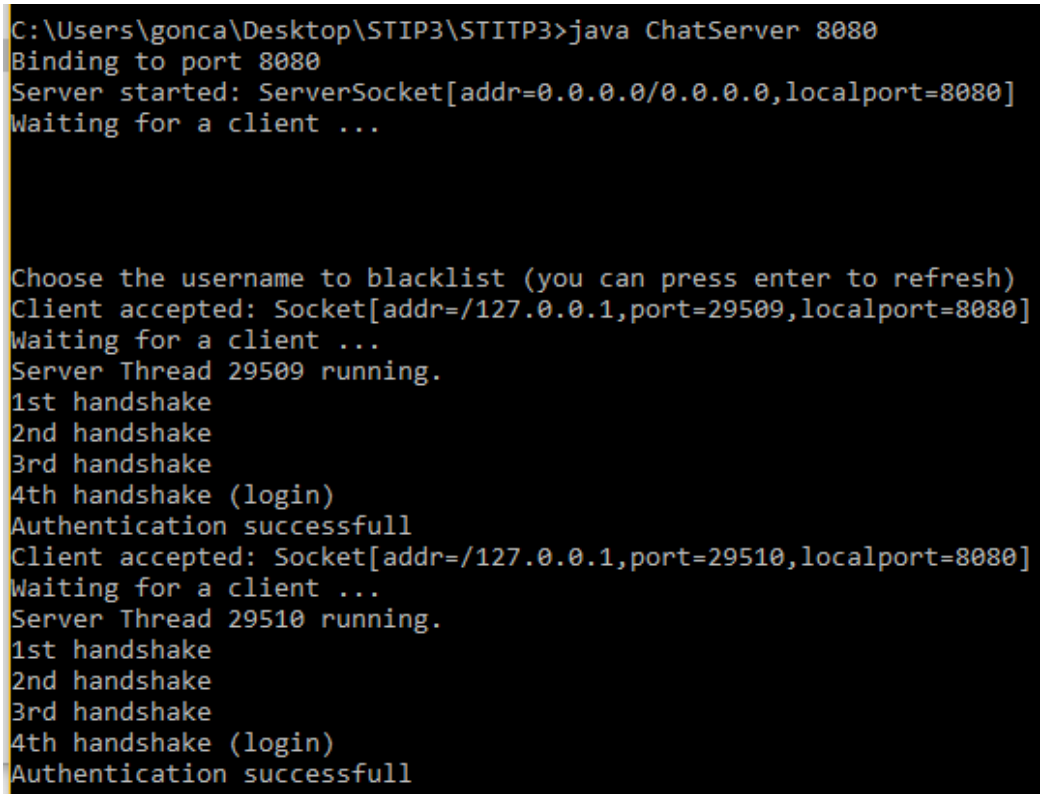

Fig. 2 Cenário Trabalho III

Pode-se também verificar a implementação correcta do refresh da chave simétrica a cada dez mensagens entre o Cliente e o Servidor.

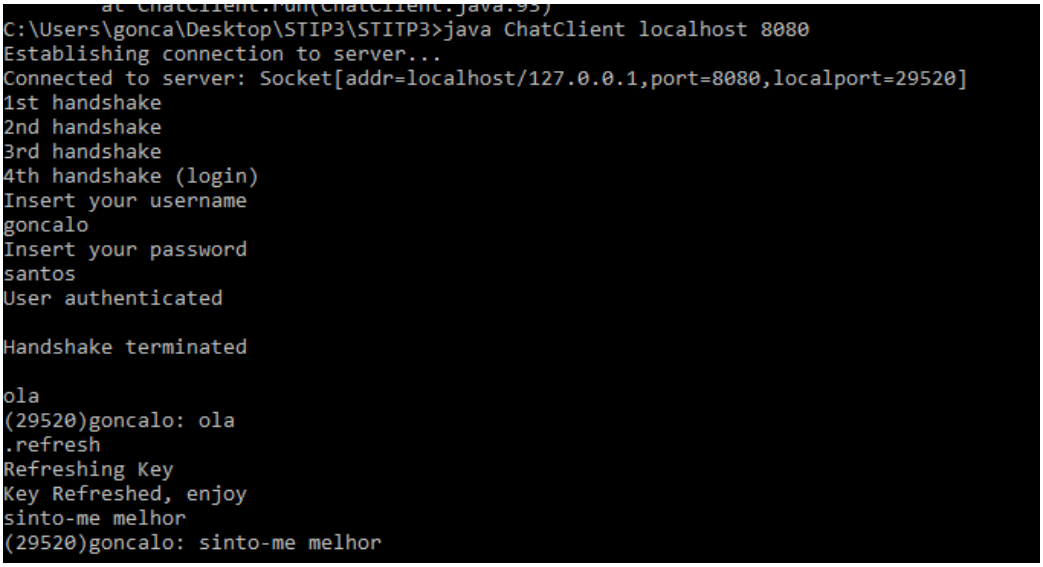

Fig. 2 Cenário Trabalho III

Com estes testes pudemos verificar que certos atributos de qualidade, apresentados acima, como a Autenticidade, o Controlo de Acessos, Confidencialidade, etc estavam a ser respeitados.

meter tabela a especificar testes

Capítulo 5

Anexos

- https://en.wikipedia.org/wiki/Public-key_cryptography
- https://en.wikipedia.org/wiki/Symmetric-key_algorithm
- Segurança Prática em Sistemas e Redes com Linux de Jorge Granjal