Ciências ULisboa | Informática

di.ciencias.ulisboa.pt

/difcul

**Segurança e Confiabilidade**

2015/2016

Trabalho 2

# 1 Objetivos

Esta fase do trabalho estende a anterior, possibilitando aos alunos experimentarem diversos mecanismos de segurança, tais como: cifras, assinaturas, comunicação com um protocolo seguro (TLS – Transport Layer Security) e gestão básica de certificados.
A envolvente do trabalho continua a ser a mesma, ou seja, a concretização de um sistema seguro de troca de mensagens de texto, fotos e vídeos, **myWhats**. O trabalho será realizado utilizando a linguagem de programação Java e a API de segurança do Java, e ferramentas complementares. Neste trabalho, tal como o anterior, o cliente e o servidor devem ser executados dentro da *sandbox*.

# 2 Modelo de adversário

Iremos assumir no trabalho que existe um adversário que pretende comprometer o correto funcionamento do sistema. O adversário terá um conjunto de capacidades que poderão ser empregues na realização das suas ações maliciosas. Torna-se assim necessário dotar o sistema dos mecanismos de proteção que lhe possibilitem manter um funcionamento correto ainda que se encontre sob ataque.

Vamos assumir que o adversário tem as seguintes capacidades:

- Acesso à rede : tendo o adversário acesso à rede, poderá escutar os pacotes trocados entre o cliente e o servidor. Potencialmente, também poderá tentar corromper, alterar, introduzir, e reproduzir mensagens de forma a enganar quer o cliente quer o servidor.

- Controlar um ou mais utilizadores : o adversário controla uma (ou mais) conta(s) de utilizadores do sistema. Através desta(s) conta(s), ele poderia tentar aceder a mensagens/fotos/vídeos para os quais não tem permissões ou corromper ficheiros com informação de outros utilizadores.

- Acesso à máquina onde corre o servidor em modo de leitura : o adversário tem acesso em modo de leitura aos ficheiros armazenados no servidor. Com esse acesso, ele pode potencialmente observar informação que eventualmente seria secreta.

Em seguida indicam-se e discutem-se as proteções que devem ser adicionadas ao sistema.

# 3 Proteções a adicionar ao sistema

Nesta fase, os alunos devem usar a mesma arquitectura da 1ª fase, e as funcionalidades a oferecer mantêm-se aproximadamente iguais. No entanto, o sistema será estendido de modo a ser garantida a sua segurança.

As alterações a introduzir são as seguintes:

1. O servidor deve **autenticar os utilizadores**. O utilizador durante a autenticação fornece o username e a password. Esta informação deve ser transmitida para o servidor protegida de ataques na rede (ver ponto 4). O servidor deve validá-la, e apenas se esta estiver correta deve ser dado acesso ao utilizador.
2. O servidor deve proteger a **privacidade das passwords** dos utilizadores. Com este objetivo, o servidor em vez de armazenar as passwords dos utilizadores em claro,

deve guardar uma síntese (hash) de "password:salt" (sem incluir as "). O salt é um número aleatório com 6 algarismos.

O servidor mantém a informação sobre os utilizadores registados num ficheiro. Para cada utilizador, este ficheiro contém a seguinte informação, separada pelo caracter ":" - identificador do utilizador, *salt* e *hash(password:salt)*.

O servidor deve também proteger a **integridade do ficheiro das *passwords***. Para tal, o ficheiro deve ser protegido com um MAC. O cálculo deste MAC utiliza uma chave simétrica calculada a partir de uma *password* que é pedida ao utilizador quando inicia a execução do servidor. No início da sua execução, o servidor deve usar o MAC para verificar a integridade do ficheiro. Se o MAC estiver errado, o servidor deve imprimir um aviso e terminar imediatamente a execução. Se não há MAC a proteger o ficheiro, o servidor deve imprimir um aviso e oferecer duas opções: terminar ou calcular o MAC e adicioná-lo ao sistema. O MAC deve ser verificado em todos os restantes acessos ao ficheiro e atualizado caso o ficheiro seja alterado.

Os utilizadores são adicionados ao ficheiro tal como definido na 1ª fase do trabalho: quando o comando *myWhats* é executado, caso o utilizador não esteja registado no servidor, efetua o seu registo, ou seja, adiciona este utilizador ao ficheiro das *passwords*.

3. A **integridade** dos restantes ficheiros de configuração do servidor deve ser assegura com um mecanismo idêntico ao utilizado para o ficheiro das passwords.

4. Na comunicação entre o cliente e o servidor pretende-se garantir a **autenticidade do servidor** (um atacante não deve ser capaz de fingir ser o servidor e assim obter a password de um utilizador) e a **confidencialidade** da comunicação entre cliente e servidor (um atacante não deve ser capaz de escutar a comunicação). Para este efeito, devem-se usar **canais seguros** (protocolo TLS/SSL). Este protocolo permite verificar a identidade do servidor utilizando chaves assimétricas.

   - Ligações TLS: Deve-se substituir a ligação TCP por uma ligação TLS/SSL. O protocolo TLS vai verificar a autenticidade do servidor e garantir a integridade e confidencialidade de toda a comunicação.
   - A utilização do protocolo TLS exige configurar as chaves tanto no cliente (truststore com o certificado do servidor) como no servidor (keystore com a sua chave privada).

5. As mensagens, fotos e vídeos devem ser protegidos de eventuais ataques que possam ocorrer na máquina servidora, nomeadamente que tenham em vista **observar o seu conteúdo ou personificar a origem**. Para isso, sugerimos a utilização de criptografia híbrida e assinaturas digitais (em alternativa a envelopes seguros para simplificar a implementação). A ideia seria aplicar genericamente o seguinte algoritmo quando se pretende enviar uma mensagem ou um ficheiro para um contato (utilizador ou grupo):

   I. O cliente pede ao servidor a informação sobre o contato. O servidor responde informando-o se o contato é um utilizador ou um grupo, e no caso de ser um grupo, o servidor envia também ao cliente a lista dos utilizadores que pertencem a esse grupo.

   II. O cliente gera a assinatura digital da mensagem ou do ficheiro em claro e envia-a para o servidor. O servidor guarda-a num ficheiro com extensão .sig

   III. O cliente gera uma chave simétrica *K* AES aleatoriamente

   IV. Cifra a mensagem ou o ficheiro com *K* usando AES e envia-a(o) para o servidor

   V. O cliente cifra K usando a(s) chave(s) pública(s) do utilizador local e do(s) contato(s) com os quais está a trocar a mensagem/ficheiro e envia cada uma das cifras para o servidor. O servidor armazena-as em ficheiros separados com

extensão *.key.user* (por ex., o ficheiro *fich.txt* enviado por Alice para Bob teria a chave armazenada em *fich.txt.key.Alice* e em *fich.txt.key.Bob*)

Quando um utilizador lê o ficheiro do servidor, deve ser executado o processo inverso ao definido anteriormente:

I. O servidor envia para o cliente os conteúdos do ficheiro cifrado e do respetivo ficheiro com a chave K cifrada.

II. O cliente obtém K decifrando-a com a sua chave privada do utilizador

III. K deve ser usada para decifrar o conteúdo do ficheiro

IV. O servidor envia para o cliente a assinatura do ficheiro. O cliente verifica a assinatura do ficheiro.

6. Quando um utilizador é adicionado a um grupo, deve passar a ter acesso às **futuras** mensagens e ficheiros que são trocados. Toda a informação que foi enviada anteriormente, não deve ser fornecida a esse utilizador. Em particular, ele não tem acesso às chaves K (dado que não foram criados os ficheiros *.key.user* quando as mensagem/ficheiros foram criados originalmente).
7. Quando um utilizador é retirado de um grupo, este deixa de ter acesso às mensagens/ficheiros trocados anteriormente e aos que venham a ser enviados no futuro. O servidor, para além de atualizar a informação de constituição do grupo, deve apagar todos os ficheiros *.key.user* referentes a mensagens/ficheiros para/de esse utilizador para o grupo.

NOTA: Toda criptografia assimétrica no projeto deve usar RSA com chaves de 2048 bits. A criptografia simétrica deve ser efetuada com AES e chaves de 128 bits. O algoritmo de síntese deve ser o SHA-256.

# 4 Relatório e discussão

No relatório devem ser apresentados e discutidos os seguintes aspetos:

- Os objetivos concretizados com êxito
- Os problemas encontrados.
- A segurança da aplicação criada, identificando possíveis **fraquezas e melhorias** a incluir em versões futuras.

O relatório deve ter no máximo 5 páginas (sem contar com o código) e **é necessário incluir o código fonte na versão em papel.**

# 5 Entrega

1. **Código.** Dia **1 de Maio**, até as 23:55 horas. O código do trabalho deve ser entregue da seguinte forma:
   1. Os grupos devem inscrever-se atempadamente de acordo com as regras afixadas para o efeito, na página da disciplina.
   2. Na página da disciplina submeter o código do trabalho num ficheiro zip.
2. **Relatório.** Dia **2 de Maio**, até as 18:00 horas.
   1. Na página da disciplina submeter o relatório num ficheiro pdf e um readme (txt) sobre como executar o trabalho.
   2. No cacifo do professor das TPs, a impressão do relatório e do código fonte.

Não serão aceites trabalhos por email nem por qualquer outro meio não definido nesta secção. Se não se verificar algum destes requisitos o trabalho é considerado não entregue.