Segurança de Redes e Sistemas de Computadores
2015/2016, 2º Semestre

Trabalho Prático nº 1 (v1.0)

Resumo

Neste pretende-se desenvolver um sistema de CHAT/MESSAGING seguro para utilização de grupos de utilizadores envolvidos em sessões de comunicação do tipo conferência (Chat/Messaging) em grupo. Este sistema precisa que se estabeleçam canais de comunicação seguros entre os participantes, capazes de criarem contra-medidas contra eventuais ataques de oponentes que visam atacar a comunicação entre os utilizadores corretos. O sistema deverá ser capaz de garantir propriedades de autenticação, confidencialidade, integridade e não repudiação, defendendo da tipologia de ataques com base na referência OSI X.800 estudada nas aulas e estabelecendo canais seguros de comunicação num ambiente de interligação que pode ser a INTERNET. O trabalho será realizado a partir de um conjunto de materiais inicialmente disponibilizados. A solução será produzida a partir da extensão desses mesmos materiais, de acordo com as indicações a seguir enunciadas e conforme a metodologia indicada que se sugere para desenvolvimento em duas fases.

1. Introdução

Pretende-se desenvolver um sistema de CHAT/MESSAGING seguro, para utilização por parte de grupos de utilizadores envolvidos em sessões de conferência em grupo (*Group-chat/conference*). O sistema garantirá propriedades de autenticação, confidencialidade, integridade e não repudiação, estabelecendo um ambiente de canais de comunicação fiável e segura, sobre um ambiente de interligação INTERNET. Para o efeito, o sistema será implementado como base de comunicação fiável sobre transporte TCP (pilha TCP/IP), à qual serão acrescentadas as propriedades de segurança requeridas.

O trabalho partirá de um conjunto de materiais inicialmente disponibilizados (*package* com código fonte JAVA), devendo a solução ser produzida a partir desse material. A realização do trabalho envolve programação em JAVA, utilização de sockets TCP/IP e a utilização do suporte JCE (*Java Cryptographic Extension*), nomeadamente a programação com técnicas, métodos e algoritmos criptográficos. Neste trabalho o enfoque será na utilização de criptografia simétrcia em funções de síntese segura de mensagens. O conhecimento e o domínio prático das ferramentas criptográficas decorre da matéria leccionada nas aulas teóricas de SRSC, bem como dos exercício ou exemplos das aulas práticas em laboratório.

2. Materiais, ambientes e ferramentas a utilizar

No trabalho serão usados os seguintes materiais e ferramentas (algumas das quais do conhecimento anterior adquirido pelos alunos).

Material inicial

- Uma aplicação *chat/messaging* em grupo, disponível com código fonte no CLIP (ver em materiais/protocolos/trabalho1-sources.tgz). Trata-se de uma aplicação que utiliza um ou mais clientes (que podem estar distribuídos numa rede local ou num ambiente do tipo internet) e um servidor (que implementa a funcionalidade de relay da comunicação entre os diversos clientes.
- Verificar o arquivo disponibilizado e o ficheiro README existente

Ferramentas de prática e conhecimento anteriores:

- Utilização de sistemas LINUX, MAC-OS ou WINDOWS e ambiente de desenvolvimento JAVA (ref. Java 7 ou Java 8). A plataforma de referencia para demonstrações em ambiente laboratorial é: LINUX (distribuição Ubuntu, conforme instalações no laboratório 123, Ed. II – Departamento de Informática, FCT/UNL.
- Programação com *Sockets* TCP/IP em Java (pacote java.net)
- Serialização de Objetos passados em *sockets data-stream* (ou *sockets* TCP)
- Domínio da plataforma de desenvolvimento Eclipse IDE (ou outra equivalente) mas também utilização do ambiente e comandos SHELL com controlo do lançamento e execução de processos, bem como domínio do sistema de ficheiros.

Ferramentas e conhecimento prático a adquirir no desenvolvimento do trabalho

- Programação com o pacote JAVA JCE (*Java Cryptographic Extension*), nomeadamente utilização de criptografia simétrica, utilização de funções seguras de síntese de mensagens e funções de cálculo e verificação de códigos de autenticação de mensagens
- Utilização de bibliotecas de algoritmos criptográficos disponibilizadas por provedores criptográficos para o ambiente JAVA JCE

3. Modelo e arquitetura do sistema a desenvolver

O modelo do sistema a desenvolver está representado na seguinte figura 1.

Um conjunto de clientes (usados por diversos utilizadores distribuídos num ambiente do tipo Internet) vão realizar sessões de conferencia em grupo (*group chat-conferecing ou messaging*). Estas sessões são seguras, suportadas por canais de comunicação fiáveis (suportados em TCP na pilha TCP/IP) e seguros (garantindo propriedades de autenticação, confidencialidade, integridade e não repudiação de todos os fluxos de mensagens trocadas entre todas as entidades do sistema, de acordo com a definição destas propriedades e respetivas garantias na *framework* X.800 estudada nas aulas.

As entidades principais do sistema (a partir do material fornecido) são as seguintes:

Cliente (*ClientSketcher*): representa um utilizador que se autenticou inicialmente (*usename/password*) no servidor SSO Server , e uma vez autorizado (através da política de controlo de acesso escrutinada pelo SSO Server está em condições de participar na sessão.

Server: (MCRServer) é um servidor (*relay*) que implementa a funcionalidade da conferencia. Todas as mensagens enviadas por um utilizador correto (cliente), são corretamente enviadas para todos os outros utilizadores (clientes) que se encontram na sessão da conferência representada pelo Server. Deve notar-se que cada instância do Server representa uma conferencia (*group chat-room ou messaging server*). Num modelo genérico podemos ter pois tantas instâncias de Servers quantas as conferencias que quisermos. Assim, um conjunto de Servers funcionaria como um *cluster* de servidores disponíveis para suportar diversas conferencias em paralelo[1].

SSO Server: (servidor inexistente no material fornecido): é um servidor que tem dois papéis: funciona como servidor de autenticação dos utilizadores (clientes) que estão interessados em participar numa dada conferência, e funciona como servidor de controlo de acesso.

[1] Por exemplo, estes servidores poderiam constituir recursos distribuídos (por exemplo instalados e prontos a usar em diferentes máquinas virtuais de um ou mais provedores *Cloud*)

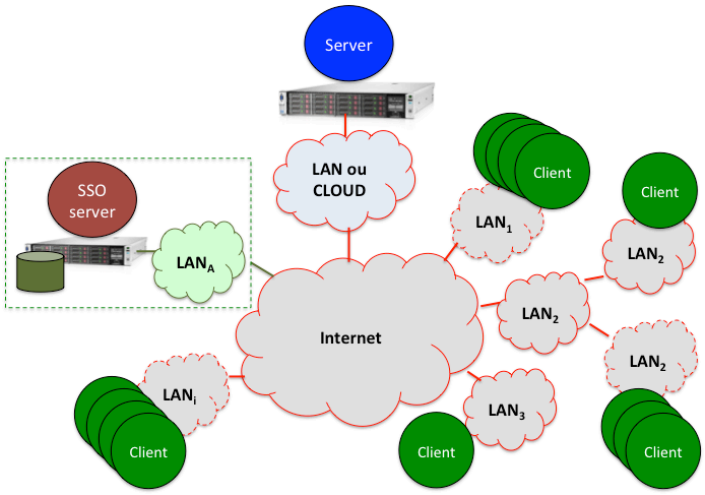

Figura 1. Modelo do sistema a desenvolver

Para o efeito, o servidor **SSO Server** gere um ficheiro (**conferences.conf**) com uma lista de conferencias registadas que vai controlar, Cada lista tem diversas entradas, na seguinte forma:

ConferenceID: Descrição: Endpoint

Exemplo:

12345678: "Conselho de Administração 21/2/2016 15h00, Assunto: Distribuição de dividendos a acionistas": 193.136.122.136

ConferenceID é um identificador (único); a descrição é uma String (conteúdo livre) que descreve a conferência e o *Endpoint* é o par "*endereçoIP:porto*" associado ao Server (*Relay*) para a respetiva conferência.

Para além do ficheiro da lista de conferências, o SSOserver gere um ficheiro de controlo de acessos (**acesscontrol.conf**) com base num modelo discricionário de utilizadores autorizados a participarem nas conferencias registadas, com o formato seguinte:

conferenceID: user1, user2, user3, user4

Cada utilizador pode ser descrito neste ficheiro de acordo com um userID (pode ser um identificador do tipo EMAIL RFC 822 (ex., fulano@organizacao.pt)

Existe também um ficheiro (**userauth.conf**) semelhante ao ficheiro /etc/password num sistema UNIX, contendo informação de associação das passwords associadas aos respetivos utilizadores. Na sua versão mais simples, cada utilizador tem uma linha neste ficheiro na forma:

user:password

Finalmente, o SSO Server possui ainda outros ficheiros de configuração adicionais com informação criptográfica, o que será explicado mais à frente

4. Desenvolvimento do trabalho e metodologia sugerida

Para desenvolver o trabalho deverá seguir uma metodologia em duas fases. A implementação da primeira fase é obrigatória e vale até 14/20 valores (ou 3,5/5 pontos). A segunda fase é valorizada até 6/20 valores (1,5/5 pontos).

4.1 FASE 1 (Até 3/5 pontos)

Deverá implementar um modelo limitado do sistema indicado em 3, atendendo aos seguintes requisitos

- Proteção das sessões com base na implementação de canais seguros, considerando um modelo de adversário que realiza ataques aos canais de comunicação entre todas as entidades, segundo a tipologia de ataques definida na *framework* X.800 (não considerando ataques á disponibilidade ou ataques do tipo *DoS* ou *DDoS*).
- Proteção da confidencialidade e integridade das mensagens nas sessões, mesmo perante possíveis oponentes do tipo "*honest-but-curious system administrators*" que administrem os sistemas onde executam servidores *relay* (*MCServer*) associados a diferentes sessões de *Chat/Conferencing* e possam inspecionar a memória desses processos. Note que se admite que estes oponentes não atentam contra a disponibilidade desses sistemas, nem comprometem a correção do software que executa nos mesmos.

Tendo em conta os requisitos, procederá a um suporte básico de configuração no servidor *relay* que possuirá um ficheiro de configuração com informação criptográfica, contendo chaves criptográficas e parâmetros de segurança. Para cada conferência registada existe um ficheiro (cujo nome é o do identificador da conferência, conforme aparece na lista de conferências) que tem o seguinte formato base:

```
SESSIONKEY: Session Key     // Representação da Chave Criptográfica
KEYSIZE: size (bytes)       // Tamanho da chave
ALG: Algorithm              // Algoritmo criptográfico simétrico, ex: DESede, AES, Blowfish,
                               RC6, Toofish, etc...
IV: Initialization Vector   // Vetor de Inicialização, para modos como CBC, CTR, OFB, CFB, ...
MODE: mode                  // Modo de cifra utilizado: ECB, CBC, CTR, OFB, CFB, ...
PAD: padding method         // Método de Padding usado, ex: PKCS5, PKCS7
MAC: Algorithm              // Método MAC, HMAC ou CMAC a usar
HMACKEY: HMAC Key           // Chave HMAC ou CMAC
HMACKEYSIZE: size (bytes)   // Tamanho da chave
```

Opcionalmente pode adicionar outras configurações que considere interessantes. Notar que a ideia é flexibilizar por configuração o uso parametrizável de quaisquer métodos criptográficos na proteção das sessões.
A abordagem mais simples consiste em partilhar o ficheiro entre todos os utilizadores que possam ter acesso às sessões.

Na primeira fase a ideia será focar-se:
- Na proteção das comunicações
- Com base na concepção de um protocolo de encapsulamento das mensagens num formato de segurança (ver em anexo).

4.2 FASE 2 (Até 2/5 pontos)

A ideia agora é evoluir o primeiro protótipo uma implementação numa implementação em que passa a existir:

- O servidor de autenticação (que é independente de qualquer servidor *relay* de sessões específicas.
- Os utilizadores para participarem nas sessões devem autenticar-se no servidor de autenticação, que escrutinará da correção da autenticação e decidirá com base na política de permissões e (*enforcement*) de controlo de acesso, se um dado utilizador pode ou não entrar numa dada sessão.
- Se um utilizador for autorizado, receberá dinamicamente as credenciais para participar na sessão (ou seja, a informação que antes estava partilhada em ficheiros com gestão manual)

Para cada utilizador registado no servidor de autenticação existe um ficheiro como se segue (cujo nome é o do *userID*), com os parâmetros necessários partilhados com cada utilizador. Este ficheiro permite suportar o protocolo de autenticação e estabelecimento dos parâmetros de associação de segurança (inspirado no modelo Needham Schroeder na

variante que usa apenas criptografia simétrica), servindo como parâmetros de configuração Bootstrap a usar e partilhar entre os clientes e o SSO Server.

SESSIONKEY: Session Key	// Representação da Chave Criptográfica
KEYSIZE: size (bytes)	// Tamanho da chave
ALG: Algorithm	// Algoritmo criptográfico simétrico, ex: DESede, AES, Blowfish, RC6, Toofish, etc...
IV: Initialization Vector	// Vetor de Inicialização, para modos como CBC, CTR, OFB, CFB, ...
MODE: mode	// Modo de cifra utilizado: ECB, CBC, CTR, OFB, CFB, ...
PAD: padding method	// Método de *Padding* usado, ex: PKCS5, PKCS7
MAC: Algorithm	// Método MAC, HMAC ou CMAC a usar
HMACKEY: HMAC Key	// Chave HMAC ou CMAC
HMACKEYSIZE: size (bytes)	// Tamanho da chave

Para distribuição dos parâmetros criptográficos da sessão, o servidor de autenticação atua como KDC (*Key Distribution Center*). Para tal, este deverá implementar para o cliente o protocolo de *Needham-Shroeder*. Este protocolo, usado como protocolo de distribuição autenticada de chaves, numa variante que usa apenas criptografia simétrica – e que será a base de implementação para a fazer 2 do trabalho, será um tema a desenvolver na aula.

Formato de mensagens

Todas as mensagens envolvidas nos fluxos de informação entre quaisquer entidade deverão serão suportadas num formato genérico seguro e que pode ser instanciado para qualquer tipo de mensagem trocada (seja entre clientes e o *SSOServer* seja entre clientes e o *MCRServer*) como se especifica a seguir (atender à notação discutida e usada nas aulas). Claro que na primeira fase, este formato apenas será usado entre os clientes e possíveis servidores *MCServer*.

Mas será interessante que mesmo na fase 1, este formato seja já concebido de forma a vir a acomodar os requisitos da fase 2. Um formato de referencia que pode usar como ponto de partida da sua implementação encontra-se em anexo.

5.3 Considerações finais

Para além das pontuações base referentes as fases 1 e 2, a sua implementação poderá ser valorizada adicionalmente com um bónus até1 ponto em 5, atendendo-se à satisfação dos seguintes critérios:

- Robustez e correção da solução
- Correção e completude dos mecanismos de segurança face ao modelo de adversário
- Completude das duas fases
- Flexibilidade de configuração para poder acomodar transparentemente a utilização de diferentes ciphersuites em quaisquer configurações, ao nível dos ficheiros de configuração
- Eventuais opções valorativas que queira introduzir na sua implementação
  // que pode endereçar discutindo-as com os docentes

Avaliação

Deve ter em conta que a avaliação total do TP1 para efeitos de frequência será feita com base na valorização do trabalho prático propriamente dito e do respetivo teste prático, numa proporção de referencia de 70% e 30% respetivamente. O conhecimento em detalhe de toda a implementação e dos fundamentos que a envolvem serão factores importantes na avaliação do teste prático. Para este teste, deverá fazer-se acompanhar do código do trabalho (listagens) pois ser-lhe-ão solicitadas perguntas ou aspectos que podem ser iterados a partir dessa listagem, devendo a mesma ser entregue conjuntamente com as restantes respostas do teste prático.

Entrega do trabalho

A entrega do trabalho deverá ser feita nos prazos indicados (de acordo com a informação no sistema CLIP). Envolve a entrega do pacote de implementação com o código e informação de reporte sobre a mesma. O formato do relatório e as instruções de entrega (que será feira por submissão por via electrónica), serão detalhadas num documento próprio a disponibilizar oportunamente.

ANEXO

Formato Genérico (partes da mensagem transportada num *payload* TCP):

Nota: ver a apresentação usada na apresentação do TP1 na aula prática e que se encontra também no sistema CLIP.

HEADER || TIPO || MENSAGEM || AUTENTICADOR

HEADER contem dois campos inteiros:
H1: Codifica a fase da implementação: Ex.: 1 (fase 1), 2 (fase 2)
H2: Inteiro com uma versão do protocolo: Ex. 1, 2, 3, ...
H2: Tamanho total da mensagem em bytes

TIPO: codifica a mensagem de acordo com o fluxo:
10: mensagem de LOGON de cliente para o *SSOServer* no protocolo NS
11: mensagem de resposta de LOGON do *SSOServer* ao cliente no protocolo NS
19: mensagem de resposta de LOGON com indicação de não autorização de participação na sessão
99: mensagem de erro ou exceção

20. mensagem de envio de mensagem numa sessão (enviada do cliente ao *MCRServer*) para difusão aos outros participantes
21 mensagem difundida pelo *MCRServer* para os clientes na sessão

Poderá codificar como tipos de mensagens, ou proceder a extensões no formato, de modo a suportar o seu trabalho e as suas ideias que visem valorizar o mesmo. Poderá discuitir essas ideias com os docentes.

MENSAGEM
Contem as seguintes partes (concatenadas)

{USER-ID: R1: R2: DADOS}K Mensagem cifrada contendo um UserID do emissor, valores R1 e R2 para *challenge/response* para prevenir *message-replaying* e DADOS – os dados da mensagem. Note que todas as partes da MENSAGEM são confidenciais, estando cifradas com um algoritmo simétrico, com base num dado modo e *padding* utilizados.

AUTENTICADOR:
Possui um código de autenticação da mensagem, sendo este código calculado por uma função do tipo HMAC, com base numa chave válida em cada caso.