Segurança de Redes e Sistemas de Computadores 2017/2018

Ficha de Reporte do Trabalho Prático nº 1 (TP1)

Grupo

Nº de Aluno	Nome (elementos do grupo)
45404	Daniel Filipe Santos Pimenta

1. Introdução e contexto do trabalho

Indique X conforme o seu caso

Implementação e completude do trabalho	SIM	NÃO	PARC. (Parcialmente)
Foram implementados totalmente todos os requisitos da FASE 1 (ou protocolo STGC/TLP)	X		
Foram implementados totalmente todos os requisitos da FASE 2 (ou protocolo STGC-SAP)	X		
A minha implementação da FASE 1 (ou implementação do STGC/STGC-TLP) concretiza completamente e exatamente as especificações desse protocolo que constam do enunciado	X		
A minha implementação da FASE 2 (ou implementação do STGC/STGC-SAP) concretiza completamente e exatamente as especificações desse protocolo que constam do enunciado			X

Se colocou X anteriormente em alguma posição PARC /Parcialmente do quadro, indique porque o fez e porque considera que a implementação é parcial. Se não deixe em branco ou indique N/A

A forma como o recetor da mensagem sabe qual o ticket a utilizar para obter os parâmetros e decifrar a mensagem pode não ser a mais correta/indicada.

O recetor após ir ao servidor buscar o ticket, guarda-o num HashMap onde a chave é o multicast ip e o valor é o objeto ticket. Após receber uma mensagem cifrada, é necessário saber qual o endereço ip multicast que o emissor colocou no pacote e assim conseguir ir buscar o ticket ao hashmap e utilizá-lo para decifrar a mensagem. Mas a linguagem Java não disponibiliza forma de saber qual o endereço destino que o emissor colocou no pacote. A forma como eu estou a buscar esse endereço ip é enviá-lo em claro juntamente com a mensagem cifrada, na parte do emissor, e obtê-lo do lado do recetor.

A solução adequada provavelmente incluiria o servidor AS e o ficheiro dacl.conf.

2. Generalidade do desenvolvimento do protocolo STGC (Subprotocolo STGC-TLP) e sua evidência

Para suportar a aplicação de teste fornecida (testeMulticast) e para que esta seja protegida pela implementação do protocolo STGC-TLP, dado o código inicial (sem proteção da comunicação) dessa aplicação:

2.1 Apenas foi necessário modificar ___2___ linhas de código, em relação ao número de linhas de código da aplicação inicial

2.2 É preciso modificar ___2___ linhas de código em relação ao número de linhas de código inicial, tendo ainda que se acrescentar mais __0___ linhas de código em relação ao código inicial

Diga em que consiste no essencial a modificação do código da aplicação para ser protegida pela sua implementação com o STGC/TLP:

Em vez de ser utilizado o MulticastSocket, foi utilizada uma classe STGCMulticastSocket que foi desenvolvida para implementar o subprotocolo STGC-TLP (e parte do STGC-SAP).

Em ambos MulticastReciever e MulticastSender, a linha que criava um objeto MulticastSocket (MulticastSocket s = new MulticastSocket()) passou agora a criar um objeto STGCMulticastSocket (STGCMulticastSocket s = new STGCMulticastSocket()).

Foi também necessário modificar a mensagem enviada pelo MulticastSender para incluir o seu username e email. O username e email estão no formato <username/email>.

3. Caracterização da implementação do protocolo STCG / subprotocolo STGC-TLP

A minha implementação do subprotocolo STGC foi feita do seguinte modo (caracterize com uma boa síntese, como construiu e desenvolveu o suporte do protocolo STGC/STGC-TLP.

O desenvolvimento para o suporte do protocolo STGC-TLP foi feito na classe STGCMulticastSocket que ao substituir a classe MulticastSocket permite proteger mensagens UDP enviadas por multicast contra o modelo de adversário apresentado no enunciado.

A mensagem enviada por um emissor é intercetada pelo STGCMulticastSocket (no método send(DatagramPacket packet)) onde é feita a seguinte transformação:

$$M' = IPMC \,||\, HEADER \,||\, PAYLOAD$$

Onde IPMC é o endereço ip destino do pacote, header é o cabeçalho do pacote que é composto pela Version-Release, Payload_type e Payload_size, e o payload é o conteúdo do pacote cifrado para proteger a informação mais o MAC para verificar a integridade e autenticidade da mensagem. O payload é gerado utilizando os parâmetros que estão no ticket obtido anteriormente através do protocolo STGC-SAP.

A mensagem recebida por um emissor é intercetada pelo STGCMulticastSocket (no método recieve(DatagramPacket packet)) onde é obtido o endereço ip destino, são comparados os MACs e decifrada a mensagem utilizando o ticket que foi obtido anteriormente através do protocolo STGC-SAP. São também feitos nesta altura outras verificações como, por exemplo, o nounce para evitar ataques de repetição.

4. Comprovação da correção da implementação do protocolo STGC-TLP

4.1 Utilizei como aplicação de comprovação e prova do funcionamento da minha implementação STGC/STGC-TLP	SIM	NÃO
a) a aplicação MCHAT	X	
b) a aplicação STREAMING		X

4.2 Nas minhas observações experimentais, a aplicação protegida pela minha implementação do protocolo STGC/STGC-TLP:	SIM	NÃO
a) Funciona corretamente	X	
b) Funciona bem, mas apenas parcialmente		X

Justifique, apenas no caso de ter respondido SIM a 4.2 b). Se não deixe em branco ou coloque N/A

N/A

5. Flexibilidade e configuração de parametrizações de segurança para a execução do protocolo STGC/STGC-TLP

A minha implementação STCG/STGC-TLP segue as especificações do enunciado do trabalho, sendo *os endpoints de comunicação* parametrizáveis pelos seguintes ficheiros (configuração):	SIM	NÃO
5.1 Ficheiro de configuração ciphersuite.conf	X	
5.2 keystore.jceks	X	

5.3 Uma configuração tipo no ficheiro ciphersuite.conf pode ser estabelecida do seguinte modo (exemplifique):

```
<root>
        <room ip = "224.10.10.10">
                <ciphersuite>AES/CBC/PKCS7Padding</ciphersuite>
                <keysize>*</keysize>
                <keyvalue>*</keyvalue>
                <mac>HMAC-SHA512</mac>
```

```
            <mackeysize>*</mackeysize>
            <mackeyvalue>*</mackeyvalue>
        </room>
        <room ip = "224.10.10.11">
            <ciphersuite>AES/CBC/PKCS5Padding</ciphersuite>
            <keysize>*</keysize>
            <keyvalue>*</keyvalue>
            <mac>HMAC/SHA384</mac>
            <mackeysize>*</mackeysize>
            <mackeyvalue>*</mackeyvalue>
        </room>
</root>
```

5.4 Com o suporte de configuração **ciphersuite.conf** e com a geração / utilização adequadas (correspondentes) do **keystore.jceks**, verifiquei que se suportarão de forma flexível quaisquer combinações criptográficas. No meu caso testei e comprovei experimentalmente as seguintes:

```
LISTA DE CIPHERSUITES testadas com sucesso: (ALG/MODO/PADDING):
AES/CBC/PKCS5Padding
AES/CTR/PKCS7Padding
AES_128/OFB/NoPadding
AES_256/OFB/NoPadding
```

```
LISTA DE MACs (HMACs ou CMACs) testadas com sucesso:
HMAC-SHA512
HMacSHA1
HMAC/SHA384
HMAC-SHA3-224
HMAC-SHA3-256
HMAC-SHA512
DES
```

6. RESPONDA A ESTA SECÇÃO APENAS SE IMPLEMENTOU O SUB-PROTOCOLO STGC-SAP, de acordo com os requisitos do enunciado. Se não, passe ao ponto 7 (Conclusões)

6.1 Apresente (usando notação apropriada) a especificação (o mais completa possível) das mensagens trocadas no contexto do processamento do subprotocolo STGC/SAP:

Ronda 1: Client > AS: Formato da mensagem com os componentes criptográficos e sua descrição:

Pedido = Cliente ID || NonceC || IPMC || AutenticadorC

Cliente ID = username/email do cliente necessário para o servidor ir buscar a sua informação ao ficheiro dos utilizadores.
NounceC = número único que identifica a mensagem.
IPMC = endereço Multicast a que o cliente se quer juntar.
AutenticadorC = E [K, NonceC || IPMC || MD(password) || MACKm (X)] significa que o autenticador é cifrado por uma cifra baseada em palavra-passe e com a chave K.
K = MD(password) é a chave para cifrar o autenticador, onde MD é uma função de dispersão criptográfica como por exemplo o SHA-512 ou MD5.
Km = MD5(NonceC || MD(password)), onde MD5 é o Message-Digest algorithm 5 que digere o nounceC juntamente com o MD(password) para produzir a chave utilizada para realizar o MAC ao conteúdo X.
X = NonceC || IPMC || MD(password)
A palavra-passe é fornecida pelo utilizador no momento de autenticação perante o servidor AS.

Ronda 2: AS > Client: Formato da mensagem com os componentes criptográficos e sua descrição:

Resposta = E[*K*, NonceC+1 || NonceS || TicketAS || MACKm (X)]

A resposta do servidor AS ao cliente é também ela cifrada por uma cifra baseada em palavra-passe e com chave K.
K = MD(password) || NonceC+1, MD é uma função de dispersão criptográfica, password é a palavra-passe do cliente em questão.
NounceC+1 = um número obtido pela soma de 1 com o número único do pedido.
NounceS = número único gerado pelo servidor para identificar a mensagem de resposta.
TicketAS = estrutura de dados que contém toda a informação necessária para o utilizador entrar numa sessão segura multicast. Esta estrutura contém o id do cliente, ip multicast ao qual dá acesso, ciphersuite para indicar o algoritmo de cifra utilizado para enviar/receber mensagens, sessionKey para cifrar/decifrar as mensagens enviadas/recebidas, macAlgorithm e macKey para produzir MACs,

timeout para indicar o tempo de vida do ticket.
Km = MD5(NonceC+1 || MD(password)), onde MD5 é o Message-Digest algorithm 5 que digere o nounceC+1 juntamente com o MD(password) para produzir a chave utilizada para realizar o MAC ao conteúdo X.
X = NonceC+1|| NonceS || TicketAS é o conteúdo digerido pelo MAC para gerar o código de autenticação da mensagem.

6.2 O servidor AS possui configurações com os seguintes ficheiros, conforme a especificação do enunciado:

Ficheiro de configuração	SIM	NÃO
ciphersuite.conf //gestão de ciphersuites utilizáveis para as sessões	X	
keystore.jceks //chaves (criptográficas simétricas ou para MACs – HMACs ou CMACs)	X	
users.conf //Utilizadores registados que podem participar em grupos multicast seguros STGC	X	
dacl.conf //configuração de listas de controlo de acesso (DAC) de utilizadores que podem participar em cada grupo multicast definido como grupo seguro STGC		X
stgcsap.conf //configuração criptográfica para possíveis construções PBEncryption e MACs para o protocolo STGC-SAP	X	

6.3 A minha implementação do protocolo STGC-SAP pode ser configurável no ficheiro stgcsap.conf, tendo sido verificado experimentalmente com configurações envolvendo:

PBE (Password-Based Encryption)	SIM	NÃO
PBEWithSHAAnd3KeyTripleDES	X	
PBEWITHSHA256AND256BITAES-CBC-BC	X	
PBEWITHSHA-1AND256BITAES-CBC-BC	X	
PBEWithHmacSHA224AndAES_256		X
OUTRA(S) QUAIS:		
MACS (HMACS)	**SIM**	**NÃO**

HMacSHA1	X	
HMAC/SHA384	X	
HMAC-SHA3-224	X	
HMAC-SHA3-256	X	
HMAC-SHA512	X	
OUTRA(S) QUAIS:		
MACS (CMACS)	**SIM**	**NÃO**
SKIPJACKMAC	X	
AESGMAC		X
RC6GMAC		X
RC5MA		X
DES	X	
OUTRA(S) QUAIS:		

6.4 Indique em que consiste o formato de um TicketAS (devolvido na ronda 2 do subprotocolo STGC-SAP). Pode copiar a estrutura de dados que o descreve:

```
package stgc;

import java.io.Serializable;
import java.security.Key;


final class TicketAS implements Serializable {

    private static final long serialVersionUID = 6146667052618121076L;

    private String client;
    private String ip;
    private String ciphersuite;
    private Key sessionKey;
    private String macAlgorithm;
    private Key macKey;
    private long timeout;
```

```
public TicketAS(String client, String ip, String ciphersuite, Key sessionKey, String macAlgorithm, Key macKey) {
	this.client = client;
	this.ip = ip;
	this.ciphersuite = ciphersuite;
	this.sessionKey = sessionKey;
	this.macAlgorithm = macAlgorithm;
	this.macKey = macKey;
	this.timeout = System.currentTimeMillis() + 3600000L; // up to 1 hour of authorization
}

public String getClient() {
	return client;
}

public void setClient(String client) {
	this.client = client;
}

public String getIp() {
	return ip;
}

public void setIp(String ip) {
	this.ip = ip;
}

public String getCiphersuite() {
	return ciphersuite;
}

public void setCiphersuite(String ciphersuite) {
	this.ciphersuite = ciphersuite;
}

public Key getSessionKey() {
	return sessionKey;
}

public void setSessionKey(Key sessionKey) {
	this.sessionKey = sessionKey;
}

public String getMac() {
	return macAlgorithm;
}
```

```
    public void setMacAlgorithm(String macAlgorithm) {
        this.macAlgorithm = macAlgorithm;
    }

    public Key getMacKey() {
        return macKey;
    }

    public void setMacKey(Key macKey) {
        this.macKey = macKey;
    }

    @Override
    public String toString() {
        return "TicketAS [client=" + client + ", ip=" + ip + ", ciphersuite=" + ciphersuite + ",
    sessionKey="
                + sessionKey + ", macAlgorithm=" + macAlgorithm + ", macKey=" + macKey
    + "]";
    }

    public boolean isExpired() {
        return System.currentTimeMillis() >= timeout;
    }

}
```

7. Conclusões e aspectos complementares

Inclua as conclusões sobre o seu desenvolvimento do TP1, podendo realçar aspectos complementares ou diferenciados da sua implementação. Se achar relevante pode argumentar sobre aspectos qualitativos que considera valorizáveis.

7.1 Conclusões resumidas:

O facto de a implementação ter sido feita numa classe isolada (STGCMulticastSocket) permitiu fazer alterações mínimas às aplicações de teste, nomeadamente MulticastSender, MulticastReciever e MulticastChat. Isto permite ter a encriptação/decifração toda escondida das aplicações do utilizador e permite que outras aplicações consigam incluir mensagens UDP por multicast seguro apenas trocando a implementação dos sockets multicast do Java pela implementação dos sockets multicast seguros que foi feita neste projeto.

7.2 Aspectos complementares a salientar:

O template do relatório não referiu a encriptação do ficheiro users.conf, no meu caso o ficheiro foi cifrado com PBEWithSHAAnd3KeyTripleDES utilizando a password b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86, o salt ec6ff495fd2f79b3 e o número de iterações 1060. Estes são argumentos do servidor que os utiliza para decifrar o ficheiro users.axx e obter a informação sobre os utilizadores.
Exemplo de execução do servidor: Java AuthenticationServer 224.224.224.224 3001 b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86 ec6ff495fd2f79b3 1060

7.3 Argumentação sobre fatores diferenciados e qualitativos implementados no TP1

Os ficheiros de configuração podem ser separados por salas o que permite haver diferentes configurações para diferentes salas. Isto só foi possível porque as configurações foram alteradas de Java Properties para XML.
Foi incluído um programa FileEncryption que permite cifrar um ficheiro. Este programa foi executado para transformar o ficheiro não cifrado users.conf no ficheiro cifrado users.axx.