SEGURIDAD EN EL DISEÑO DE SOFTWARE

### GOCHA UN CHAT SEGURO

#### 

#### MICHEL PADRÓN MORALES

#### Universidad de Alicante

### Índice

Introducción ---------------------------------------------------------- 1

Objetivos ----------------------------------------------------------- 2

Modulo de AUTH –-------------------------------------------------- 3

Módulo de Chat –---------------------------------------------------- 5

Configuración -------------------------------------------------------- 6

Conclusiones -------------------------------------------------------- 6

### Introducción

### 

La asignatura de seguridad en el diseño de software ha despertado mis deseos de conocimientos acerca del inmenso mundo de la seguridad informática. Gracias a los conocimientos adquiridos en la misma he desarrollado GOCHA.

GOCHA es un chat seguro implementado en lenguaje GO.

La seguridad nunca será garantizada al 100% pero sí podemos proteger nuestro sistema lo suficiente como para que muchos posibles atacantes pierdan el interés por el mismo.

### Objetivos

Los objetivos que se han marcado para la primera versión de GOCHA son los siguientes:

1 - Crear un módulo de registro y autenticación de usuarios

2 - Utilizar TLS sobre TCP en la autenticación aportando una capa más de seguridad sobre las conexiones

3 - Crear el módulo de chat con las siguientes funcionalidades:

* Conectarse a al lobby
* Desconectarse del lobby
* Ignorar a un usuario
* Entrar a una sala privada
* Salir de una sala privada
* Enviar mensaje
* Recibir mensaje

### Módulo de Autenticación

1- Modulo de LOGIN/REGISTRO

Para tener una fuerte seguridad el usuario se conectará al servidor mediante TLS.

El servidor expone un puerto para la recepción de clientes y dispone de un manejador de peticiones

Servidor.

Al conectarse se le ofrecerá un panel tipo MENU con las funcionalidades:

Cliente

**1 - Entrar al Lobby**

Esta opción te permite acceder a la sala principal del chat. Primero deberás loguearte.

* En el cliente
* Se pide el usuario y la contraseña
* Internamente se realiza un hash con SHA512 de la contraseña
* Se utilizan los primeros 256 bits para el login y la otra mitad para datos
* Enviamos por TLS la petición de login

* En servidor
* El menejador espera el comando “login”
* Al recibir la petición de login se extrae de la estructura enviada el usuario, la contraseña.
* Se hace un hash de la contraseña con una sal generada previamente y se comprueba si el usuario existe
* Se realiza la comparación de hashes
* Se envía una respuesta

**2 - Nuevo Usuario**

* En el cliente
* Se pide el usuario y la contraseña
* Internamente se realiza un hash con SHA512 de la contraseña
* Se utilizan los primeros 256 bits para el login y la otra mitad para datos
* Se genera un par de claves (privada, pública) para el servidor
* Se codifican en JSON las claves generadas
* Se extrae la clave publica por separado y se codifica en JSON
* Se crea un paquete con la estructura comando, usuario, password, clave publica y clave privada la cual se encripta previamente con un cifrador en bloque (AES)
* Enviamos por TLS la petición de creación de usuarios
* En el servidor
* El menejador espera el comando “register”
* Al recibir la petición de registro se extrae de la estructura enviada el usuario, la contraseña, la clave publica y la privada.
* Se hace un hash de la contraseña con una sal generada previamente y se comprueba si el usuario ha sido creado previamente.
* Se guardan los datos del usuario y se envía una confirmación

### Módulo de Chat

Al momento de loguearse el usuario de forma segura por TLS, se le permite una conexión mediante tcp al servicio de chat.

**Todos los mensajes enviados a partir de este momento por el usuario seran encriptados con el Key generado por el Sha512 del password. Luego el servidor realizara una desencriptacion utilizando el Key almacenado en la fase de autenticación por TLS.**

Con esto logramos que los mensajes viajen cifrados. Luego todos los mensajes van doblemente codificados.

Entramos al modo chat, existen varias funcionalidades que el servidor organiza con un manejador:

/entrar “xxx” - Te permite acceder a una sala privada con el nombre “xxx”

/ignorar “user” - Si no te interesa saber que escribe el usuario “user”

/desconectar - Salir de la aplicación

/dejar “xxx” - Abandonas la sala “xxx”

### Configuracion

Arrancar el servidor desde la raiz del proyecto:

go run server.go

Arrancar tantos clientes como se necesite desde la raiz del proyecto:

go run cliente.go

Tanto cliente como servidor toman su configuración desde config.go

**Extras:**

Se ha añadido un fichero para logs en el servidor, con lo que podemos almacenar todas las acciones generadas por los usuarios y revisar anomalías.

### Conclusiones

Se han cumplido los objetivos satisfactoriamente. Se ha tratado la seguridad de forma especial, aunque no se puede garantizar que el sistema no sea atacado de forma satisfactoria.

Hemos introducido un archivo de monitoreo de acciones de los usuarios que nos puede servir para detectar anomalías.

Con estas funcionalidades versionamos la aplicación GOCHAT v1.0.0