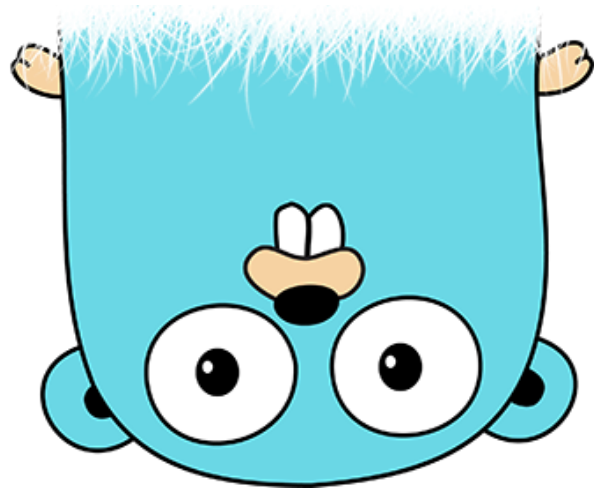
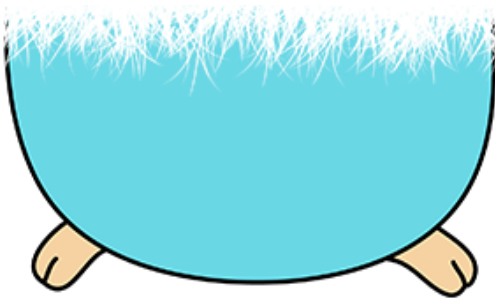


Gestor de Contraseñas

26 de mayo del 2019



Autores

- Pedro Giménez Aldeguez
- Melanie Mariam Cruz Morgado

Contenido

Introducción	2
Descripción	2
Despliegue	3
Resultados	4
Aspectos a mejorar	6
Conclusiones	6

Introducción

Este documento es una memoria sobre la aplicación de las prácticas de la asignatura de Seguridad del Diseño del Software de este curso 2018/2019 en que se incluyen las decisiones tomadas, las funcionalidades implementadas, propuestas de mejora y unas conclusiones a nivel personal de ambos integrantes de la pareja de desarrollo de la práctica, respecto a varios temas de esta.

Descripción


En esta práctica hemos desarrollado un sistema de gestión de contraseñas con arquitectura cliente/servidor y otras características que serán explicadas a continuación.

En general la aplicación es utilizada para registrar datos personales como notas, contraseñas y número de tarjetas bancarias, poder consultarlas en cualquier momento mediante el identificador o visualizar todas las guardadas o borrarlas.

El mecanismo de autenticación es seguro, ya que, el cliente, tras escribir la contraseña realiza un hash con *SHA 512*, a continuación coge los primeros 256 bits de dicho hash, por último, lo envía al servidor para utilizarlos como clave de autenticación. Esos bits enviados al servidor más el añadido de la “sal” son utilizados en una función de derivación de clave (*Scrypt*) para aumentar la seguridad.

El transporte de red entre el cliente y el servidor es seguro gracias a que hemos utilizado *Https* para el envío de datos. Para esto ha sido necesario generar un certificado no firmado con *Openssl* y, añadiéndolo a la conexión, hemos conseguido que esta sea segura y privada.

El conocimiento cero también lo hemos implementado, es decir, el servidor no puede leer ninguno de los datos guardados por el usuario desde el cliente, lo hemos implementado de forma de que dichos datos pasen ya cifrados al servidor, quien simplemente hace la función de almacenaje seguro y ante cualquier petición devuelve el archivo completo de dicho usuario, encargándose así el cliente de hacer toda la gestión, volver a cifrar los datos y mandárselo.



Además, guardamos los datos que almacena el cliente cifrado con la mitad de su contraseña y los datos de registro de los usuarios con una contraseña dada al servidor en archivos *json*.

Cabe destacar que en todo momento, en nuestra aplicación, hemos usado AES para el cifrado.

Aunque sea un gestor de contraseñas, hemos añadido otros datos adicionales que puede guardar el usuario, en específico, notas y números de tarjetas de crédito.

También tenemos la opción de generación de contraseñas aleatorias escogiendo la opción que queremos ya que nos pedirá que escojamos entre ocho opciones, como sólo minúsculas, sólo número, cualquier carácter, pronunciable, etc.

Respecto al diseño, no hemos realizado interfaz gráfica ya que hemos priorizado la correcta implementación de las funcionalidades, pero hemos realizado unos menús muy claros con las diferentes opciones propuestas en cada caso.

Despliegue

Debemos abrir dos terminales: una colocada en la carpeta del servidor y otra en la del cliente. En la primera ejecutamos ***go run *.go*** y en la segunda tenemos dos opciones y de eso dependerá qué comando ejecutar:

1. Registrarse: ***go run *.go signup***
2. Iniciar sesión: ***go run *.go signin***

A partir de este momento, es tan sencillo como seguir las indicaciones que aparecen en la terminal.

Resultados

Contamos con dos partes: cliente y servidor, explicaremos ambas por separado.

Comenzaremos por el servidor que tiene un funcionamiento más sencillo. El servidor contiene un archivo inicial llamado *“server.go”* que se encarga de esperar las órdenes del cliente y comparar si la contraseña que le pasan para iniciarse coincide con el cifrado de *“registro.json”*. Cuando el cliente le envía una orden, dependiendo de cual sea, pasará al archivo correspondiente y a su funcionamiento esperado.

Ahora pasamos al cliente, que por el hecho de usar conocimiento cero, es donde se realiza la mayor parte de las funcionalidades de la aplicación. Para empezar, tenemos un archivo base, donde se encuentran todas las funciones y estructuras (*“func.go”*) que se van a ir utilizando durante todo el proceso. El archivo *“incio.go”* se encarga de gestionar las diferentes órdenes que desea el cliente a los diferentes archivos que realizan las operaciones necesarias y su respectiva llamada al servidor para que lo lea o almacene.

La estructura utilizada para almacenar los datos de los registros de usuarios es la siguiente.

```
type registry struct {  
    Key    []byte  
    Users map[string]user  
}
```

Esta estructura utiliza dentro otra estructura con los datos necesarios del usuario para almacenar los usuarios de una forma segura.

```
type user struct {  
    Name string // nombre de usuario  
    Hash []byte  // hash de la contraseña  
    Salt []byte  // sal para la contraseña  
    Data map[string]string // datos adicionales del usuario  
}
```

El cliente utiliza varias estructuras para la respuesta del servidor, una estructura con un File y otra sin ella.

```
// respuesta del servidor
type Respfix struct {
    Ok bool // true -> correcto, false -> error
    File string
    Msg string // mensaje adicional
}
```

```
// respuesta del servidor
type Resp struct {
    Ok bool // true -> correcto, false -> error
    Msg string // mensaje adicional
}
```

Además la aplicación utiliza distintas estructuras para almacenar los diferentes tipos de datos que los usuarios tienen la opción de guardar.

```
type Data struct {
    Contraseña map[string]contraseña
    Nota map[string]nota
    Tarjeta map[string]tarjeta
}

var gData map[string]Data
```

```
type nota struct {
    Titulo string
    Cuerpo string
}

var gNota map[string]nota
```

```
type contraseña struct {
    Url string
    Pwd string
}

var gContraseña map[string]contraseña
```

```
type tarjeta struct {
    Tipo string
    Titular string
    Numero int64
    CVV int64
    Fecha string
}

var gTarjeta map[string]tarjeta
```

Cuando el cliente inicia sesión, aparecerá el un menú en el terminal con diferentes opciones que ofrece nuestra aplicación para poder usar las diferentes funcionalidades de esta. Pudiendo añadir, editar, ver, buscar y borrar datos, y cerrar la aplicación.

Se espera que después de toda esta implementación, el funcionamiento del producto sea seguro, aunque tiene varias mejoras en este aspecto que no se han implementado por falta de conocimiento, pero en general es bastante seguro, según lo que se nos ha pedido.


Aspectos a mejorar

En el caso de haber tenido más tiempo, unos aspectos que se hubiesen podido implementar y estaban como extras propuestas en la práctica son: la creación de una interfaz gráfica, la compartición de contraseñas con grupos de usuarios y programar una extensión de Google Chrome que se comunicará con el servidor para buscar las contraseñas guardadas y se puedan usar fácilmente en este navegador.

Otro aspecto que mejoraríamos de nuestra aplicación es la visibilidad de lo que introduce el usuario, ya que habría que diferenciar entre: si lo que el usuario introduce en un texto o una opción, que sí que se debe ver en la pantalla; o si lo que introduce es una contraseña, que no se debe ver. Este control no sabíamos cómo realizarlo, por esta razón, en todos los casos se ve lo que el usuario introduce, pero creemos que es algo que se debería controlar y, por esta razón, lo nombramos en este apartado.

Conclusiones

El objetivo de la práctica nos ha parecido interesante incluso para uso propio, ya que cada vez tenemos cuentas en más sitios webs, aplicaciones, etc. y es una forma sencilla para guardarlas de forma segura y poder consultarlas en caso de olvido sin la necesidad de optar por la recuperación de contraseña, que obliga a cambiarla.



Además al aprendizaje, creemos que ha sido el correcto con aspectos de seguridad que tenían relación con conocimientos previos recibidos en Sistemas Distribuidos, pero en esta asignatura se explican de otra manera en la que, en nuestra opinión, hemos adquirido mejor estos conocimientos. Lo único que hemos echado en falta ha sido la reacción que hubiese que tener en caso de ataques, ya que es algo que puede pasarnos en el mercado laboral y no se nos prepara para ello.

Respecto el lenguaje utilizado, que era la primera vez que usábamos este pensamos que ha sido el correcto. Nos ha gustado trabajar con Go por su sencillez, su limpieza en el código y su rendimiento. Sí que es cierto que en un principio tuvimos dificultades para separar el código en varios archivos usando diferentes paquetes, por no saber el correcto uso, pero una vez aprendimos, no tenía mayor dificultad.

Por último, aunque esta práctica tenga la opción de realizarse individualmente e incluso en algunos casos lo han realizado 3 integrantes. En nuestra opinión, ha sido una idea acertada la de realizarla en pareja, ya que contando con más personas, no se hubiesen adquirido tanto los conocimientos que se quiere conseguir con la realización de esta, aunque se hubiesen podido implementar más funcionalidades. En estos casos en los que además el lenguaje era nuevo, creemos que es enriquecedor realizarlo con alguien más que siempre sirve de apoyo y lo que no ve uno, lo ve el otro y que, en definitiva, los proyectos informáticos en general no se realizan por un solo desarrollador y, en este caso, tal vez hubiese faltado tiempo o hubiese supuesto una sobrecarga demasiado grande de trabajo.