Evaluación de Aprendizaje n°1

Seguridad y Calidad en Aplicaciones Web

Alumno: Marcos German Cabral, DNI: 42.684.487

Profesores: Pomar Pablo. Ureta Walter.

Fecha de entrega: 12/10/2020

1. Dentro de las amenazas que forman parte del TopTen del OWASP la que mayor riesgo correría en dicho código sería la del ataque por Inyección. La misma es de fácil explotabilidad, y de un impacto técnico muy grave.

Podemos observar esta parte del código en las líneas “**String query = "select \* from usuarios where usuario='" + username + "' and password = '" + password + "'";**” Donde fácilmente se podrían alterar los parámetros con una consulta a la base de datos de tipo OR. De esta manera podría conocer registros de nuestra base de datos y hasta manipularla.

1. Hay distintas maneras y opciones para mejorar el código, en un proyecto quizás agregar todas o la mayoría sería algo costoso y el rendimiento se vería un poco afectado. En mi opinión podríamos utilizar un ORMS, o herramienta de Mapeo Relacional de Objetos tal como Hibernate, ya que el uso de Sql parametrizado nos dará mayor seguridad.

Otra alternativa seria tener una White list, o “Lista blanca” para validar entradas positivas a usuarios.

Y una opción más enfocada al código es la de prepared statement, es recomendado utilizarlo siempre que vayamos a trabajar con algún lenguaje que funcione con SQL, ya que no reciben valores concretos hasta que se ejecutan dentro del sistema.

Esto es gracias a que son plantillas para consultas a sistemas de bases de datos en lenguaje SQL cuyos parámetros esperan valores. Para reemplazar dichos valores, estas plantillas trabajan con “variables”, que no son sustituidos por los valores reales hasta estar dentro del sistema. A diferencia, cuando las consultas se introducen a mano los valores se asignan en el mismo momento de ejecutarlas.

1. La medida para proteger las contraseñas de los usuarios que dispone la aplicación es el encriptado con un Hash en Hexadecimal. Podemos verlo en el siguientes métodos:

**byte[] hash = digest.digest((password+"escf").getBytes(StandardCharsets.UTF\_8)); password = this.bytesToHex(hash);**

**private static String bytesToHex(byte[] hash) {**

**StringBuffer hexString = new StringBuffer();**

**for (int i = 0; i < hash.length; i++) {**

**String hex = Integer.toHexString(0xff & hash[i]);**

**if(hex.length() == 1) hexString.append('0');**

**hexString.append(hex);**

**}**

**return hexString.toString();**

**}**