Listado de errores incluidos

localhost/main.html

Input Validation and Representation

- Cross-Site Scripting. Sending unvalidated data to a Web browser can result in the browser executing malicious code (usually scripts).

  Usar el texto a continuación en la pizarra de pruebas para probar el punto de inseguridad.

```
<script>
 alert("hola");
</script>
```

- SQL Injection. Constructing a dynamic SQL statement with user input may allow an attacker to modify the statement's meaning or to execute arbitrary SQL commands

  http://www.journaldev.com/2489/jdbc-statement-vs-preparedstatement-sql-injection-example

```
public List <UserDTO> searchAll(UserDTO userDto) {
Connection connection = null;
PreparedStatement preparedStatement = null;
List<UserDTO> list = new ArrayList<>();
try {
 connection = conManager.open();
 Statement statement = connection.createStatement();
 String sql = "SELECT name FROM USER WHERE name = '" + userDto.getName() +
"'";
 logger.debug(sql);
 ResultSet resultSet = statement
   .executeQuery(sql);
 while (resultSet.next()) {
  UserDTO userDtoTemp = new UserDTO();
  userDtoTemp.setName(resultSet.getString("name"));
  list.add(userDtoTemp);
 }
 return list;
} catch (Exception e) {
 logger.error(e);
 throw new RuntimeException(e);
} finally {
 conManager.close(preparedStatement);
 conManager.close(connection);
}
```

  Probar con el siguiente texto para poder ver todos los usuarios de base de datos. Cuando realmente el usuario solo debería poder hacer consulta sobre uno conocido

```
sdfssd' or '1'='1
```

- Integer Overflow. Not accounting for integer overflow can result in logic errors or buffer overflows.

  Introduzca el valor numérico en la pizarra de pruebas 324235346346345645645654 para probar el error. Este error se produce porque no existe una validación de que el tipo de dato introducido por el usuario concuerde en tamaño con el tipo de dato introducido en la base de datos

```
org.h2.jdbc.JdbcSQLException: Valor numerico fuera de rango:
"324235346346345645645654"
Numeric value out of range: "324235346346345645645654"; SQL statement:
```

- Log Forging. Writing unvalidated user input into log files can allow an attacker to forge log entries or inject malicious content into logs.

  Tanto en el fichero de log como en la salida por consola estandar con el artefacto malo se sacan mensajes sobre las contraseñas que se han intentado a lo largo de todas las pruebas de acceso

```
[DEBUG] 2017-01-05 16:09:39.418 [qtp1415887228-54] UserRepositoryImpl -
probando si es valido el usuario: user password: user
[DEBUG] 2017-01-05 16:09:39.419 [qtp1415887228-54] UserRepositoryImpl -
userResource: Resource "user.name.key"
[DEBUG] 2017-01-05 16:09:39.419 [qtp1415887228-54] UserRepositoryImpl -
passResource: Resource "user.pass.key"
[DEBUG] 2017-01-05 16:09:40.037 [qtp1415887228-54] UserRepositoryImpl -
Buscando el usaurio en la base de datos
[DEBUG] 2017-01-05 16:09:40.038 [qtp1415887228-54] UserRepositoryImpl -
es.gorka.edu.dto.UserDTO@88961e6[name=user,password=user]
[DEBUG] 2017-01-05 16:09:40.038 [qtp1415887228-54] UserRepositoryImpl - En la
base de datos existe el usuario
[DEBUG] 2017-01-05 16:09:40.038 [qtp1415887228-54] UserRepositoryImpl -
es.gorka.edu.dto.UserDTO@1ea93056[name=user,password=null]
```

API Abuse

- J2EE Bad Practices: getConnection(). The J2EE standard forbids the direct management of connections.

  La clase AbstractConnectionManager hace uso del método open() para abrir conexiones a la base de datos. Esta forma de conexión puede hacer que se creen conexiones zombie

```
public Connection open() {
 Connection conn = null;
 try {
  Class.forName(getClassDriver());
  conn = DriverManager.getConnection(getJdbcUrl(), getUser(), getPass());
 } catch (Exception e) {
  logger.error(e);
  throw new RuntimeException(e);
 }
 return conn;
}
```

- Often Misused: Exception Handling. A dangerous function can throw an exception, potentially causing the program to crash.

```
public void insert(UserDTO userDto) {

  Connection connection = null;
  PreparedStatement preparedStatement = null;

  User user = new User();
  asesembler.toEntity(userDto, user);
  try {
   connection = conManager.open();
   preparedStatement = connection.prepareStatement("INSERT INTO
user(name,password) " + "VALUES (?, ?)");
   preparedStatement.setString(1, user.getName());
   preparedStatement.setString(2, user.getPassword());
   preparedStatement.executeUpdate();
   conManager.close(preparedStatement);

  } catch (Exception e) {
   logger.error(e);
   throw new RuntimeException(e);
  }

}
```

  Si por lo que sea se falla al insertar el usuario, la conexión no se cierra y el hecho de dejar una conexión colgada impide a la aplicación volver a funcionar

- Often Misused: Privilege Management. Failure to adhere to the principle of least privilege amplifies the risk posed by other vulnerabilities.

  Poniendo la url http://localhost:8080/main.html en el navegador hacemos un bypass de la aplicación sin necesidad de pasar por la pantalla previa de login

Security Features

- Missing Access Control. The program does not perform access control checks in a consistent manner across all potential execution paths.

  Poniendo la url http://localhost:8080/main.html en el navegador hacemos un bypass de la aplicación sin tener que pasar por la pantalla de login

- Password Management. Storing a password in plaintext may result in a system compromise. En el html HomePage.html si miramos su código fuente veremos

```
<input type="text" class="form-control" value="" name="p::name">
<!-- <input type="text" value="user"></input>  -->
<input type="text" class="form-control" value=""
   name="password">
<!-- <input type="password" value="user"></input>  -->
```

- Password Management: Password in Config File. Storing a password in a configuration file may result in system compromise.

  La contraseña se encuentra dentro del código fuente en el fichero wicket-package.properties

```
user.name.key=user
user.pass.key=password
```

- Password Management: Hard-Coded Password. Hard coded passwords may compromise system security in a way that cannot be easily remedied.

```
final String Credential = "user";
valid = (Credential.equals(userDto.getName()) &&
Credential.equals(userDto.getPassword()));
```

- Password Management: Weak Cryptography. Obscuring a password with a trivial encoding does not protect the password.

  La contraseña se guarda en la base de datos sin cifrar

Time and State

- J2EE Bad Practices: System.exit(). A Web application should not attempt to shut down its container.

  se ha incorporado a la aplicación un enlace que realizar la acción de system.exit

```
public interface CloseAble {

 default void close() {
  System.exit(0);
 }
}
```

- J2EE Bad Practices: Threads. Thread management in a Web application is forbidden in some circumstances and is always highly error prone.

  Para probar el error de manejar clases, entrar a http://localhost:8080/thread.html gestionado por la clase ThreadPage.java, wicket impide añadir componentes a su hilo principal a través de hilos diferentes

Errors

- Catch NullPointerException. Catching NullPointerException should not be used as an alternative to programmatic checks to prevent dereferencing a null pointer. En la clase User repositoryImpl en el método existUser

```
try {
 logger.debug("userResource: " + userResource);
 logger.debug("passResource: " + passResource);
 valid = valid || (userResource.equals(userDto.getName()) &&
userResource.equals(userDto.getName()));
} catch (NullPointerException e) {
}
```

- Empty Catch Block. Ignoring exceptions and other error conditions may allow an attacker to induce unexpected behavior unnoticed.

  el método puede fallar sin notificar del problema, lo que puede crear un posible camino a un ataque mediante un fallo de seguridad

```
try {
 connection = conManager.open();
 preparedStatement = connection.prepareStatement("DELETE FROM snippet WHERE
username = ?");
 preparedStatement.setString(1, user.getUserName());
 preparedStatement.executeQuery();

} catch (Exception e) {
} finally {
 conManager.close(preparedStatement);
 conManager.close(connection);
}
```

- Overly-Broad Catch Block. Catching overly broad exceptions promotes complex error handling code that is more likely to contain security vulnerabilities. En la clase UserRepositoryImpl en el método update tenemos el ejemplo de capturar la excepcion de la manera más genérica, sin clasificar las actuaciones a realizar según el tipo de excepcion

```
try {
 connection = conManager.open();
 preparedStatement = connection.prepareStatement("UPDATE user SET password =
? WHERE name = ?");
 preparedStatement.setString(1, user.getPassword());
 preparedStatement.setString(2, user.getName());
 preparedStatement.executeQuery();

} catch (Exception e) {
 logger.error(e);
]
```

  En lo personal soy partidario de no usar checked exception nunca, a menos que esté contruyendo una api de interfaz http://stackoverflow.com/questions/4639560/checked-vs-unchecked-exception

Code Quality

- Leftover Debug Code. Debug code can create unintended entry points in an application.

```
[DEBUG] 2017-01-05 16:08:22.087 [qtp1415887228-55] UserRepositoryImpl -
probando si es valido el usuario: asdsadsadasd password: asdasdasdasd
```

Encapsulation

- Data Leaking Between Users. Data can "bleed" from one session to another through member variables of singleton objects, such as Servlets, and objects from a shared pool.

  El objeto userDTO es un singleton de spring debido al cual cuando el usuario desconecta, en los campos de login todavía continuan los campos que el usuario previo insertó

```
@SpringBean
UserDTO userDto;


@Component
public class UserDTO implements IEntityDTO {
```

Environment

- J2EE Misconfiguration: Missing Error Handling. A Web application must define a default error page for 404 errors, 500 errors and to catch java.lang.Throwable exceptions to prevent attackers from mining information from the application

  Poniendo una url como http://localhost:8080/noexiste.html entraremos en una página de error genérica que en este caso como es una página controlada por spring, no muestra información clave para entrar, pero por ejemplo no indica que esta web usa spring, así el atacante podría intentar atacar alguna vulnerabilidad de spring conocida.