DATABASE & SYSTEM SECURITY WORKBOOK

LAB-11

Q1. You are a database security consultant. You store some confidential information previously in your database using AES encryption.

First, create a html page with the username and secret message fields. Once the inputs are given and submit button is pressed, the javascript program in the background will encrypt the message using AES and check it with the data (stored in the previous lab) of the secret message in the database table 'dbusers2'. If both of them match then print "Secret Verified Successfully", else print "Invalid Secret".

Create a function to check if both the values in the table are matching.

NOTE – There can only be two values in the table. One, the values entered in the previous lab and two, the values entered in this lab.

AES Java Code –

```
---------------------------------------------------------------------
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AdvancedEncryption
{
   private byte[] key;

   private static final String ALGORITHM = "AES";

   public AdvancedEncryption(byte[] key)
```

```
{
    this.key = key;
}

/**
 * Encrypts the given plain text
 *
 * @param plainText The plain text to encrypt
 */
public byte[] encrypt(byte[] plainText) throws Exception
{
    SecretKeySpec secretKey = new SecretKeySpec(key, ALGORITHM);
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.ENCRYPT_MODE, secretKey);

    return cipher.doFinal(plainText);
}

/**
 * Decrypts the given byte array
 *
 * @param cipherText The data to decrypt
 */
public byte[] decrypt(byte[] cipherText) throws Exception
{
    SecretKeySpec secretKey = new SecretKeySpec(key, ALGORITHM);
    Cipher cipher = Cipher.getInstance(ALGORITHM);
    cipher.init(Cipher.DECRYPT_MODE, secretKey);

    return cipher.doFinal(cipherText);
}
```

}

Usage example:

```
byte[] encryptionKey = "MyKeyValue16char".getBytes(StandardCharsets.UTF_8);
byte[] plainText = "(MySecretMessage)".getBytes(StandardCharsets.UTF_8);
AdvancedEncryptionStandard advancedEncryptionStandard = new
AdvancedEncryptionStandard(
    encryptionKey);
byte[] cipherText = advancedEncryptionStandard.encrypt(plainText);
byte[] decryptedCipherText = advancedEncryptionStandard.decrypt(cipherText);
```

Sol)

1. Connect as system to sqlplus.
2. Switch on the Eclipse IDE.
   Go to 'File' → New → Other → Scroll down to 'Web' → Expand the folder and choose 'Dynamic Web Project'.
3. Set project name as something of your choice.
4. Choose 'Dynamic web module version' as 2.5.
5. Click Finish.
6. Then expand the project file in 'Project Explorer'. Expand 'WebContent' → WEB-INF → Right click on 'lib' → Build Path… → Configure Build Path… → Click on 'Classpath' → Click 'Add External JARs…' → Navigate to the 'ojdbc14.jar' and add it.
7. Create a package called 'p1' with the java AES code.
   Expand your project in Project Explorer → Java Resources → Right click on 'src' → New → Package → Type Name as 'p1'.
   Right click on 'p1' → New → Class → Name it AdvancedEncryption → Press Finish.
   You can use the AES functions in your jsp file by utilising the package.
8. Then right click on 'WebContent'. This is where you make your html and jsp files.
   Right click on 'WebContent' → New → HTML File.
   Give a filename and end it with '.html' extension. Press Finish.

secretdataenc.html

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="aesencryption.jsp" method="post">

Username <input type="text" name="uname">

<br>
My secret message <input type="text" name="secret">

<br>
<input type="submit">



</form>
</body>
</html>
```

Save the file.

9. Right click on 'WebContent' → New → JSP File.

   Give a filename and end it with '.jsp' extension. Press Finish.

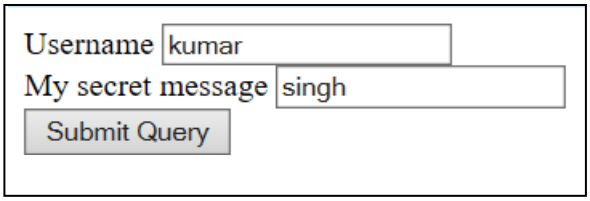
   aesencryption.jsp

```
<%@

page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ page  import="java.sql.*" %>
<%@page import="p1.*" %>
<%@page import="java.lang.Object" %>
<%@page import="java.nio.charset.StandardCharsets" %>



<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
```

```
<body>
<h1>  Welcome    </h1>

<%


String u=request.getParameter("uname");

String p=request.getParameter("secret");

 byte[] encryptionKey = "MyKeyValue16char".getBytes(StandardCharsets.UTF_8);
//give 16 chars as 16x8=128 bits

 byte[] plainText = p.getBytes(StandardCharsets.UTF_8);

 AdvancedEncryption aes = new AdvancedEncryption(encryptionKey);
 byte[] cipherText = aes.encrypt(plainText);



out.println(" Input:\n"+u +"---"+p);
try
{

        Class.forName("oracle.jdbc.driver.OracleDriver");


        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","sy
stem");

         PreparedStatement ps=con.prepareStatement("insert into dbusers2
values(?,?,?)");
         ps.setString(1,u);
         ps.setString(2,p);
         ps.setBytes(3,cipherText);
         System.out.println(cipherText);

int a=ps.executeUpdate();

out.println(a+ "  Record Inserted Successfully");
}
catch(Exception e)
{
      out.println(e);

}

%>

----------------------------
```

Save the file.

10. Right click on the html file and press 'Run As' → 1 Run on Server → (The Tomcat server should already be selected) → Press Finish.



11. Give your username and message.

The inputs I am giving are 'kumar' as username and 'singh' as message.

12. The hashed password must be stored in the table 'dbusers2' now.

Display the table using '**select * from dbusers2;**'.

```
SQL> select * from dbusers2;

USERNAME             MESSAGE              ENC
-------------------- -------------------- ----------------------------------
----------------------------------------------------------
kumar                singh                D276A3BD16CB6924A3B479752501B784
kumar                singh                D276A3BD16CB6924A3B479752501B784
```

13. Create a function to verify the table values in dbusers.

Type '**ed**' in sql command line. A notepad file with the '.buf' extension will open up.

Type your function in that file and press 'ALT+F+X'. Save the file when prompted.

CREATE OR REPLACE FUNCTION ValidateThis

return varchar2

as

cursor c1 is select * from dbusers2;

r1 dbusers2%ROWTYPE;

r2 dbusers2%ROWTYPE;

```
BEGIN
 open c1;
  fetch c1 into r1;
fetch c1 into r2;
if((r1.username = r2.username) AND (r1.message=r2.message))
THEN
return 'Secret Verified Successfully';
else
return 'Invalid Secret';
END IF;
 close c1;
END;
```

Type '**/**' (slash) and the function is created.

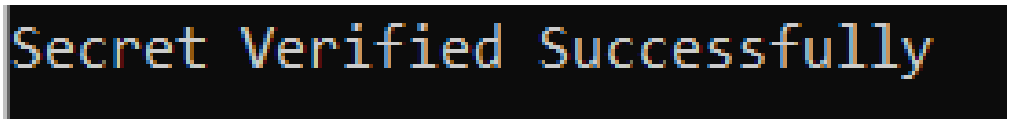
```
SQL> ed
Wrote file afiedt.buf

  1  CREATE OR REPLACE FUNCTION ValidateThis
  2      return varchar2
  3      as
  4         cursor c1 is select * from dbusers2;
  5      r1 dbusers2%ROWTYPE;
  6         r2 dbusers2%ROWTYPE;
  7      BEGIN
  8       open c1;
  9         fetch c1 into r1;
 10     fetch c1 into r2;
 11     if((r1.username = r2.username) AND (r1.message=r2.message))
 12     THEN
 13     return 'Secret Verified Successfully';
 14     else
 15     return 'Invalid Secret';
 16     END IF;
 17      close c1;
 18*   END;
SQL> /

Function created.
```

14. Now write a PL/SQL code to execute the function.

```
declare
   a varchar2(30);
   begin
   a:=ValidateThis();
   dbms_output.put_line(a);
  END;
```

Type '**/**' (slash).

```
SQL> ed
Wrote file afiedt.buf

  1  declare
  2      a varchar2(30);
  3      begin
  4      a:=ValidateThis();
  5      dbms_output.put_line(a);
  6*    END;
  7  /
Secret Verified Successfully

PL/SQL procedure successfully completed.
```



Post-Lab:

Q1.We want HR to execute the appsec security structures. We'd like to create a role to which we might grant execute on a package, and then grant that role to whomever needs it.

Sol)

Step1:

Create a role

CREATE ROLE hrview;

```
SQL> CREATE ROLE hrview;

Role created.
```

Step2:

Create appsec user

CREATE USER appsec1 IDENTIFIED BY password;

```
SQL> CREATE USER appsec1 IDENTIFIED BY password;

User created.
```

Step3:

Create a appsec package

```
CREATE OR REPLACE PACKAGE appsec1.app_sec_pkg IS

TYPE RESULTSET_TYPE IS REF CURSOR;

END;

/
```

```
SQL> CREATE OR REPLACE PACKAGE appsec1.app_sec_pkg IS
  2  TYPE RESULTSET_TYPE IS REF CURSOR;
  3  END;
  4  /

Package created.
```

Step4:

Grant the execute on package

GRANT EXECUTE ON appsec1.app_sec_pkg TO hr;

```
SQL> GRANT EXECUTE ON appsec1.app_sec_pkg TO hr;

Grant succeeded.
```

Q2. Write a query to grant access to HR Security Package, hr_sec_pkg, to a role that appusr has, the hrview_role

Sol)

Step1:

Create a HR Security package

```
CREATE OR REPLACE PACKAGE hr.hr_sec_pkg IS

TYPE RESULTSET_TYPE IS REF CURSOR;

END;

/
```

```
SQL> CREATE OR REPLACE PACKAGE hr.hr_sec_pkg IS
  2  TYPE RESULTSET_TYPE IS REF CURSOR;
  3  END;
  4  /

Package created.
```

Step2:

Grant execute on package

GRANT EXECUTE ON hr.hr_sec_pkg TO hr;

```
SQL> GRANT EXECUTE ON hr.hr_sec_pkg TO hr;

Grant succeeded.
```

Q3. Connect to Oracle database as appsec user, and set your role to the non-default role, appsec_role:

Sol)

Step1:

Connect the Oracle database as appsec user

```
SQL> CONNECT AS SYSDBA
Enter user-name: appsec
Enter password:
Connected.
```

Step2:

Create a role

CREATE ROLE appsec1_role;

```
SQL> CREATE ROLE appsec1_role;

Role created.
```

Step2:

Set the role

SET ROLE appsec1_role;

```
SQL> SET ROLE appsec1_role;

Role set.
```

Q4. Create the Application Security Error Log Table, t_appsec_errors

Sol)

Step1:

Create a user

CREATE USER appsec2 IDENTIFIED BY password;

```
SQL> CREATE USER appsec2 IDENTIFIED BY password;

User created.
```

Step2:

Create a table

```
CREATE TABLE appsec2.t_appsec_errors (
        err_no     NUMBER,
        err_txt    VARCHAR2(2000),
        msg_txt    VARCHAR2(4000) DEFAULT NULL,
         update_ts  DATE DEFAULT SYSDATE
);
```

```
SQL> CREATE TABLE appsec2.t_appsec_errors (     err_no     NUMBER,     err_txt    VARCHAR2(2000),     msg_txt    VARCHAR
2(4000) DEFAULT NULL,     update_ts  DATE DEFAULT SYSDATE );

Table created.
```

Q5. Create Index for the Application Security Error Log Table, t_appsec_errors

Sol)

```
CREATE INDEX i_appsec_errors00 ON appsec2.t_appsec_errors (
                update_ts
);
```

```
SQL> CREATE INDEX i_appsec_errors00 ON appsec2.t_appsec_errors (       update_ts );

Index created.
```