SQL injection

برای serialization از کتابخانه مخصوص جاوا java.io.serializable استفاده می کنیم و دو تا اطلاعات ذخیره میشوند، یکی در فایل معمولی برای back up و دیگری در SQL Table که دیتابیس اصلی برنامه است. اولا تمامی اطلاعات برنامه در کلاسی به نام Object ذخیره میکنیم

```
public class Object implements Serializable {

    public static Object object = new Object();
    public ArrayList<Boss> bosses = new ArrayList<>();
    private ArrayList<BuyLog> buyLogs = new ArrayList<>();
    private ArrayList<Category> categories = new ArrayList<>();
    private ArrayList<Comment> comments = new ArrayList<>();
    private ArrayList<Customer> customers = new ArrayList<>();
    private ArrayList<OffCode> offCodes = new ArrayList<>();
    private ArrayList<Point> points = new ArrayList<>();
    private ArrayList<Product> products = new ArrayList<>();
    private ArrayList<Sale> sales = new ArrayList<>();
    private ArrayList<Salesman> salesmen = new ArrayList<>();
    private ArrayList<SpecialOffCode> specialOffCodes = new ArrayList<>();
    private ArrayList<SellLog> sellLogs = new ArrayList<>();
    private ArrayList<Cart> carts = new ArrayList<>();
    private ArrayList<Request> requests = new ArrayList<>();
    private static final long serialVersionUID = 4L;

    public Object() {
        bosses.addAll(Storage.getAllBosses());
        buyLogs.addAll(Storage.allBuyLogs);
        categories.addAll(Storage.getAllCategories());
        comments.addAll(Storage.allComments);
        customers.addAll(Storage.getAllCustomers());
        offCodes.addAll(Storage.allOffCodes);
        points.addAll(Storage.allPoints);
        products.addAll(Storage.getAllProducts());
        sales.addAll(Storage.allSales);
        salesmen.addAll(Storage.getAllSalesmen());
        specialOffCodes.addAll(Storage.allSpecialOffCodes);
        sellLogs.addAll(Storage.allSellLogs);
        carts.addAll(Storage.allCarts);
        requests.addAll(Storage.getAllRequests());
    }
```

و سپس ان را سریال می کنیم و سپس encode و در databse ذخیره میکنیم و بنا بر این حتی با وصل شدن به سرور SQL و داشتن پسورد به دست اوردن اطلاعات مفید از آن غیر ممکن است چون که صرفا یک استرینگ رمز شده داریم که بدون الگوریتم بازگردانی اولا هیچ معنی ای ندارد و اگر هم الگوریتم بازگردانی هم داشته باشند چون سریال شده به صورت جیسون نیست که اطلاعات در آن معلوم باشد بلکه برای deserialization حتما باید که دقیقا عین خود کلاس رو داشته باشیم وگرنه deserialization رخ نخواهد داد ...

الگوریتم های رمزنگاری را در اینجا می توانیم ببینیم: (در حقیقت دیتا بیس ما فقط یک استرینگ کد شده است)

```
private String encode(String string) {
    String result = RandomString.getRandomString( n: 100) + string;
    result = base64Encoder.encodeToString(result.getBytes());
    result = ".xs,dsghjkdf,a3uih238ewajnksdhjf hsjkd jdzjfhgs " + result;
    result = base64Encoder.encodeToString(result.getBytes());
    result = new StringBuilder(result).reverse().toString();
    return result;
}

private String decode(String string) {
    string = new StringBuilder(string).reverse().toString();
    String result = new String(Base64.getDecoder().decode(string));
    result = result.substring(".xs,dsghjkdf,a3uih238ewajnksdhjf hsjkd jdzjfhgs ".length());
    result = new String(Base64.getDecoder().decode(result));
    result = result.substring(100);
    return result;
}
```

پس نتیجه می گیریم برای دسترسی به دیتا بیس از طریق SQL طریق زیر باید طی شود:

- داشتن رمز SQL برای Connect شدن به ان
- داشتن الگوریتم کامل encode , decode کردن (!!!)
- داشتن فیلد های کلاس Object به طور دقیق

و برای خواندن فایل هم همان فرایند خواهد بود بجز مرحله داشتن پسورد ها ولی باز هم موانع دوم و سوم پابرجا هستند.

و همان طور که گفته شد یک کلاس جدا به نام FileManager داریم که مسئول ذخیره سازی در فایل ولی این دیتابیس در برنامه استفاده نمی شود و صرفا به عنوان back up است که یعنی اگر به سرور SQL وصل نشدیم از فایل شروع به خواندن می کنیم

دو تابع مهم دیتابیس SQL:

```
public void insert(String data) {
    try {
        data = encode(data);
        PreparedStatement statement = connection.prepareStatement( s: "INSERT  INTO Neuer  (id,name) values (?,?)");
        statement.executeUpdate( s: "DELETE FROM Neuer");
        statement.setInt( i: 1,  i1: 1);
        statement.setString( i: 2, data);
        statement.executeUpdate();
        statement.close();
    } catch (SQLException s) {
        s.printStackTrace();
    }
}
```

*** تابع نوشتن serialized object در SQL table

```java
public byte[] show() throws SQLException {
    String string = "";
    String sql = "SELECT id, name FROM Neuer";
    Statement stmt = connection.createStatement();
    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        rs.getInt( s: "id");
        string = rs.getString( s: "name");
    }
    string = decode(string);
    rs.close();
    return Base64.getDecoder().decode(string);
}
```

*** تابع برداشتن اصلاعات از table

نمونه استرینگ رمز شده در دیتابیس:

rO0ABXNyABVDb250cm9sbGVyLlNRTC5PYmplY3QAAAAAAAAABAIADkwABmJvc3Nlc3QAFUxqYXZh
L3V0aWwvQXJyYXlMaXN0O0wAB2J1eUxvZ3NxAH4AAUwABWNhcnRzcQB+AAFMAApjYXRlZ29yaWV
zcQB+AAFMAAhjb21tZW50c3EAfgABTAAJY3VzdG9tZXJzcQB+AAFMAAhvZmZDb2Rlc3EAfgABTAAGc
G9pbnRzcQB+AAFMAAhwcm9kdWN0c3EAfgABTAAIcmVxdWVzdHNxAH4AAUwABXNhbGVzcQB+AAF
MAAhzYWxlc21lbnEAfgABTAAIc2VsbExvZ3NxAH4AAUwAD3NwZWNpYWxPZmZDb2Rlc3EAfgABeHBz
cgATamF2YS51dGlsLkFycmF5TGlzdHiB0h2Zx2GdAwABSQAEc2l6ZXhwAAAAAXcEAAAAAXNyABJNb
2RlbC5BY2NvdW50LkJvc3OQWV7N2p+AAwIAAUwACmZhdGhlckJvc3N0ABJMamF2YS9sYW5nL1N0c
mluZzt4cgAVTW9kZWwuQWNjb3VudC5BY2NvdW50A+/psGWbY0QCAApaAAhpc09ubGluZUwABUVtY
WlscQB+AAZMAAZhdmF0YXJxAH4ABkwACWZpcnN0TmFtZXEAfgAGTAAHaW1nUGF0aHEAfgAGTAA
IcGFzc3dvcmRxAH4ABkwABHJvbGV0ABRMTW9kZWwvQWNjb3VudC9Sb2xlO0wACnNlY29uZE5hbW
VxAH4ABkwACXRlbGVwaG9uZXEAfgAGTAAIdXNlcm5hbWVxAH4ABnhwAHQAGWFhYWFhYWFhYWW
FhYWFhYUBnbWFpbC5jb210AAdhdmF0YXIzdAAPYWFhYWFhYWFhYWFhYWFhcHQAD2FhYWFhYW
FhYWFhYWFhYX5yABJNb2RlbC5BY2NvdW50LlJvbGUAAAAAAAAABIAAHhyAA5qYXZhLmxhbmcuR
W51bQAAAAAAAAAEgAAeHB0AARCT1NTdAAPYWFhYWFhYWFhYWFhYWFhdAALMDkzMzM4MD
UyODh0AA9hYWFhYWFhYWFhYWFhYWFweHNxAH4AAwAAAAB3BAAAAAB4c3EAfgADAAAAAHcEA
AAAAHhzcQB+AAMAAAAAdwQAAAAAeHNxAH4AAwAAAAB3BAAAAAB4c3EAfgADAAAAAHcEAAAAA
HhzcQB+AAMAAAAAdwQAAAAAeHNxAH4AAwAAAAB3BAAAAAB4c3EAfgADAAAAAHcEAAAAAHhzc
QB+AAMAAAAAdwQAAAAAeHNxAH4AAwAAAAB3BAAAAAB4c3EAfgADAAAAAHcEAAAAAHhzcQB+A
AMAAAAAdwQAAAAAeHNxAH4AAwAAAAB3BAAAAAB4

توضیح مختصری درمورد ذخیره سازی در فایل: (دیتا بیس فرعی)

همان طور که گفته شد در حین ران شدن سرور اطلاعات همواره و به طور پیوسته در دیتابیس ها در حال اپدیت شدن هستن (یعنی حتی با قطعی نگهانی سرور هم مشکلی پیش نمیاد) و هر دو دیتا بیس (در صورت وصل شدن صحیح به SQL البته) به روز رسانی می شوند اما اگر SQL کار نکند با توجه به تابع زیر که ابتدا صدا زده می شود بجای استفاده از دیتابیس اصلی از فایل استفاده خواهیم کرد: (پس نتیجتا چون فایل همواره به روز رسانی می شود هیچ گاه مشکلی پیش نخواهد آمد)

```
public void startProgramme() {
    try {
        Object.deserialize(show());
        return;
    } catch (Exception exception) {
        System.out.println(exception.getMessage());
    }
    try {
        Object.deserialize(fileManager.readFromFile());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

ساختار کلی فایل منیجر هم به شکل زیر است که صرفا استرینگ کد شده را در فایل می نویسد و درصورت لزوم read میکند.

```
class FileManager {

    public FileManager() {
        File file = new File( pathname: "database.Neuer");
        try {
            file.createNewFile();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    public void updateFile(String string) throws IOException {
        FileWriter myWriter = new FileWriter( fileName: "database.Neuer");
        myWriter.write(SQL.encode(string));
        myWriter.close();
    }

    public byte[] readFromFile() throws IOException {
        BufferedReader bufferedReader = new BufferedReader(
                new FileReader(new File( pathname: "database.Neuer")));
        String s = bufferedReader.readLine();
        s = SQL.decode(s);
        return Base64.getDecoder().decode(s);
    }

}
```