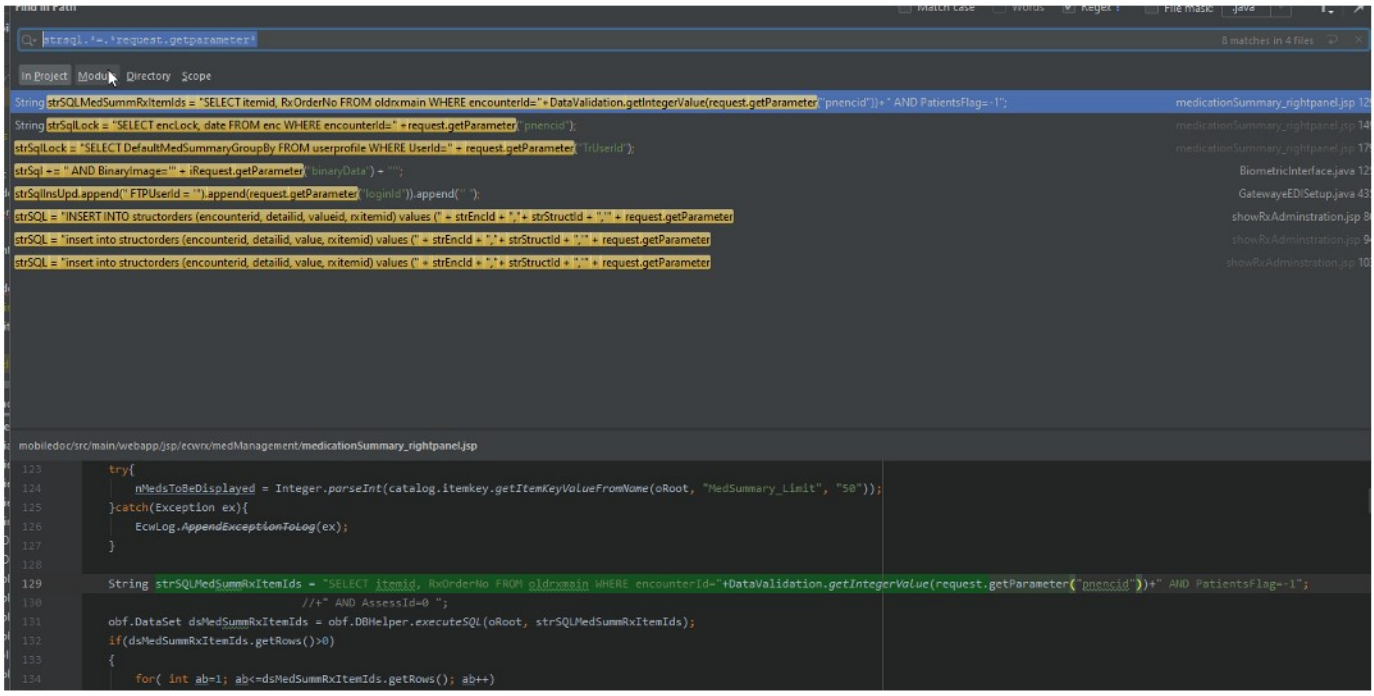
CodeReview 08/19/2020

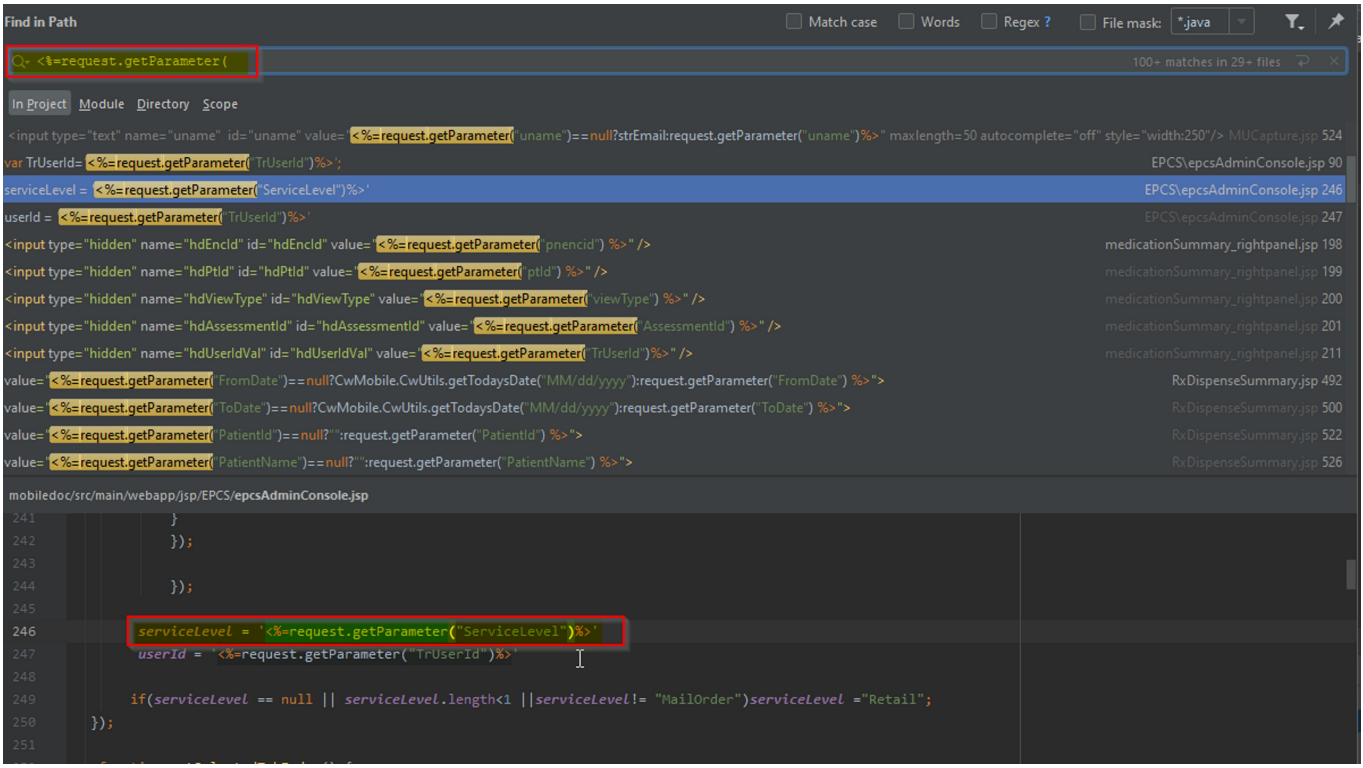
August 19, 2020 1:11 PM

1. **SQL Injection -**

- strsql.*=.*request.getparameter*
- Concatination of strings to a query. ---> sql.*=*+"



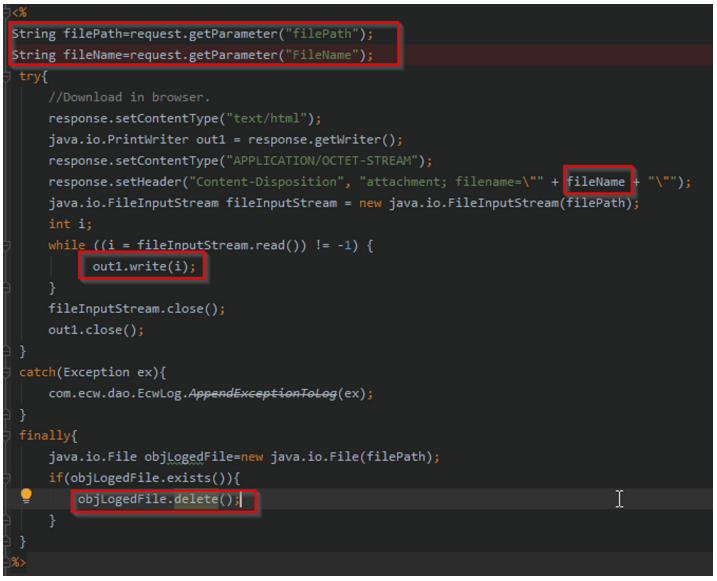
2. **XSS - CrossSite Scripting :**
   - <%=request.getParameter(
   - <%=.*request.getParameter
   - .InnterHtml
   - .eval(



3. Path Traversal

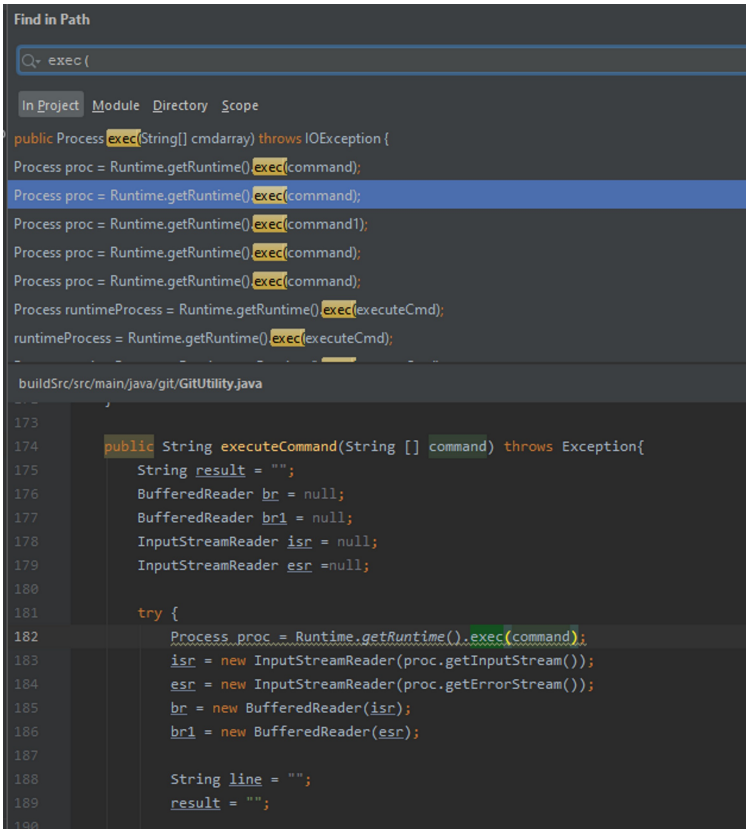
- java.io.FileInputStream fileInputStream=new java.io.FileInputStream(strLogDirName + strFileName);
- File.delete()
- fileName
- path

- http://localhost:8080/mobiledoc/jsp/webemr/webpm/visionplan/downloadRefundReport.jsp?filePath=C:\eClinicalWorks\tomcat\test.txt&FileName=test.txt



4. Command injection

- Runtime.getRuntime().exec(command);



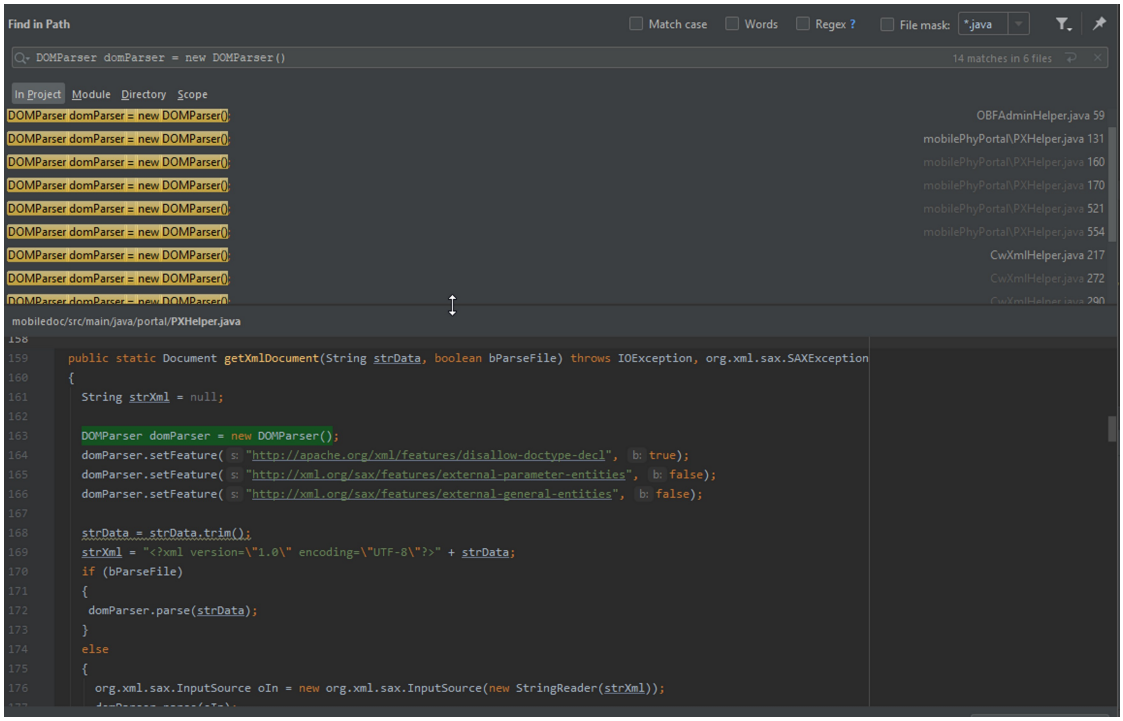
5. XXE:

pattern:

.parse(xml)

XML Parsers in Java:

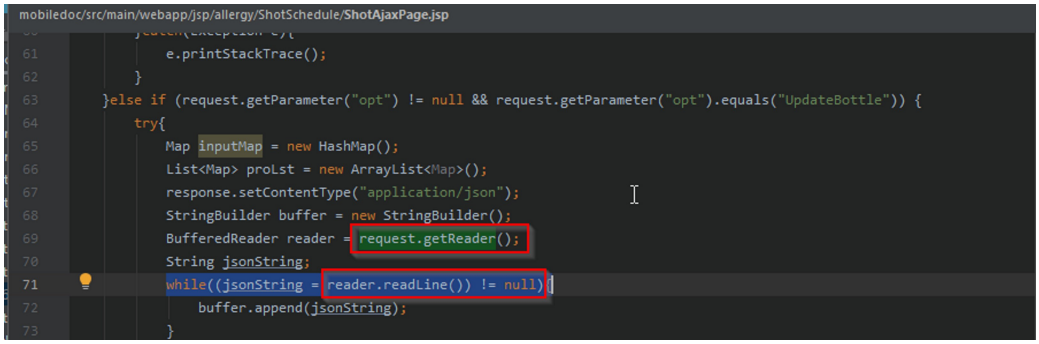
- javax.xml.validation.Validator
- SchemaFactory
- SAXTransformerFactory
- XMLReader
- SAXReader
- JAXB Unmarshaller



Reference : https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html

6. ReadLine Vulnerability :

- request.getReader
- readLine()
- while((jsonString = reader.readLine()) != null){
  - http://10.211.35.77:8123/mobiledoc/jsp/allergy/ShotSchedule/ShotAjaxPage.jsp?opt=SaveBottle



8. Session test on the product.

- Collect all the project routes(URLs)
- Collect all the servlets
- Hit all the urls without authenticated session,Using intruder
- Observe the responses 200,500
- identify urls supposed to work w/o authenticated session. Audit them.

9. Header Injection

response.setHeader("Content-Disposition", "attachment; filename=\"" + request.getParameter("strZipFileName") + "\"");

10. Insecure Deserialization

r00

Be aware of the following Java API uses for potential serialization vulnerability.

1. `XMLdecoder` with external user defined parameters
2. `XStream` with `fromXML` method
   (xstream version <= v1.46 is vulnerable to the serialization issue)
3. `ObjectInputStream` with `readObject`
4. Uses of `readObject`, `readObjectNodData`, `readResolve` or `readExternal`
5. `ObjectInputStream.readUnshared`
6. `Serializable`

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html

11. Out of band communication:

- HttpURLConnection conn = (HttpURLConnection) url.openConnection();
- https://
- http://
- ftp://
-

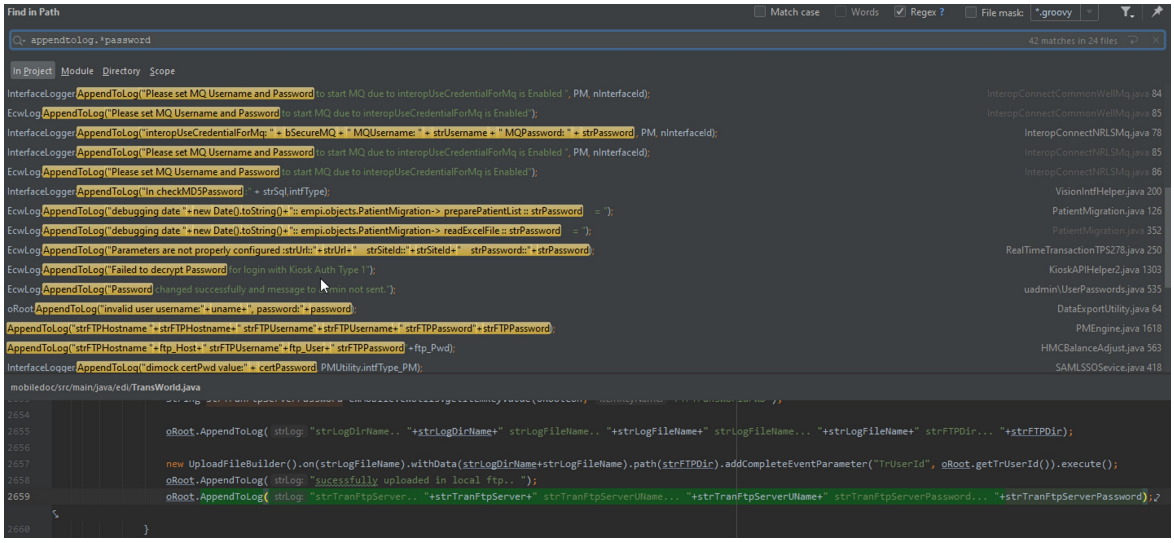
12. Sensitive information in queryString :

- +password+
- &password=
- &SSN
- Sessionid
- token
- PHI
- PII

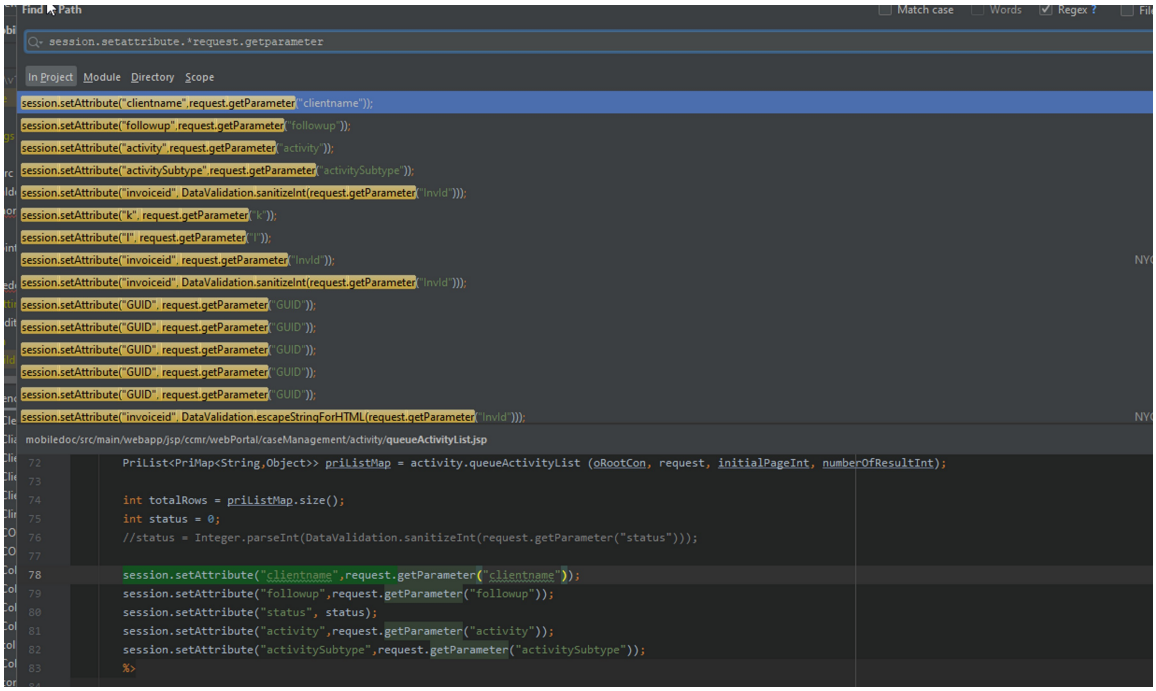
13. Vulnerable Cryptography Algorithms used

- MD5
- MD4
- SHA-1

14. **Sensitive information in logs:**
    - **username**
    - **password**
    - **token**
    - **Phi**
    - **Pii**
    - **Sessionid**



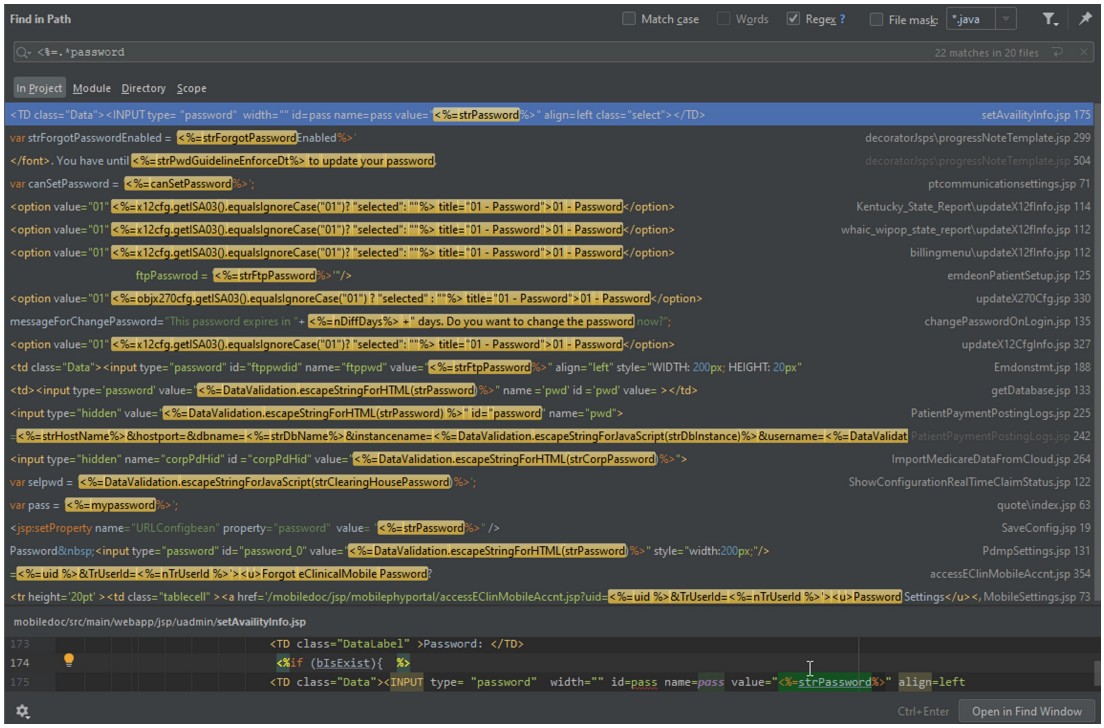
15. **Trust Boundary Violation :**



16. **Sensitive Data Disclosure in Response:**

    - **<%=.*password**
    - **<%=.*session**

```
try
{
    catalog.SafeMed oSM = new catalog.SafeMed();
    strXML = oSM.GetProblemList(strData);
}
catch( Exception ex )
{
  strXML = CwXmlHelper.getErrorXml(ex.getMessage());
}
%><%=strXML%><%@page import="catalog.CwXmlHelper"%>
```



sendRedirect(Request.getparam(url))
Req.forward(Request.getparam(url))

Response.Set-header(Request.getparam(url), 302)