

# Java 代码审计常用漏洞总结

## 1.1 审计方法总结

主要代码审计方法是跟踪用户输入数据和敏感函数参数回溯：

跟踪用户的输入数据，判断数据进入的每一个代码逻辑是否有可利用的点，此处的代码逻辑可以是一个函数，或者是条小小的条件判断语句。

敏感函数参数回溯，根据敏感函数，逆向追踪参数传递的过程。这个方法是最高效，最常用的方法。大多数漏洞的产生是因为函数的使用不当导致的，只要找到这些函数，就能够快速挖掘想要的漏洞。

以下是基于关键词审计技巧总结：

在搜索时要注意是否为整个单词,以及小写敏感这些设置

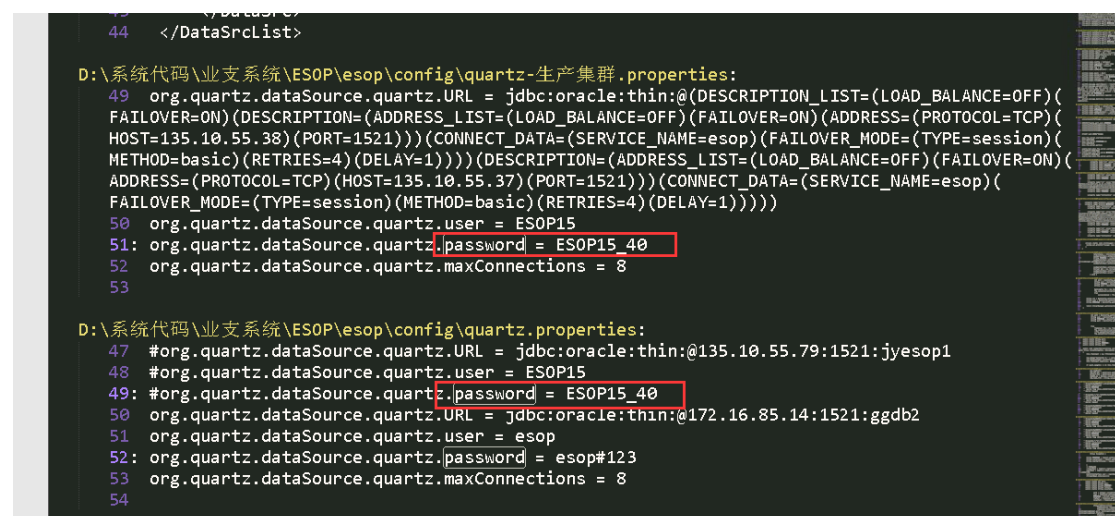
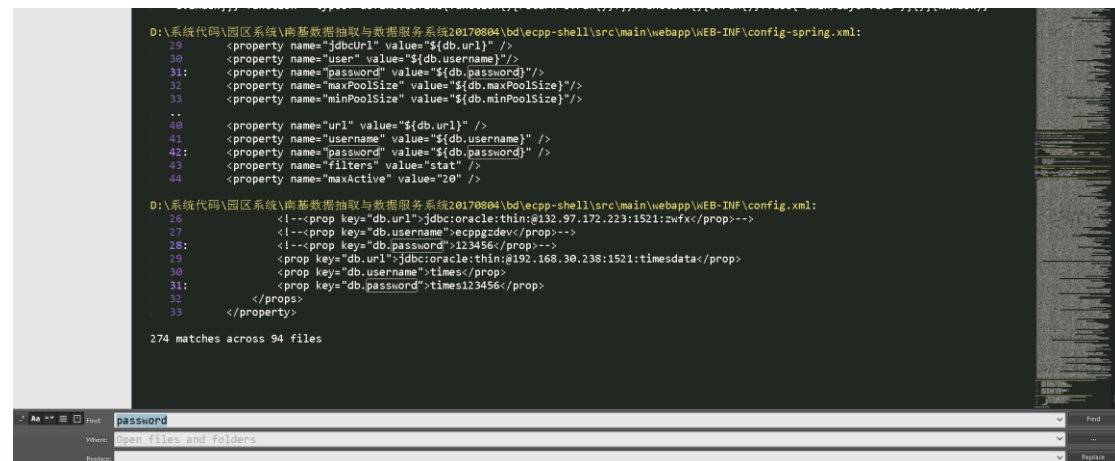
漏洞名称	关键词
密码硬编码、密码明文存储	password、pass、jdbc
XSS	getParamter、<%=、param.
SQL 注入	Select、Dao 、from 、delete 、update、insert
任意文件下载	download 、fileName 、filePath、write、getFile、getWriter
任意文件删除	Delete、deleteFile、fileName 、filePath
文件上传	Upload、write、fileName 、filePath
命令注入	getRuntime、exec、cmd、shell
缓冲区溢出	strcpy, strcat, scanf, memcpy, memmove, memecpp Getc(), fgetc(), getchar; read, printf
XML 注入	DocumentBuilder、XMLStreamReader、SAXBuilder、SAXParser SAXReader 、XMLReader SAXSource 、TransformerFactory 、SAXTransformerFactory 、 SchemaFactory
反序列化漏洞	ObjectInputStream.readObject 、ObjectInputStream.readUnshared、XMLDecoder.readObject Yaml.load 、XStream.fromXML 、 ObjectMapper.readValue 、 JSON.parseObject
url 跳转	sendRedirect、setHeader、forward
不安全组件暴露	activity、Broadcast Receiver、Content Provider、Service、 inter-filter
日志记录敏感信息	log log.info logger.info
代码执行	eval、system、exec

## 2.1 可基于关键词审计的漏洞

### 2.1.1 密码硬编码

审计方法

密码硬编码最容易找，直接用 Sublime Text 打开项目目录，然后按 Ctrl + Shift + F 进行全局搜索 password 关键词：



### 2.1.2 反射型 XSS

审计方法：反射型 XSS 一般 fortify 一般都能扫描出来

如果是手工找，可全局搜索以下关键词

getParamter

<%=

param.

漏洞代码示例:

## 1. EL 表达式输出请求参数:

```
70807\代码整合\jsp\smp\safetyLiability\resourceAffiliationManage_list.jsp (代码整合) - Sublime Text (UNREGISTERED)
Find Results x resourceAffiliationManage_list.jsp x liability_main.jsp x SmpKpiappealController.java x UploadAndDownloadUtil.java x
165
166 var jqHandler={
167
168   init:function(){
169     var gridUrl = '${base }/resource/affiliation/viewAssetList';
170     var groupId=${param.groupId};
171     var viewType=${param.viewType}";
172     var name = $("#name").val();
173     var currentPostData = {
174       "name":name,
175       "groupId":groupId,
176       "viewType":viewType
177     };
178     var mygrid = $("#collectors").jqGrid({
179       url:gridUrl, //通过共享的方式去获取Id值的过程的词
180       datatype: "json",
181       mtype: "POST"
182     });
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199 /**
200  * 通过条件查询列表的过程
201  */
202   function queryJobByCnd()
203   {
204     var groupId=${param.groupId};
205     var viewType=${param.viewType}";
206     var name = $("#name").val();
207     var currentPostData = {
208       "name":name,
209       "groupId":groupId,
210       "viewType":viewType
211     };
212     var gridUrl = '${base }/resource/affiliation/viewAssetList';
213     $("#collectors").jqGrid('setGridParam',{url:gridUrl,postData:currentPostData}).trigger("reloadGrid");
214   }
215 }
```

代码在 170 行和 305 行处获取请求参数中的 groupId 值, 在未经检查参数合法性的情况下输出在 JavaScript 代码中, 存在反射型 XSS 漏洞。

## 2. 输出 getParamter 获取的参数

```
Project Summary SelectNumber.jsp
21 long orgID =SessionManager.getUser().getOrgId();
22 String regionId = (String) SessionManager.getUser().get(
23     ISysmgrConstant.REGION_ID);
24 String realRegionId =regionId;
25 String orgName= SessionManager.getUser().getOrgName();
26 String resSpecId = request.getParameter("resSpecId")==null?"":(String)request.g
27
28 //CRMXZ_REQ_20131113_0419 add liuml
29
30 IBOBsParaDetailValue[] bsParaDetailValue1 = SoConstUtil.getParaDetailValues("X"
31 String noPermissionCertType = "";//没有权限的不能显示的证件类型 介绍信、十丘证、营业执照
```

然后在 224 行打印到如下的 js 代码中:

```
Project Summary | SelectNumber.jsp
219 var baoOrgName = "<%=orgName%>";
220 var formInfo = g_FormRowSetManager.get("frmNormal_userForm");
221 var isFamily = '<%=HttpUtil.getAsString(request, "IS_FAMILY")%>'; //是否家庭套餐成
222 var familyMemOffer = '<%=HttpUtil.getAsString(request, "FAMILY_MEMBER_OFFER")%>';
223 var billFlag = 0;
224 var resSpecId = "<%=resSpecId%>";
225
226 var noPermissionCertType = "<%=noPermissionCertType%>";
227 var delCertType = "<%=delCertType%>";
228
229
```

## 2.1.3 存储型 XSS

审计方法：方法主要有两种：

1. 全局搜索数据库的插入语句(关键词: insert,save,update)，然后找到该插入语句所属的方法名如(insertUser())，然后全局搜索该方法在哪里被调用，一层层的跟踪。直到getParamter()方法获取请求参数的地方停止，如果没有全局 XSS 过滤器，跟踪的整个流程都没有对获取的参数过滤，则存在存储型 XSS。

```
54 resp.setContentType("text/html;charset=UTF-8");
55 try {
56     System.out.println(req.getQueryString());
57     String app_id = req.getParameter("app_id");
58     String Mobile = req.getParameter("phone");
59     String cid = req.getParameter("cid");
60     String clecs = req.getParameter("clecs");
61     System.out.println("app_id=" + app_id + "phone=" + Mobile + "cid="
62         + cid + "clecs=" + clecs);
63     // JDBC 连接
64     Class.forName("com.mysql.jdbc.Driver");
65
66     // 连接数据库
67     conn = DriverManager.getConnection(DB_URL, USER, PASS);
68
69     // SQL 语句
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
26
```

```

40 public ActionForward execute(ActionMapping mapping, ActionForm form,
41     HttpServletRequest request, HttpServletResponse response,
42     ActionErrors errors, HttpSession session) {
43     logger.info("enter into execute of AddCertAppTypeAction ");
44     try {
45         String usertype = request.getParameter("usertype");
46         String name = request.getParameter("name");
47         logger.info("usertype= " + usertype + " name=" + name);
48         if (!CheckEmpty.isEmpty(usertype) && !CheckEmpty.isEmpty(name)) {
49             // 填充VO
50             CertApplyTypeView view = new CertApplyTypeView();
51             view.setType(usertype);
52             view.setName(name);
53             view.setRaid(getRaid(request));
54             CertApplyTypeDelegate delegate = new CertApplyTypeDelegate();
55             // 添加证书应用类型
56             int result = delegate.addCertAppType(view);
57             logger.info("left execute of AddCertAppTypeAction, result is "
58                 + result);
59             if (result == RAConstant.OPERATE_SUCCESS) {
60                 // 审计日志
61                 String log = request.getParameter("log");

```

代码中 45 行和 46 行获取 usertype 和 name 的值，然后在 56 行存进数据库由于没有过滤传进来的参数，所以会在显示时出来触发 XSS

## 2.1.4SQL 注入

审计方法：

SQL 注入一般 fortify 一般都能扫描出来

手动找的话，一般直接搜索 select、update、delete、insert 关键词就会有收获

如果 sql 语句中有出现+ append、\$ ( ) # 等字眼，如果没有配置 SQL 过滤文件，则判断存在 SQL 注入漏洞

```

D:\系统代码\园区系统\南基数据抽取与数据服务系统20170804\bd\ecpp\src\main\java\com\chinatmg\ecpp\action\DataModelTaskSasController.java:
776 }
777 page.setCount(sourceCube.getDocsNum().intValue());
778: ResultSet rs = impala.queryData("select " + StringUtils.join(fieldList, ",") + " from " + tableName
779     + " where dt=" + sourceCube.getDocsDate() + " "
780     + " order by 1 limit " + page.getPageSize() + " offset " + page.getStart());

D:\系统代码\园区系统\南基数据抽取与数据服务系统20170804\bd\ecpp\src\main\java\com\chinatmg\ecpp\action\DataModelTaskSpssController.java:
740 }
741 page.setCount(sourceCube.getDocsNum().intValue());
742: ResultSet rs = impala.queryData("select " + StringUtils.join(fieldList, ",") + " from " + tableName
743     + " where dt=" + sourceCube.getDocsDate() + " "
744     + " order by 1 limit " + page.getPageSize() + " offset " + page.getStart());

D:\系统代码\园区系统\南基数据抽取与数据服务系统20170804\bd\ecpp\src\main\java\com\chinatmg\ecpp\action\DataModelTaskSqlController.java:
425 }
426 page.setCount(sourceCube.getDocsNum().intValue());
427: ResultSet rs = impala.queryData("select " + StringUtils.join(fieldList, ",") + " from " + tableName
428 //     + " where dt=" + sourceCube.getDocsDate() + " "
429     + " order by 1 limit " + page.getPageSize() + " offset " + page.getStart());

D:\系统代码\园区系统\南基数据抽取与数据服务系统20170804\bd\ecpp\src\main\java\com\chinatmg\ecpp\action\MarketingActivityController.java:
328 response.setCharacterEncoding("GBK");
329 //找出所有策略最近一次的清单 load_date
330: String sql = "select distinct instance_id, load_date, ruleid from marketing_act_list where actid='+id+' order by
load_date desc";
331 Map<String, MarketingProgramInstance> instanceMap = new HashMap();
332 ImpalaCommonDao impala = ImpalaCommonDao.getInstance();
...
369 CsvWriter w = new CsvWriter(response.getOutputStream(), ',', Charset.forName("GBK"));
370 w.writeRecord(headers.toArray(new String[headers.length])); //表头
371: sql = "select instance_id, load_date, sms, " + StringUtils.join(indexes, ",") + " from marketing_act_list ";
372 sql += " where actid="+id+" and ruleid in (" + StringUtils.join(ruleIds, ",") + ") ";
373 sql += " and load_date in (" + StringUtils.join(load_date, ",") + ") order by instance_id";

D:\系统代码\园区系统\南基数据抽取与数据服务系统20170804\bd\ecpp\src\main\java\com\chinatmg\ecpp\action\UserGroupController.java:

```

当找到某个变量关键词有 SQL 注入漏洞时，还可以直接全局搜索那个关键词找出类似漏洞的文件，上面中可以直接全局搜索 tableName 关键词：

```

D:\系统代码\园区系统\南基数据抽取与数据服务系统20170804\bd\ecpp\src\main\java\com\chinatmg\ecpp\flow\FlowS
216: boolean flag = true;
217: try {
218:     String tableName = managementService.getTableName(ProcessDefinition.class);
219:     String sql = "update " + tableName + " set name_=? where id_=?";
220:     Object[] obj = new Object[]{name, id};
221:     jdbcTemplate.update(sql, obj);

D:\系统代码\园区系统\南基数据抽取与数据服务系统20170804\bd\ecpp\src\main\java\com\chinatmg\ecpp\quartz\Ecpp
157: if(tmpTasks != null && tmpTasks.length > 0){
158:     ImpalaCommonDao impala = new ImpalaCommonDao();
159:     for(String tableName : tmpTasks){
160:         String databaseName = "hxzc";
161:         String where = "";
162:         if(tableName.contains(".")){//自定义表名
163:             String[] ss = tableName.split("\\.");
164:             databaseName = ss[0];
165:             tableName = ss[1];
166:         }
167:         if(tableName.contains("#")){//自定义where条件
168:             String[] ss = tableName.split("#");
169:             tableName = ss[0];
170:             where = ss[1].replaceAll("'", "'\\''");//把条件里面的单引号变成嵌套单引号
171:         }
172:         ...
174:         shell.append("sh /sybase/backup5/shell/IQTTransfer.sh ");
175:         shell.append(databaseName);
176:         shell.append(" ").append(tableName);
177:         if(StringUtils.isNotEmpty(where)){
178:             shell.append(" ").append(where).append(" ");
179:         }
180:     }
181: }

```

要查找那个页面调用到含有漏洞的代码，就要跟踪方法的调用栈。以上面的注入点 `tableName` 为例：

双击打开该文件，然后查看该变量所在函数：

```

730:
731: /**
732:  * 查看模型任务生成的文件
733:  * @param tableName impala表的英文名
734:  * @param page
735:  * @return
736:  * @update 张磊|2016-07-22 旧版本传入参数为模型任务ID，改为传入模型输出列表表英文名
737:  */
738: @RequestMapping("/lookOverCubeFile")
739: public ModelAndView lookOverCubeFile(String tableName, Long lookOverId, PageInfo page) {
740:     ModelAndView mv = new ModelAndView("dataModel/taskSas/taskSas_lookover_list");
741:     String msg = null;
742:     if(tableName == null){
743:         msg = "宽表英文名为空";
744:     }else{
745:         //ModelTaskSpss entity = null;
746:         DataCubeAttttable sourceCube = null;
747:         List<Map<String, Object>> codeValue = new LinkedList<Map<String, Object>>();
748:         List<String> codeName = new LinkedList<String>();
749:         List<String> fieldList = new ArrayList<String>();
750:         List<Map<String, String>> fieldAndCodeList = new ArrayList<Map<String, String>>();
751:         ImpalaCommonDao impala = new ImpalaCommonDao();
752:         try {
753:             entity = service.getId(lookOverId);
754:             if(entity==null){
755:                 msg = "对象不存在[ID: "+lookOverId+"]";
756:             }else{
757:                 String tableName = "dmt.ecpp " + entity.getId();

```

发现该函数对应的 URL 为 `/lookOverCubeFile`，对应的功能为查看模型任务生成的文件。

## 2.1.5 任意文件下载

审计方法：全局搜索以下关键词

`fileName`

`filePath`

`getFile`

`getWriter`

漏洞示例：

```
007\代码整合\java\main\com\venustech\tsac\cupid\smf\view\system\safety\controller\SafetyAssessReportController.java (代码整合) - Sublime Text (UNREGISTERED)
Find Results x SafetyAssessReportController.java x SmpKpiappealController.java x resourceResponsibility_list.jsp x resourceAffiliationManage_list.jsp x liability_main.jsp x UploadAndDownload
182
183 /**
184  * 下载安全评估报告文件
185  */
186 @At("/download")
187 @Fail("error")
188 @Ok("void")
189 @Log(value = "附件下载")
190 public void downloadFile(HttpServletRequest request, ServletContext context, HttpServletResponse response,
191 @Param("affixalName") String affixalName, @Param("uuidString") String reportCheckFileID)
192 throws FileNotFoundException, IOException {
193     String fileName = new String(affixalName.getBytes("ISO-8859-1"), "UTF-8");
194     if (fileName != null && fileName.length() > 0) {
195         String path = "D:/nanfang/apache-tomcat-7.0.78/webapps/smf/upload/safetyAssess";
196         String downPath = path + "/" + reportCheckFileID + "/" + fileName;
197         logger.info("downPath=====+++++++" + downPath);
198         download(downPath, response, request, false);
199     } else {
200         return;
201     }
202 }
203
```

代码在 downloadFile() 函数中获取请求参数中的 affixalName 的值，然后赋值给 fileName 变量，接着在 196 行处通过拼接字符串赋值给 downPath 变量，然后在 198 行处调用 download 函数并把 downPath 的值传进函数，download 函数的代码如下：

```
Find Results x SafetyAssessReportController.java x SmpKpiappealController.java x resourceResponsibility_list.jsp x resourceAffiliationManage_list.jsp x liability_main.jsp x UploadAndDownload
203
204 public void download(String filepath, HttpServletResponse response,
205 HttpServletRequest request, boolean isOnline) {
206     logger.info("filepath=====+++++++" + filepath);
207
208     OutputStream ops = null;
209     BufferedInputStream bis = null;
210     File file = new File(filepath);
211
212     try {
213         if (!file.exists()) {
214             response.sendError(404);
215             return;
216         }
217         bis = new BufferedInputStream(new FileInputStream(file));
218         byte[] buy = new byte[1024];
219         int len;
220         response.reset();
221         if (isOnline) { // 在线打开方式
222             URL url = new URL("file:/// " + filepath);
223             response.setContentType(url.openConnection().getContentType());
224             response.setHeader("Content-Disposition", "inline;filename="
225                 + file.getName());
226         } else {
227             String fileName = file.getName();
228
229             String agent = request.getHeader("USER-AGENT");
230             if (null != agent && -1 != agent.indexOf("MSIE")) { // IE
231                 fileName = URLEncoder.encode(fileName, "UTF-8");
232                 response.setContentType("application/x-download");
233                 response.setHeader("Content-Disposition",
234                     "attachment;filename=\"" + fileName + "\"");
235             } else if (null != agent && -1 != agent.indexOf("Mozilla")) { // Firefox
236                 fileName = new String(fileName.getBytes("GB2312"),
237                     "ISO-8859-1");
238
239                 response.setContentType("application/x-download");
240                 response.setHeader("Content-Disposition",
241                     "attachment;filename=\"" + fileName + "\"");
242             }
243         }
244     }
245 }
```



```

234         "attachment;filename=\"" + fileName + "\"";
235     } else if (null != agent && -1 != agent.indexOf("Mozilla")) { // FirFox
236         fileName = new String(fileName.getBytes("GB2312"),
237             "ISO-8859-1");
238
239         response.setContentType("application/x-download");
240         response.setHeader("Content-Disposition",
241             "attachment;filename=\"" + fileName + "\"");
242     }
243 }
244
245 ops = response.getOutputStream();
246 while ((len = bis.read(buy)) > 0) {
247     ops.write(buy, 0, len);
248 }
249 } catch (Exception e) {
250     // TODO: handle exception
251 } finally {

```

download 函数把 filePath 处的文件写到 http 响应中，在整个流程中并没有对文件名的合法性进行检查，存在任意文件下载漏洞，如通过把 affixalName 的值设置为 ../../WEB-INF/web.xml 可以下载网站的 web.xml 文件。

## 2.1.6 任意文件删除

审计方法：任意文件删除漏洞搜索以下关键词可以找到：

delete, deleteFile, fileName, filePath

漏洞案例：

```

DeleteBakAction.java
private static final Logger logger = Logger
    .getLogger(DeleteBakAction.class);

@Override
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response,
    ActionErrors errors, HttpSession session) {
    logger.info("enter into execute of DeleteBakAction ");
    request.getSession().setAttribute("returnURL", "/dbbackup/preDBBackup.do");
    try {
        DBBackupService ds = new DBBackupService();
        String fileName = request.getParameter("fileName");
        boolean res = false;
        if (fileName != null) {
            res = ds.deleteFile(fileName);
        }
        if (res) {
            messageSuccess(request, "file.operation.success");
            return mapping.findForward("success");
        } else {
            messageError(request, "file.operation.fail");
            return mapping.findForward("bigerror");
        }
    } catch (Exception e) {
        logger.error("Exception in execute is ", e);
        messageError(request, "system.exception");
        return mapping.findForward("bigerror");
    }
}

```

代码在 41 行获取 fileName 的值，在 44 行处调用 ds.deleteFile() 函数删除文件，该函数的代码如下：



```

149  /**
150   * 删除文件
151   * @param fileName
152   * @return
153   */
154  public boolean deleteFile(String fileName) {
155      boolean res = false;
156      URL urls = Thread.currentThread().getContextClassLoader().getResource(
157          "");
158      String path = urls.getPath(); // 删除文件
159      String filePath = path.replaceAll("WEB-INF/classes", "databackupzone");
160      String name = filePath + fileName;
161      File f = new File(name);
162      if (f.exists()) {
163          res = f.delete();
164      }
165      return res;
166  }
167

```

在整个流程中并没有对文件名的合法性进行检查，存在任意文件删除漏洞，如通过把 fileName 的值设置为../WEB-INF/web.xml 可以删除网站的 web.xml 文件。

## 2.1.7 文件上传

审计方法：

文件上传可以搜索以下关键词：（需注意有没有配置文件上传白名单）

upload, write, fileName, filePath

在查看时，主要判断是否有检查后缀名，同时要查看配置文件是否有设置白名单或者黑名单，像下面这种是检查了的：

```

(outConfigFileName != null){
    String realName=outConfigFileName.getOriginalFilename();
    String extension = realName.lastIndexOf(".") == -1 ? "" : realName.substring(realName.lastIndexOf(".") + 1);
    //支持Excel格式
    if ("xls".equals(extension) || "xlsx".equals(extension)) {
        outConfigReader outReader = new outConfigReader();
        List<ModelTaskSpssOut> outs = outReader.readFromExcel(outConfigFileName);
        if(outs != null && 0 !=outs.size()){

```

下面的这种没检查：

```

List<FileItem> fileItems = servletFileUpload.parseRequest(request);
for (int i = 0; i < fileItems.size(); ++i) {
    FileItem fi = (FileItem) fileItems.get(i);
    String strFileName = fi.getName();
    if (strFileName != null && !"".endsWith(strFileName)) {
        String fileName = opId + "_" + getTimeSequence() + "."
            + getFileNameExtension(strFileName);

        String diskFileName = path + fileName;
        File file = new File(diskFileName);
        if (file.exists()) {
            file.delete();
        }
        fi.write(new File(diskFileName));
        resultArrayNode.add(fileName);
    }
}
.....

```

```

private String getFileNameExtension(String fullFileName) {
    if (fullFileName == null) {
        return null;
    }

    int pos = fullFileName.lastIndexOf(".");

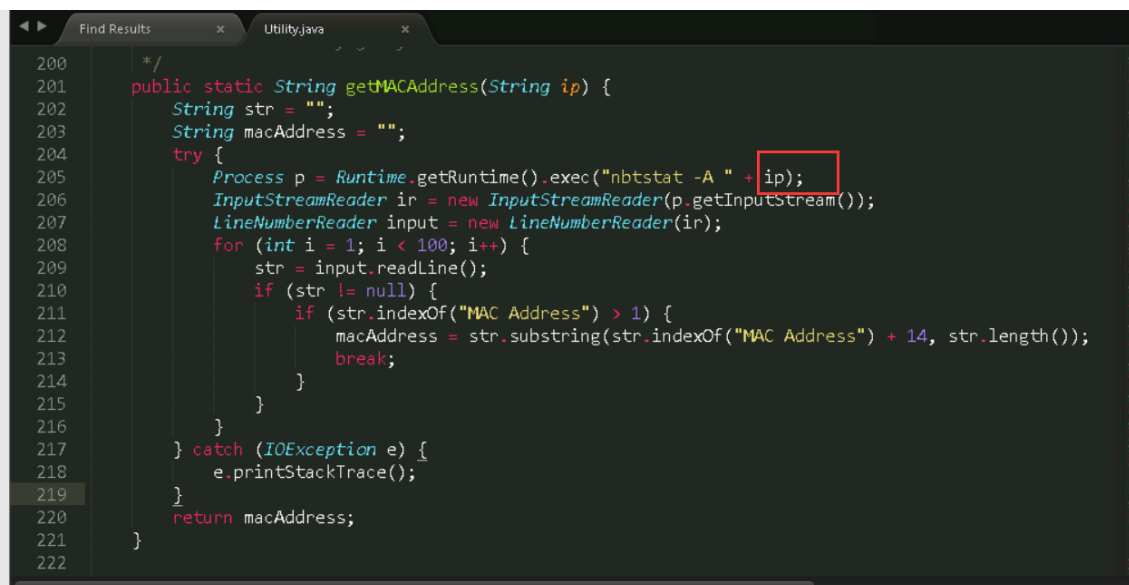
    if (pos != -1) {
        return fullFileName.substring(pos + 1,
fullFileName.length());
    } else {
        return null;
    }
}

```

## 2.1.8 命令注入

审计方法：可以搜索以下关键词：

getRuntime,exec,cmd,shell



在第 205 行中，通过拼接传过来的 ip 值来执行命令。如果 ip 值通过外部传入，则可以构造以下的 ip 值来执行 net user 命令：

127.0.0.1&&net user

## 2.1.9 缓冲区溢出

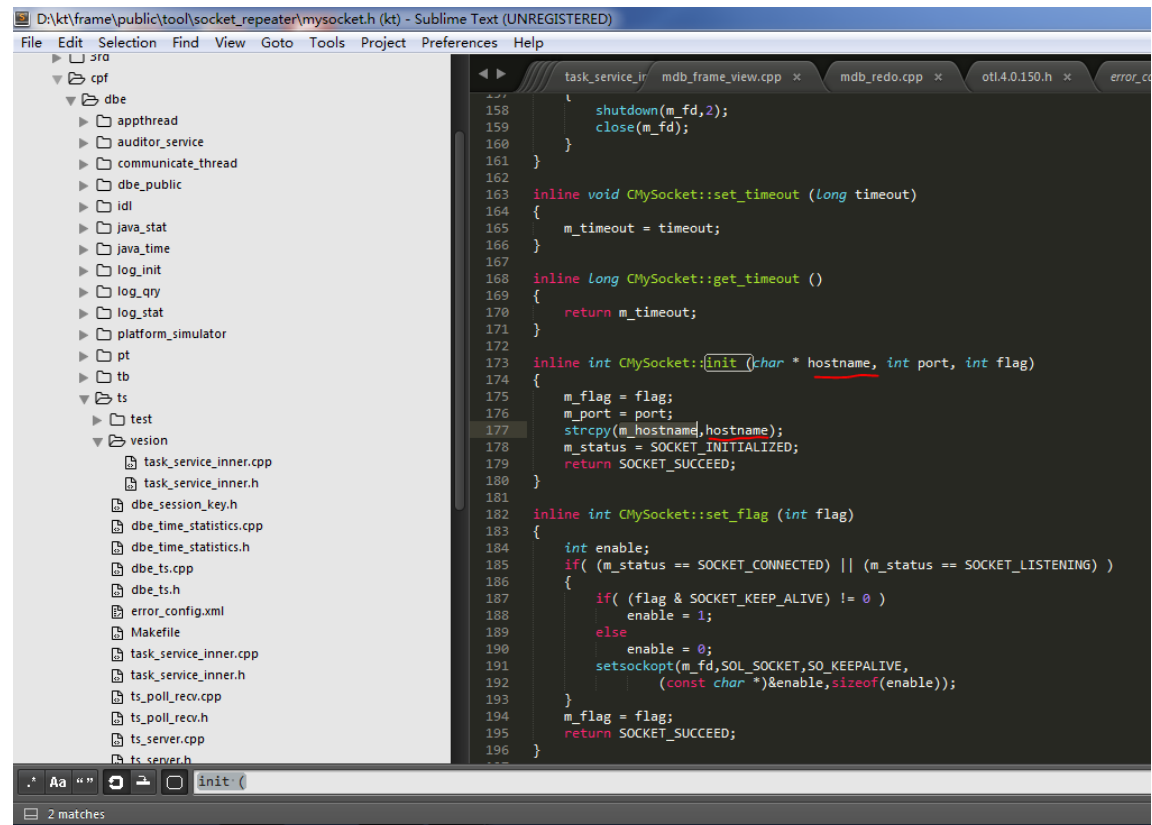
审计方法：主要通过搜索关键词定位，再分析上下文

可搜索以下关键字：

strcpy, strcat, scanf, memcpy, memmove, memcpy Getc(), fgetc(), getchar, read, printf

漏洞示例：

文件 `kt\frame\public\tool\socket_repeater\mysocket.h` 中第 177 行，这里的参数 `hostname` 拷贝到 `m_hostname`，具体如下图所示：



`m_hostname` 的大小为 `MAXNAME`

:

```
D:\kt\frame\public\tool\socket_repeater\mysocket.h (kt) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
└─ sra
  └─ cpf
    └─ dbe
      └─ appthread
      └─ auditor_service
      └─ communicate_thread
      └─ dbe_public
      └─ idl
      └─ java_stat
      └─ java_time
      └─ log_init
      └─ log_qry
      └─ log_stat
      └─ platform_simulator
      └─ pt
      └─ tb
      └─ ts
        └─ test
        └─ vesion
          └─ task_service_inner.cpp
          └─ task_service_inner.h
          └─ dbe_session_key.h
          └─ dbe_time_statistics.cpp
          └─ dbe_time_statistics.h
          └─ dbe_ts.cpp
          └─ dbe_ts.h
          └─ error_config.xml

79  /*
80  int m_flag;
81
82  /* Read Timeout */
83  long m_timeout;
84
85  /* Socket File Descriptor */
86  int m_fd;
87  /* Socket address */
88  struct sockaddr_in m_addr;
89
90  /* Socket status: 0 uninitialized
91  /* connected, 2 connected, 3 list
92  /* CMyServerSocket)
93  int m_status;
94
95  /* Host name */
96  char m_hostname[MAXNAME];
97
98  /* port number */
99  int m_port;
100 };
101
102 class CMyServerSocket : public CMySocket
103 {
104 public:
105     CMyServerSocket () {};;
106     ~CMyServerSocket ();
107
108     /* Listen on hostname and port, w
109     /* recived, it call callback func
110     /* ... */
```

继续看，可以看到大小为 255

```
D:\kt\frame\public\tool\socket_repeater\mysocket.h (kt) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
└─ sra
  └─ cpf
    └─ dbe
      └─ appthread
      └─ auditor_service
      └─ communicate_thread
      └─ dbe_public
      └─ idl
      └─ java_stat
      └─ java_time
      └─ log_init
      └─ log_qry
      └─ log_stat
      └─ platform_simulator
      └─ pt
      └─ tb
      └─ ts
        └─ test
        └─ vesion
          └─ task_service_inner.cpp
          └─ task_service_inner.h
          └─ dbe_session_key.h
          └─ dbe_time_statistics.cpp
          └─ dbe_time_statistics.h

1  #ifndef _MY_SOCKET_H_
2  #define _MY_SOCKET_H_ 1
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <unistd.h>
7  #include <errno.h>
8  #include <string.h>
9  #include <strings.h>
10 #include <sys/poll.h>
11 #include <sys/select.h>
12 #include <sys/types.h>
13 #include <sys/socket.h>
14 #include <arpa/inet.h>
15 #include <netdb.h>
16 #include <sys/time.h>
17
18 #define MAXNAME 255
19
20 /* define stauts */
21 #define SOCKET_UNINITIALIZED 0
22 #define SOCKET_INITIALIZED 1
23 #define SOCKET_CONNECTED 2
24 #define SOCKET_LISTENING 3
25
26 /* define error code */
27 #define SOCKET_SUCCEED -1 /* open
```

如果传入的长度比 255 要大，就会造成缓冲区溢出。

## 2.1.10 XML 注入

审计方法：

XML 解析一般在导入配置、数据传输接口等场景可能会用到，可通过搜索以下关键字定位：

DocumentBuilder、XMLStreamReader、SAXBuilder、SAXParser、SAXReader、XMLReader、SAXSource、TransformerFactory、SAXTransformerFactory、SchemaFactory

涉及到 XML 文件处理的场景可留意下 XML 解析器是否禁用外部实体，从而判断是否存在 XXE 漏洞示例：

在代码 6 行处、获取 DOM 解析器，解析 XML 文档的输入流，得到一个 Document

```
1 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
2     String result="";
3     try {
4         //DOM Read XML
5         DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
6         DocumentBuilder db = dbf.newDocumentBuilder();
7         Document doc = db.parse(request.getInputStream());
8
9         String username = getValueByTagName(doc,"username");
10        String password = getValueByTagName(doc,"password");
11        if(username.equals(USERNAME) && password.equals(PASSWORD)){
12            result = String.format("<result><code>%d</code><msg>%s</msg></result>",1,username);
13        }else{
14            result = String.format("<result><code>%d</code><msg>%s</msg></result>",0,username);
15        }
16    } catch (ParserConfigurationException e) {
17        e.printStackTrace();
18        result = String.format("<result><code>%d</code><msg>%s</msg></result>",3,e.getMessage());
19    } catch (SAXException e) {
20        e.printStackTrace();
21        result = String.format("<result><code>%d</code><msg>%s</msg></result>",3,e.getMessage());
22    }
23    response.setContentType("text/xml;charset=UTF-8");
24    response.getWriter().append(result);
25 }
```

如果没有禁用 DTD 则存在 XXE 漏洞，以下代码为 XXE 防御代码

```
28
29
30 dbf.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true); //禁用DTDs (doctypes),几乎可以防御所有xml实体攻击
31 //如果不能禁用DTDs,可以使用下两项,必须两项同时存在
32 dbf.setFeature("http://xml.org/sax/features/external-general-entities", false); //防止外部普通实体POC 攻击
33 dbf.setFeature("http://xml.org/sax/features/external-parameter-entities", false); //防止外部参数实体POC攻击
```

## 2.1.11 日志记录敏感信息

审计方法：

通过搜索关键词 log.info logger.info 来进行定位

在SFtpOperate.java文件中，代码134行处，直接将用户名密码记录在日志中

```

D:\2017\2017年度代审\业支\智能终端\智能终端20180307全\crm4apps\src\main\java\com\ailk\util\SftpOperate.java:
81 //      upload.setSizeMax(41943040); // 41943040MB
82 //      // %x%E1'<
83: //      sftp = SftpOperate.connect(ftpConfig.getHostIp(), ftpConfig.getPort(),
ftpConfig.getUserName(),ftpConfig.getPassword());
84      InputStream in = new FileInputStream(fileItem);
85 //      SftpOperate.upload(ftpConfig.getRemotePath(), in, newFileName, sftp);
86
87:      FtpUtil.uploadFile(ftpConfig.getHostIp(), ftpConfig.getPort(), ftpConfig.getUserName(), ftpConfig.getPassword(),
ftpConfig.getRemotePath(), newFileName, in);
88
89      fileMap.put("OriginalFileName", fileName);
90
91
123  * @param port%E1Ü
124  * @param username0Äv$Äü
125:  * @param passwordÄÜÄ8
126  * @return
127  */
128: public static Channelsftp connect(String host, int port, String username, String password) {
129      Channelsftp sftp = null;
130      try {
131
132          jsch.getSession(username, host, port);
133          Session sshSession = jsch.getSession(username, host, port);
134:          logger.info "Session param: " + "username: " + username + "password: " + password + "host: " + host;
135:          sshSession.setPassword(password);
136          Properties sshConfig = new Properties();
137          sshConfig.put("StrictHostKeyChecking", "no");

```

## 2.1.12 URL 跳转

审计方法：通过搜索以下关键词定位：

sendRedirect、setHeader、forward

需注意有没有配置 url 跳转白名单

漏洞示例：

以下代码中 40 行处只判断 site 是否为空，没有对 url 进行判断是否为本站 url，导致了 url 跳转漏洞

```

38
39      String site = request.getParameter("url");
40      if(!site.isEmpty()){
41          response.sendRedirect(site);
42      }
43

```

## 2.1.13 敏感信息泄露及错误处理

审计方法：查看配置文件是否配置统一错误页面，如果有则不存在此漏洞，如果没有再通过搜索以下关键词搜索定位，

Getmessage、exception

漏洞代码示例：

在以下文件中代码 89 行处打印出程序发生异常时的具体信息

```

85     }
86     <%
87     } catch (Exception e) {
88     %>
89     alert('<%= e.getMessage() %>');
90     window.close();
91     <%
92     }

```

### 2.1.14 反序列化漏洞

审计方法：

Java 程序使用 `ObjectInputStream` 对象的 `readObject` 方法将反序列化数据转换为 java 对象。但当输入的反序列化的数据可被用户控制，那么攻击者即可通过构造恶意输入，让反序列化产生非预期的对象，在此过程中执行构造的任意代码。

反序列化操作一般在导入模版文件、网络通信、数据传输、日志格式化存储、对象数据落磁盘或 DB 存储等业务场景,在代码审计时可重点关注一些反序列化操作函数并判断输入是否可控，如下：

`ObjectInputStream.readObject`

`ObjectInputStream.readUnshared`

`XMLDecoder.readObject`

`Yaml.load`

`XStream.fromXML`

`ObjectMapper.readValue`

`JSON.parseObject`

漏洞示例：

以下代码中，程序读取输入流并将其反序列化为对象。此时可查看项目工程中是否引入可利用的 `commons-collections 3.1`、`commons-fileupload 1.3.1` 等第三方库，即可构造特定反序列化对象实现任意代码执行。

```

44
45 //读取输入流,并转换对象
46 InputStream in=request.getInputStream();
47 ObjectInputStream ois = new ObjectInputStream(in);
48 //恢复对象
49 ois.readObject();
50 ois.close();
51

```

### 2.1.15 不安全组件暴露

审计方法：

通过查看配置文件 `AndroidManifest.xml`,查看`<inter-filter>`属性有没有配置 `false`

`AndriodManifest.xml` 文件中,代码 24 行处 `activity` 组件添加`<intent-filter>`属性,没有配置 `false`,默认组件可被导出



```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.silk.module.groucustbill"
4     android:sharedUserId="com.silk.android.portal"
5     android:versionCode="1"
6     android:versionName="1.03" >
7
8     <uses-sdk
9         android:minSdkVersion="14"
10        android:targetSdkVersion="17" />
11
12     <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
13     <uses-permission android:name="android.permission.INTERNET" />
14
15     <application
16         android:allowBackup="true"
17         android:icon="@drawable/ic_launcher"
18         android:label="@string/app_name"
19         android:theme="@style/Theme.NG40">
20         <activity
21             android:name="com.silk.module.groucustbill.AcctBillQryActivity"
22             android:label="@string/app_name" >
23             <intent-filter>
24                 <action android:name="android.intent.action.MAIN" />
25             </intent-filter>
26         </activity>
27
28         <activity android:name="com.silk.module.groucustbill.AcctBillDetailActivity">
29             </activity>
30     </application>
31
32
33
34

```

## 3.1 其他漏洞审计方法

### 3.1.1 CSRF

审计方法:通过查看配置文件有没有配置 csrf 全局过滤器, 如果没有则重点看每个操作前有没有添加 token 的防护机制

在 Smpkpiappealcontroller.java 中 200 处, 直接用 ids 控制删除操作, 而没有添加防 csrf 的随机 token 验证检查, 存在 csrf 漏洞。

```

194 /**
195  * 删除考核规则
196  */
197 @4t("/deletesmpKpiDeclara")
198 @Ok("jsp:jsp.smp.kpi.smpKpiappealList")
199 @Log(isEnabled=false)
200 public void deletesmpKpiDeclara(HttpServletRequest request, @Param("ids") String ids){
201     String idArray [] = ids.split(",");
202     for(String id : idArray){
203         smpDeclaraService.delete(Long.parseLong(id));
204     }
205 }
206
207
208

```

Java/main/com/venustech/tsoc/cupid/smp/kpi/dao/smpkpideclardao.java 517 行, 对传入的 ids 进行删除操作。

```

508      /**
509       * 通过id删除申告
510       * @param id
511       * @return
512       */
513      public int delete(DeclarationBean declaration) {
514
515          return super.delete(declaration);
516      }
517      public int delete(String ids){
518          String sql = "delete from smp_kpi_declaration where id = " + ids;
519          Sql sqls = Sqls.create(sql);
520          this.execute(sqls);
521          return sqls.getInt();
522      }
523
524

```

### 3.1.2 Struts2 远程代码执行漏洞

审计方法：查看 struts 插件的版本信息是否为漏洞版本

漏洞版本查询网址：<https://www.exploit-db.com/>

### 3.1.3 越权操作

审计方法：重点关注用户操作请求时查看是否有对当前登陆用户权限做校验从而确定是否存在漏洞，有些厂商会使用一些主流的权限框架，例如 shiro ,spring security 等框架，那么需要重点关注框架的配置文件以及实现方法

漏洞示例：

在以下文件中采用了 shiro 框架进行权限控制，在代码 58-72 行处为控制访问路径的权限设置，51-55 行处为对 admin 路径下访问限制，只有 SysyUserFilter 设置了 isAccessAllowed 方法,其他过滤均没有

```
44
45
46 @Bean
47 public ShiroFilterFactoryBean shiroFilter(DefaultWebSecurityManager securityManager) {
48     ShiroFilterFactoryBean shiroFilterFactoryBean = new ShiroFilterFactoryBean();
49     shiroFilterFactoryBean.setSecurityManager(securityManager);
50
51     Map<String, Filter> filterMap = new HashMap<>();
52     filterMap.put("sysUser", new SysUserFilter());
53     filterMap.put("authc", new AjaxAuthenticationFilter());
54     filterMap.put("users", new UserAuthFilter());
55     filterMap.put("perms", new PermissionAuthFilter());
56     filterMap.put("roles", new RoleAuthFilter());
57     filterMap.put("jCaptchaValidate", jCaptchaValidateFilter());
58     shiroFilterFactoryBean.setFilters(filterMap);
59
60     Map<String, String> filterChainDefinitionMap = new LinkedHashMap<String, String>();
61     filterChainDefinitionMap.put("/admin/login", "jCaptchaValidate");
62     filterChainDefinitionMap.put("/admin/login/sms", "anon");
63     filterChainDefinitionMap.put("/admin/logout", "anon");
64     filterChainDefinitionMap.put("/admin/captcha", "anon");
65     filterChainDefinitionMap.put("/admin/captcha/sms", "anon");
66     filterChainDefinitionMap.put("/admin/sync/serverPort", "anon");
67     filterChainDefinitionMap.put("/swagger-ui.html", "anon");
68     filterChainDefinitionMap.put("/webjars/springfox-swagger-ui/**", "anon");
69     filterChainDefinitionMap.put("/swagger-resources/**", "anon");
70     filterChainDefinitionMap.put("/v2/api-docs/**", "anon");
71
72     filterChainDefinitionMap.put("/admin/**", "sysUser,users,perms,roles");
73     shiroFilterFactoryBean.setFilterChainDefinitionMap(filterChainDefinitionMap);
74
75     return shiroFilterFactoryBean;
76
77 }
78
79 @Bean
80 public DefaultWebSecurityManager securityManager() {
81     DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();
82     securityManager.setRealm(userRealm());
83     // 自定义session管理
84     securityManager.setSessionManager(sessionManager());
85 }
```

SysUserFilter 中 isAccessAllowed 具体实现方法如下, 90-93 行处没有对是否为当前用户进行判断, 导致了越权

```
84
85
86 }
87
88 @Override
89 protected boolean isAccessAllowed(ServletRequest request, ServletResponse response, Object mappedValue)
90     throws Exception {
91     User user = (User) request.getAttribute(Constants.CURRENT_USER);
92     if (user == null) {
93         return true;
94     }
95
96     if (User.STATUS_DELETE.equals(user.getStatus()) || User.STATUS_LOCKED.equals(user.getStatus())) {
97         getSubject(request, response).logout();
98         // by mail to do ajax
99         saveRequestAndRedirectToLogin(request, response);
100         return false;
101     }
102     return true;
103 }
```

其他过滤文件均只设置了 onAccessDenied()方法

```
4 import com.ai.iisc.es.core.base.util.MessageUtils;
5 import com.ai.iisc.es.core.web.converter.MessageConverterUtil;
6 import com.ai.iisc.es.utils.AjaxUtil;
7 import com.alibaba.fastjson.JSON;
8 import org.apache.shiro.web.filter.authz.PermissionsAuthorizationFilter;
9 import org.apache.shiro.web.util.WebUtils;
10 import org.springframework.http.HttpStatus;
11
12 import javax.servlet.ServletException;
13 import javax.servlet.ServletResponse;
14 import javax.servlet.http.HttpServletRequest;
15 import java.io.IOException;
16
17 public class PermissionAuthFilter extends PermissionsAuthorizationFilter {
18     /**
19      * shiro认证perms资源失败后回调方法
20      * @param servletRequest
21      * @param servletResponse
22      * @return
23      * @throws IOException
24      */
25     @Override
26     protected boolean onAccessDenied(HttpServletRequest servletRequest, ServletResponse servletResponse) throws IOException {
27         if (AjaxUtil.isAjaxRequest(servletRequest)) {
28             ExceptionInfo result = new ExceptionInfo("1102", MessageUtils.getMessage("1102"));
29             HttpServletResponse httpServletResponse = WebUtils.toHttp(servletResponse);
30             httpServletResponse.setCharacterEncoding("UTF-8");
31             httpServletResponse.setContentType("application/json");
32             httpServletResponse.setStatus(HttpStatus.FORBIDDEN.value());
33             httpServletResponse.getWriter().write(JSON.toJSONString(MessageConverterUtil.autoWrapException(result)));
34             httpServletResponse.sendError(HttpStatus.FORBIDDEN.value());
35             return false;
36         } else { //如果是普通请求进行重定向
37             return super.onAccessDenied(servletRequest, servletResponse);
38         }
39     }
40 }
```

如果没有使用框架的话，就要注意每个操作是否有权限

代码 7 行处获取 session 里的 username，只判断了 username 是不是为空，如果在截取数据包的时候将 username 再重新赋一个值就有可能造成越权漏洞。

```
1 <%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8"
2 isELIgnored="false"%>
3 <%@ taglib prefix="c" uri="http://java.sun.com/jstl/core_rt"%>
4 <!DOCTYPE html>
5 <%
6 // 获取session，如果为空重定向到登录页面
7 String username = (String) session.getAttribute("username");
8 if (username == null) {
9     response.sendRedirect(request.getContextPath() + "/index.jsp");
10 }
11 %>
12 <%
13 String loginName="";
14 String feeYear="";
15 String taskType="";
16 String starttime="";
17 %>
18 <html lang="zh-cn">
```

以这个年度服务费用编制功能为例，测试一下，代码如下所示：

```
204 </a> <b class="arrow"></b>
205
206 <ul class="submenu">
207
208 <li id="fulccx" style="display:none"><a
209 href="/cmccmtf/CMCCMis3/toCreateProcess.action?p
210 rocessId=CMCCMIS_2017_000003_1.0&
211 loginName=<%=username%>"
212 target="_blank">年度服务费用编制</a></li>
213
214 <li id="tzlccx" style="display:none"><a
215 href="/cmccmtf/CMCCMis3/toCreateProcess.action?p
216 rocessId=CMCCMIS_2017_000002_1.0&
217 loginName=<%=username%>"
```

```
449
450
451
452
453
454
455
456
457
458
459
460

<table data-toggle="table" id="
taskPend" class="table
table-bordered table-hover"
data-page-list="[3,5]"
data-pagination="true">
  <thead>
    <tr>
      <th data-field="state"
class="center"
data-formatter="
getIndex">序号</th>
      <th data-field="
titleName" class="center
">工单标题</th>
      <th data-field="
ticketId" class="center"
">工单编号</th>
      <th data-field="
createUserId" class="
center">创建人</th>
      <th data-field="
createTime" class="
center">创建时间</th>
      <th data-field="
pendUrl" class="center"
data-formatter="
formatter" >操作</th>
    </tr>
```

### 3.1.4 会话超时设置

审计方法：

JavaWeb 应用会话超时设置一般有俩种方法：

一是在配置文件 web.xml 设置

```
1 <session-config>
2 <session-timeout>15</session-timeout>
3 </session-config>
```

二是通过 java 代码设置

```
HttpSession session = request.getSession();
session.setMaxInactiveInterval(60);//单位为秒
```

### 3.1.5 敏感数据弱加密

审计方法：

敏感数据弱加密主要看数据传输中的加密方法，一般写在工具类 util 中

以下文件中为 base64 编码方法

```
231
232
233     /**
234     * Base64编码
235     * @param str
236     * @return
237     */
237     public String getBase64(byte[] str){
238         if (null == str || 0 == str.length){
239             return null;
240         }
241         BASE64Encoder encoder = new BASE64Encoder();
242         String encode = encoder.encode(str);
243         return encode;
244     }
245
246 }
247
```

## 4.1 工具使用

### 1. Fortify

#### 1.1 新建扫描

##### 1.1.1 命令行自定义扫描目录：

如果想自定义 Fortify 扫描的目录的话，下面命令比较方便：

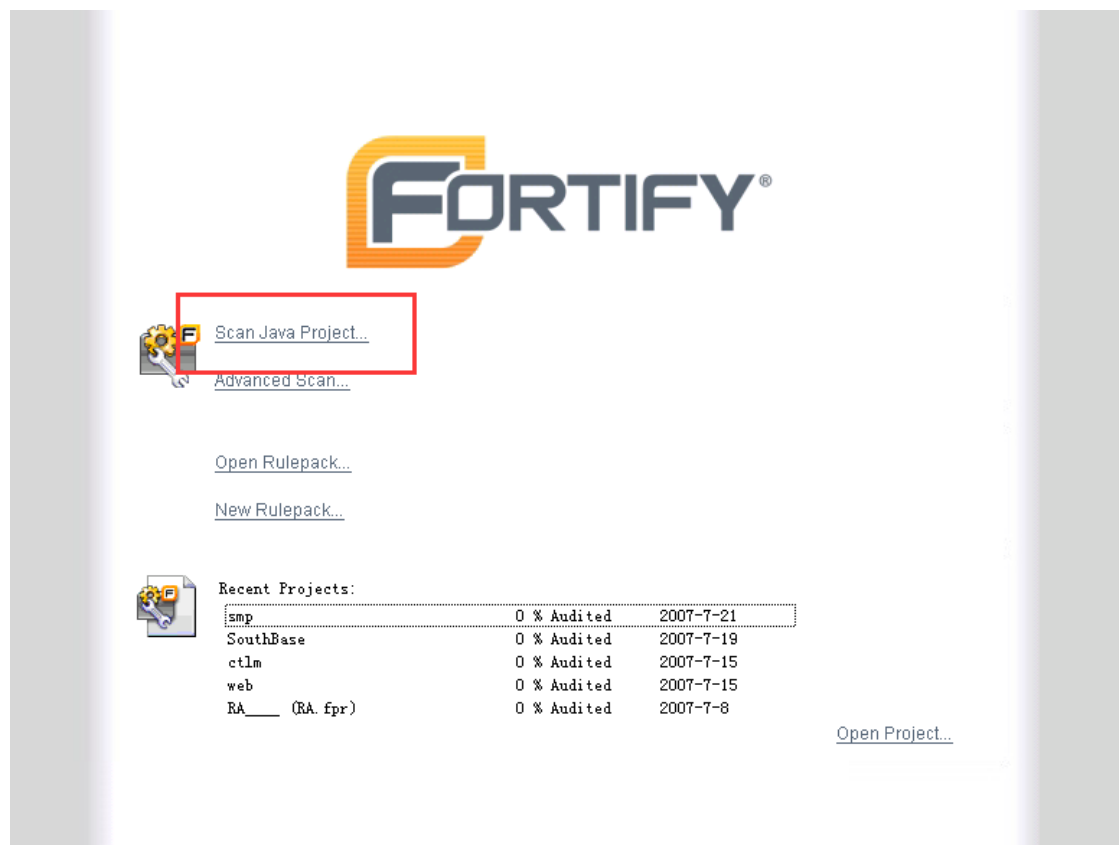
```
sourceanalyzer -scan -cp "lib/*.jar" "src/**/*.java" "web/**/*.jsp" -f result.fpr
```

-cp 指定类库的路径，如果没有就不用这个选项

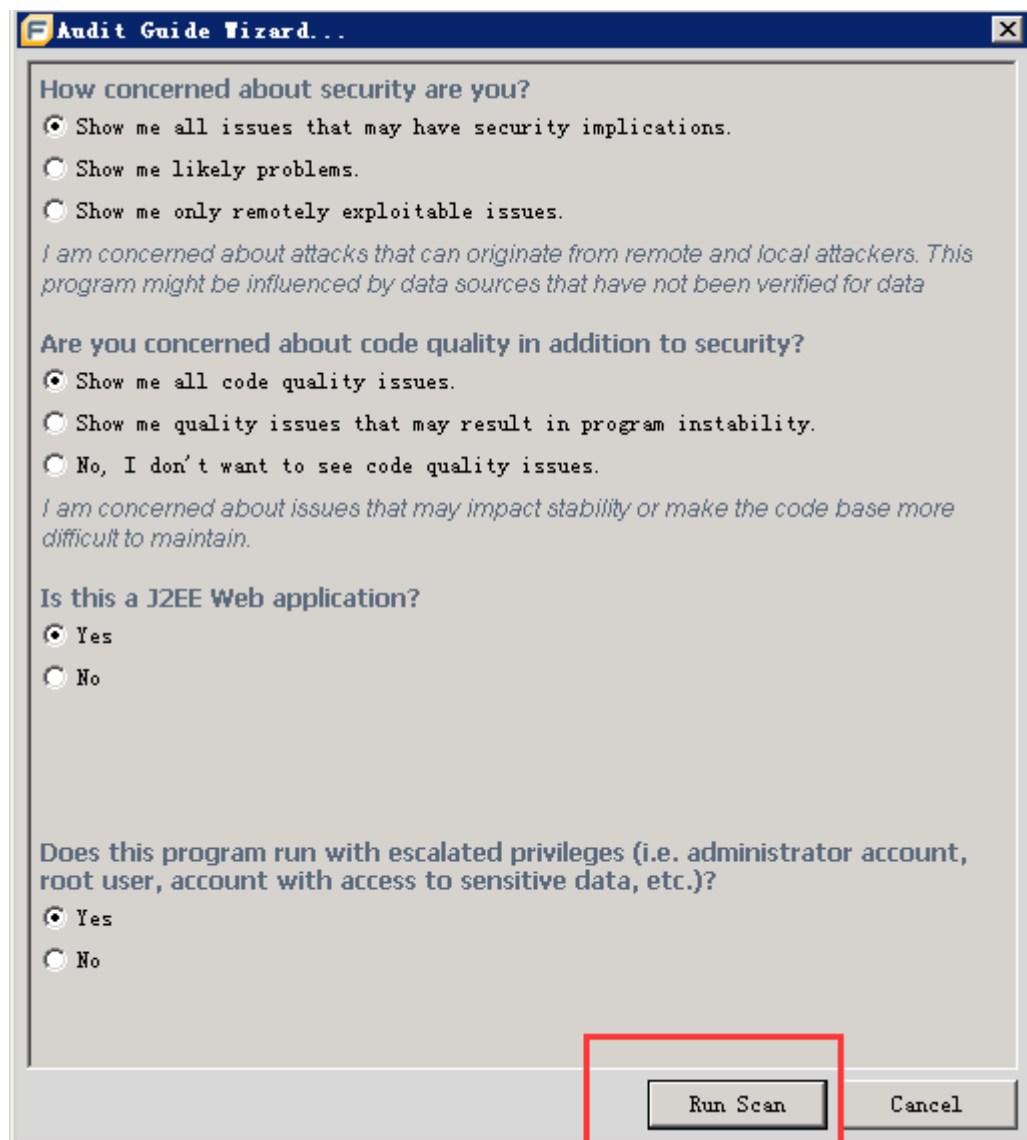
"src/\*\*/\*.java" "web/\*\*/\*.jsp" 这两个参数指定扫描 src 目录下的所有 java 文件和 web 目录中的所有 jsp 文件

-f 指定扫描结果的输出文件为 result.fpr,扫描完后双击就可以通过 Fortify 查看了。

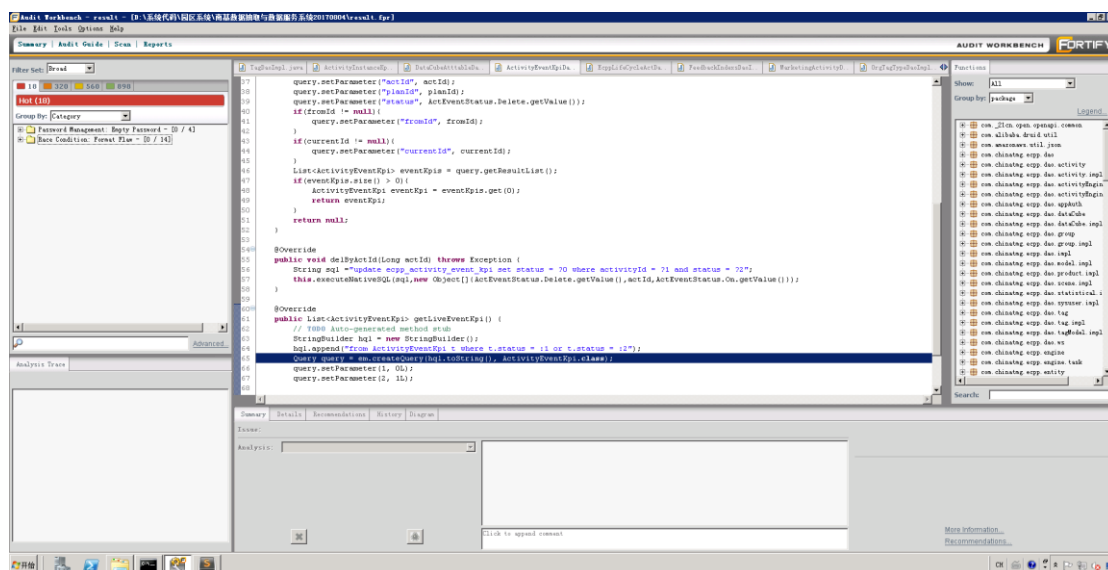
##### 1.1.2 图形化界面



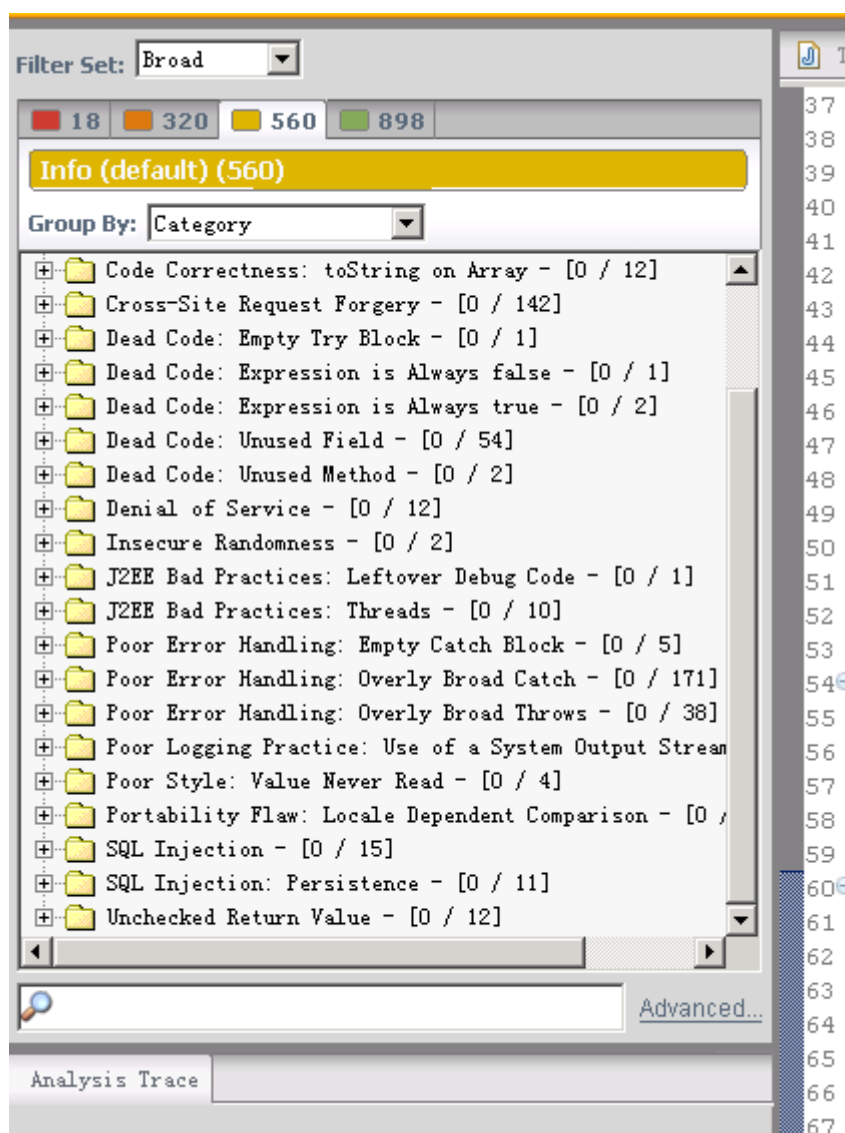




## 1.2 查看结果

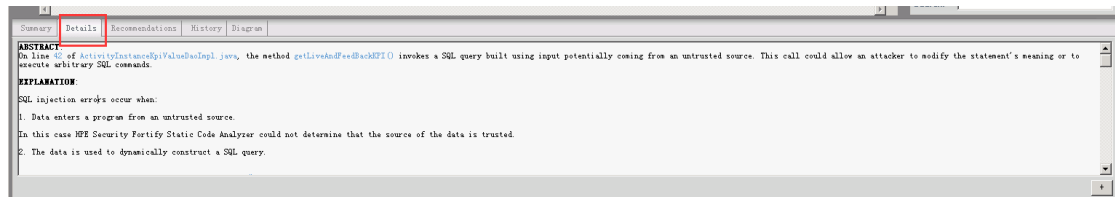


漏洞列表:



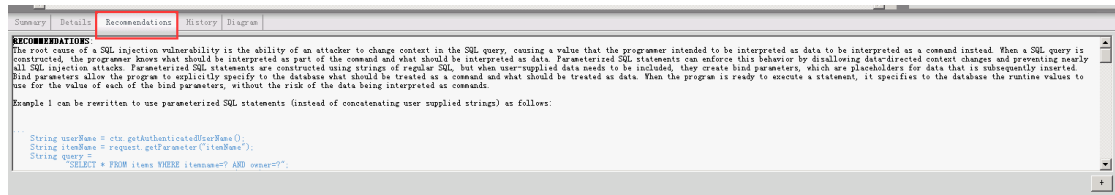
## 漏洞介绍:

### Details



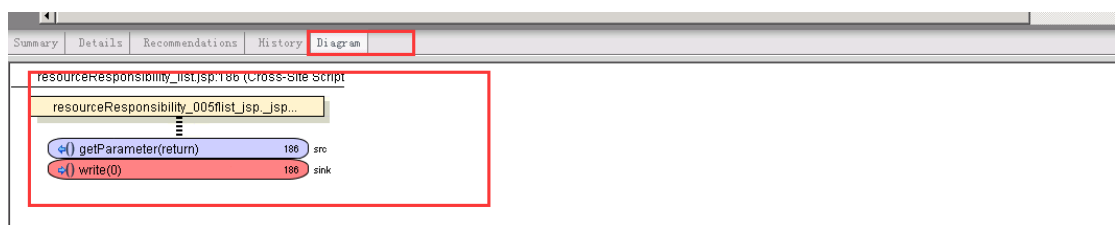
## 漏洞修复建议:

### Recommendations



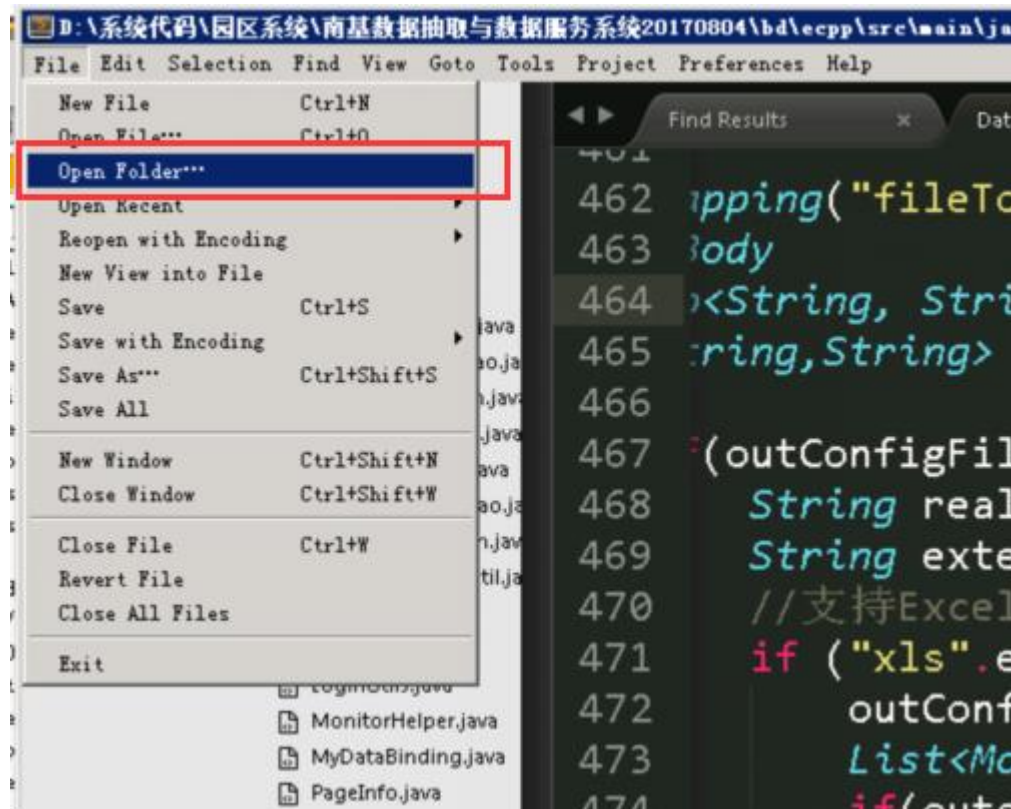
## 漏洞跟踪图:

### Diagram



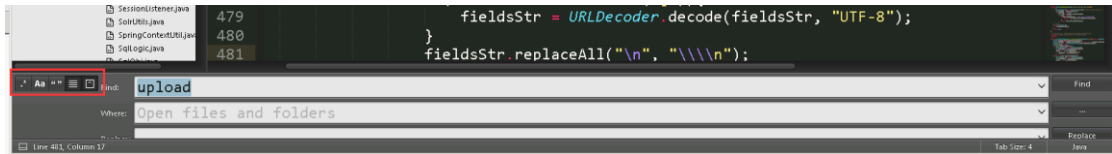
## 2. Sublime Text

### 2.1 打开项目相应目录:



## 2.2 全局搜索

Ctrl + Shift + F 全局搜索

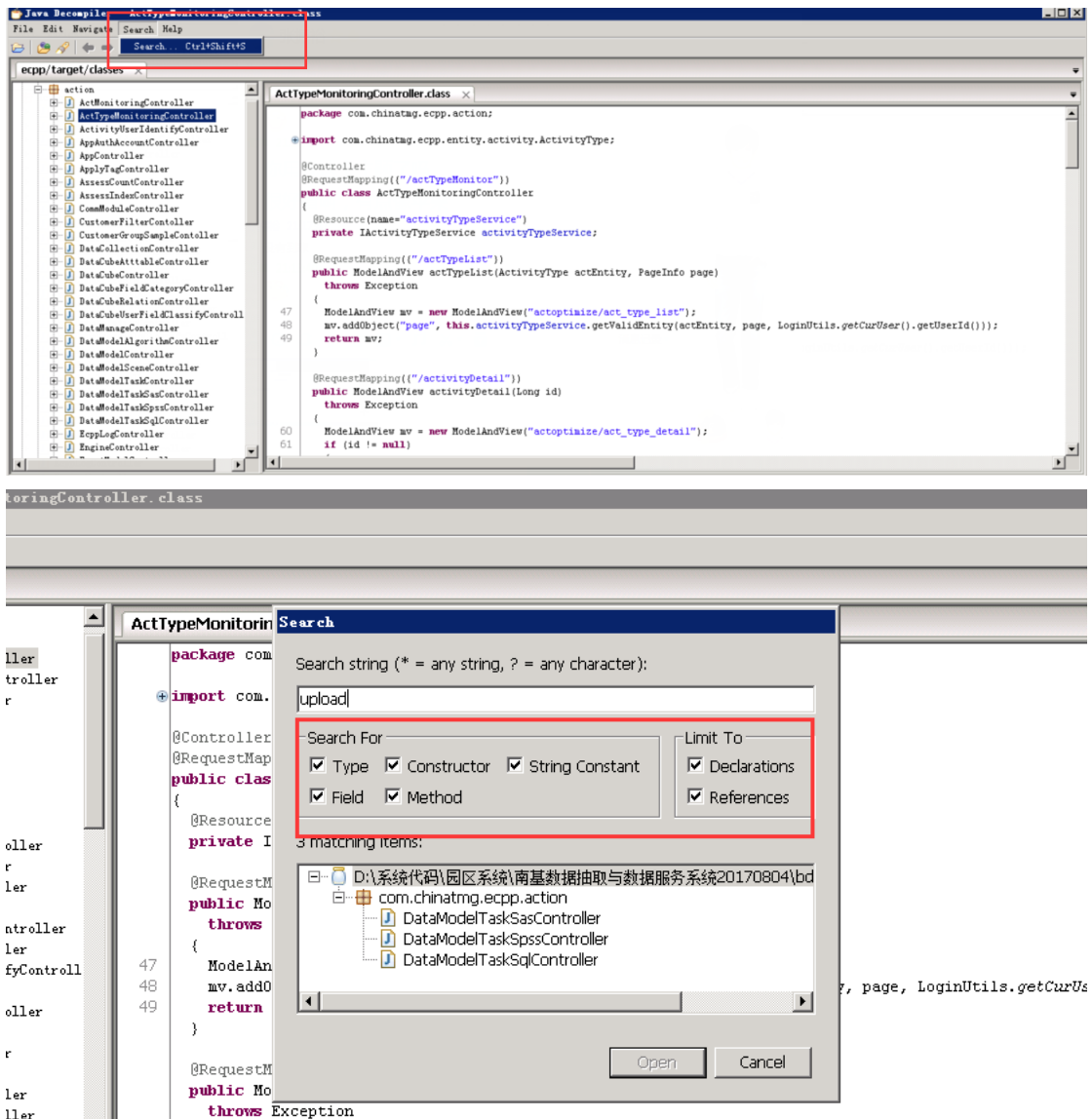


上面红框的几个图标可以设置是否大小写敏感，是否搜索整个单词。

## 3. JD-GUI

没源码时，要分析 jar 包或者 class 文件，就要用到 JD-GUI。

直接拖动某个 jar 或者 class 文件进 jd-gui 就可以打开了，然后搜索关键词审计：



把那些勾都勾上搜索。

#### 4. 文件浏览器

Windows 自带的文件浏览器可以方便地搜索某个文件或者 java,jsp 文件：



实际中，都是 Fortify、Sublime Text 和文件浏览器结合在一起用最高效。