Java 代码审计常用漏洞总结

1.1 审计方法总结

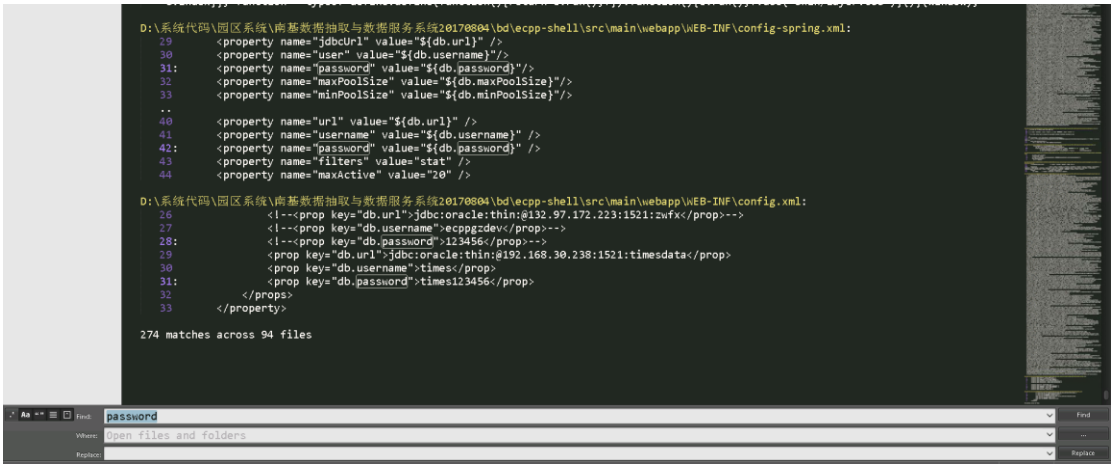
主要代码审计方法是跟踪用户输入数据和敏感函数参数回溯：
跟踪用户的输入数据，判断数据进入的每一个代码逻辑是否有可利用的点，此处的代码逻辑可以是一个函数，或者是条小小的条件判断语句。
敏感函数参数回溯，根据敏感函数，逆向追踪参数传递的过程。这个方法是最高效，最常用的方法。大多数漏洞的产生是因为函数的使用不当导致的，只要找到这些函数，就能够快速挖掘想要的漏洞。
以下是基于关键词审计技巧总结：
在搜索时要注意是否为整个单词,以及小写敏感这些设置

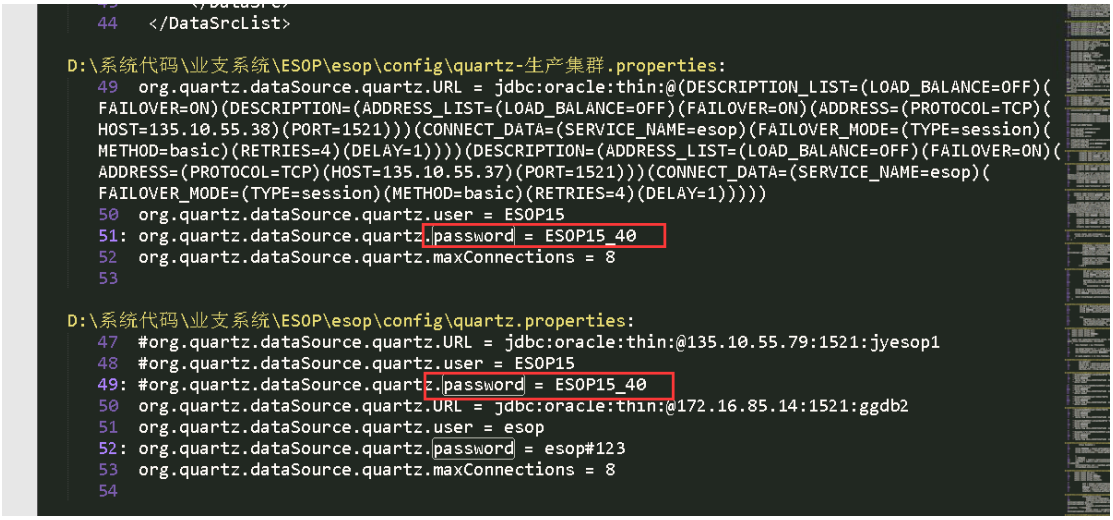
漏洞名称	关键词
密码硬编码、密码明文存储	password 、pass、jdbc
XSS	getParamter、<%=、param.
SQL 注入	Select、Dao 、from 、delete 、update、insert
任意文件下载	download 、fileName 、filePath、write、getFile、getWriter
任意文件删除	Delete、deleteFile、fileName 、filePath
文件上传	Upload、write、fileName 、filePath
命令注入	getRuntime、exec、cmd、shell
缓冲区溢出	strcpy,strcat,scanf,memcpy,memmove,memeccpy 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：



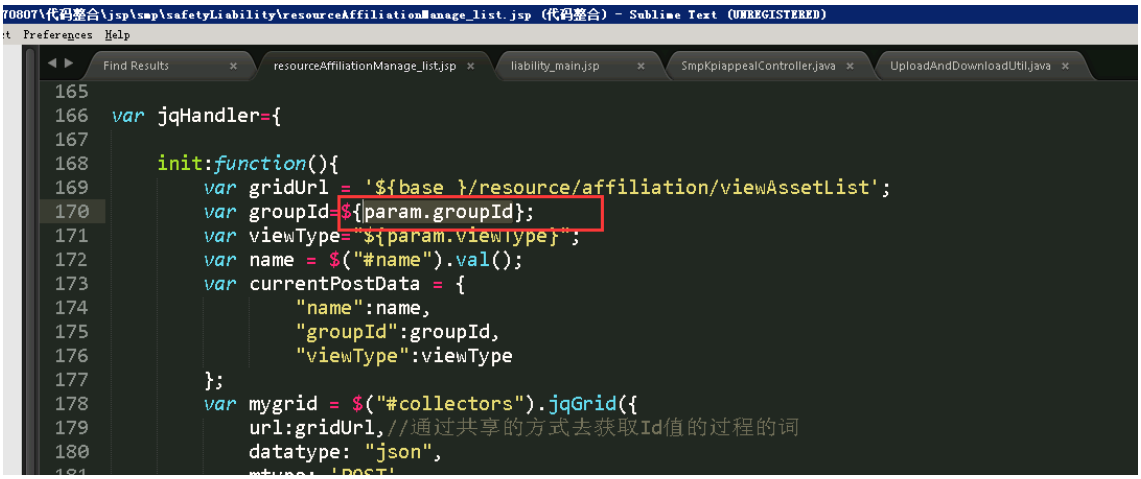


2.1.2 反射型 XSS

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：

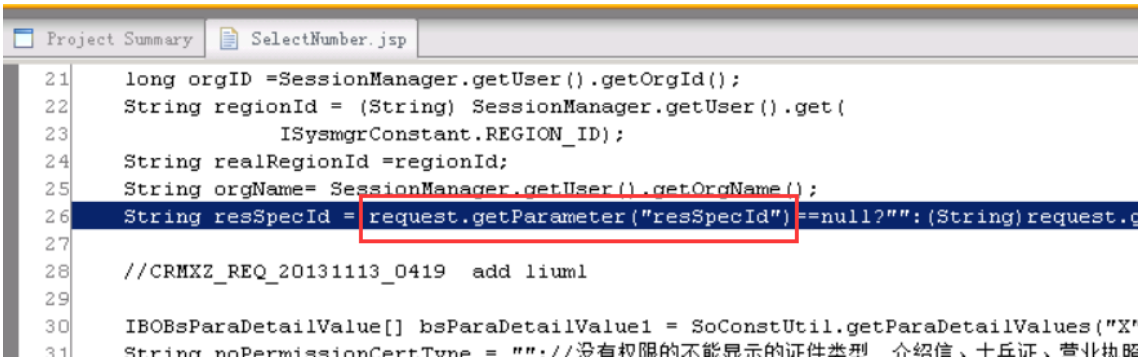


```
165
166  var jqHandler={
167
168      init:function(){
169          var gridUrl = '${base }/resource/affiliation/viewAssetList';
170          var groupId=${param.groupId};
171          var viewType="${param.viewType}";
172          var name = $("#name").val();
173          var currentPostData = {
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",
```

```
298
299  /**
300   * 通过条件查询列表的过程
301   */
302   function queryJoBByCnd()
303  {
304      var groupId=${param.groupId};
305      var viewType="${param.viewType}";
306      var name = $("#name").val();
307      var currentPostData = {
308              "name":name,
309              "groupId":groupId,
310              "viewType":viewType
311      };
312      var gridUrl = '${base }/resource/affiliation/viewAssetList';
313      $("#collectors").jqGrid('setGridParam',{url:gridUrl,postData:currentPostData}).trigger("reloadGrid");
314  }
```

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



```
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'
```

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：方法有主要有两种：

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              // y□□ JDBC □□□□□□
64              Class.forName("com.mysql.jdbc.Driver");
65
66              // □□k□□□□□□
67              conn = DriverManager.getConnection(DB_URL, USER, PASS);
68
69              // o□□ SQL □□□
```

```
154
155                 String sql3 = "insert into liu_order(channel_id,app_id,phonenumber"
156                         + ",format,price,fee,plat_id,create_time,trade_id,"
157                         + "firststatus,result) values(?,?,?,?,?,?,?,?,?,?,?)";
158                 pre = conn.prepareStatement(sql3);
159                 pre.setString(1, cid);// channel_id
160                 pre.setString(2, app_id);// app_id
161                 pre.setString(3, Mobile);// phonenumber
162                 pre.setString(4, format);// format
163                 pre.setString(5, price);// price
164                 pre.setString(6, "-" + fee);// fee
165                 pre.setString(7, plat_id);// plat_id
166                 pre.setString(8, create_time);// create_time
167                 // pre.setString(8,order_time);//order_time
168                 pre.setString(9, TransactionID);// trade_id
169                 pre.setInt(10, 2);// firststatus
170                 pre.setInt(11, ResultCode);// result
171                 int row = pre.executeUpdate();
172                 // □k□□p□□□³ u□
173                 if (row > 0) {
174                     System.out.println("□□□□□□" + row + "□□□□□□");
175                 }
176                 // liu_cache□□k□□□
```

2. 从 getParamter 关键词开始 ，跟踪请求参数，直到插入数据库的语句，如果中间没有过滤参数，则存在存储型 XSS。

```
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[]{}));//表头
371:                    sql = "select instance_id, load_date, sms, "+StringUtils.join(indexs, ",") + " 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\quarz\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                  }
...
174                      shell.append("sh /sybase/backup5/shell/IQTransfer.sh ");
175                      shell.append(databaseName);
176:                     shell.append(" ").append(tableName);
177                      if(StringUtils.isNotEmpty(where)){
178                          shell.append(" '").append(where).append("'");
...
```

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宽表英文名
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              DataCubeAtttable 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.getById(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

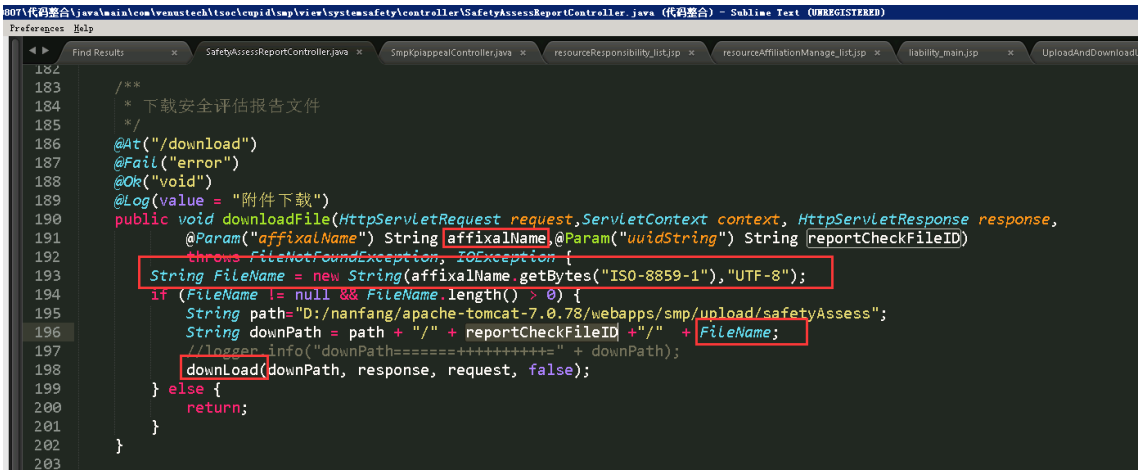
fileName

filePath

getFile

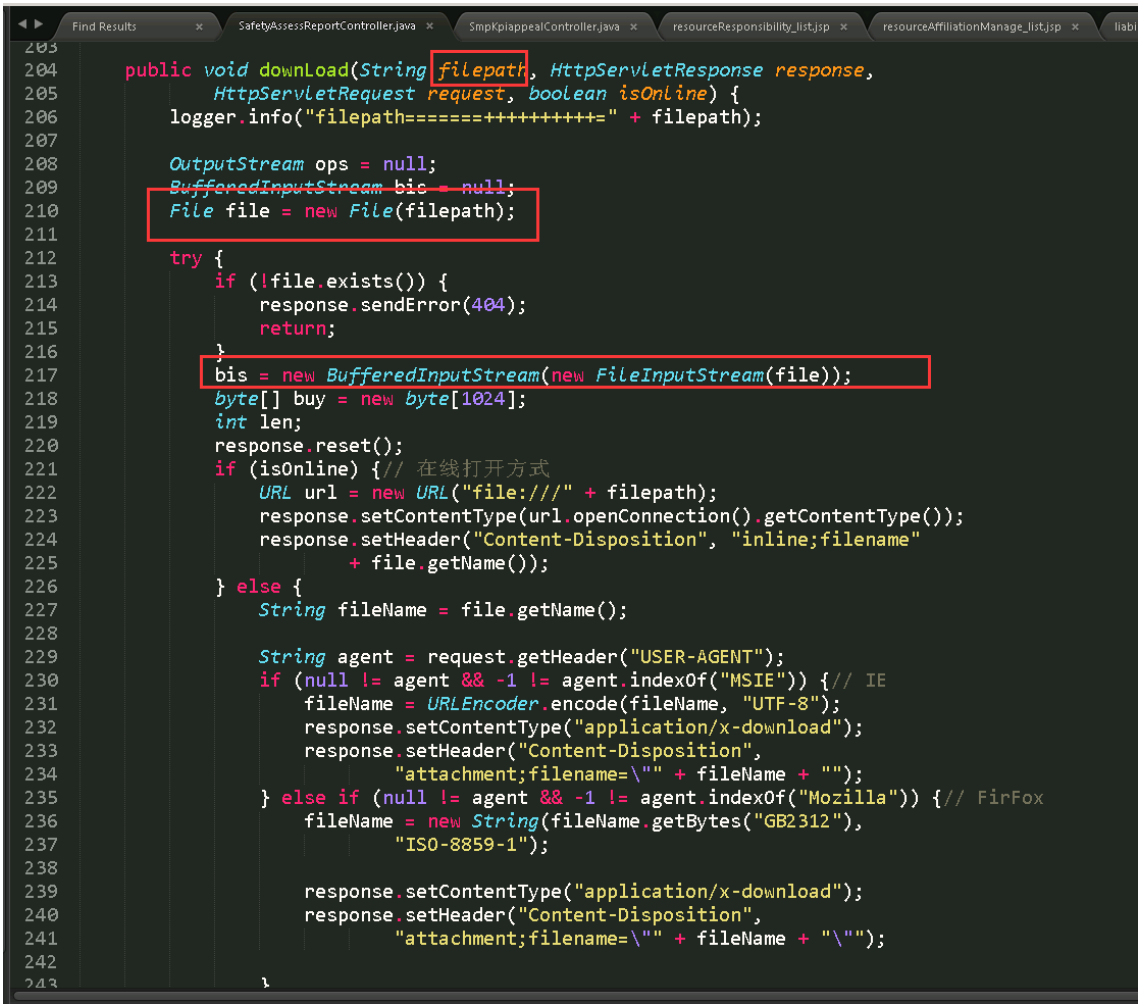
getWriter

漏洞示例：



```
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/smp/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     }
```

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：



```
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")) {// 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                 }
```

```
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：

```
月例行版本源代码20170714\增里代码\kmc系统代码\src\com\ccit\networld\kms\controller\dbbackup\DeleteBakAction.java (增里代码) -
Tools  Project  Preferences  Help
DeleteBakAction.java ×  DBackupDownloadAction.java ×  DBackupServiceImpl.java ×  Find Results ×  DBackup
30      private static final Logger logger = Logger
31              .getLogger(DeleteBakAction.class);
32
33      @Override
34      public ActionForward execute(ActionMapping mapping, ActionForm form,
35              HttpServletRequest request, HttpServletResponse response,
36              ActionErrors errors, HttpSession session) {
37          logger.info("enter into execute of DeleteBakAction ");
38          request.getSession().setAttribute("returnURL", "/dbbackup/preDBackup.do");
39          try {
40              DBackupService ds = new DBackupService();
41              String fileName = request.getParameter("fileName");
42              boolean res = false;
43              if (fileName != null) {
44                  res = ds.deleteFile(fileName);
45              }
46              if (res) {
47                  messageSuccess(request, "file.operation.success");
48                  return mapping.findForward("success");
49              } else {
50                  messageError(request, "file.operation.fail");
51                  return mapping.findForward("bigerror");
52              }
53          } catch (Exception e) {
54              logger.error("Exception in execute is    ", e);
55              messageError(request, "system.exception");
56              return mapping.findForward("bigerror");
57          }
```

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：

```java
    /**
     * É¾³ýÎÄ¼þ
     * @param fileName
     * @return
     */
    public boolean deleteFile(String fileName) {
        boolean res = false;
        URL urls = Thread.currentThread().getContextClassLoader().getResource(
                "");
        String path = urls.getPath(); // µÃµ½¹¤³ÌÂ·¾¶databackupzone
        String filePath = path.replaceAll("WEB-INF/classes", "databackupzone");
        String name = filePath + fileName;
        File f = new File(name);
        if (f.exists()) {
            res = f.delete();
        }
        return res;
    }
```

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：

```java
(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()){
```

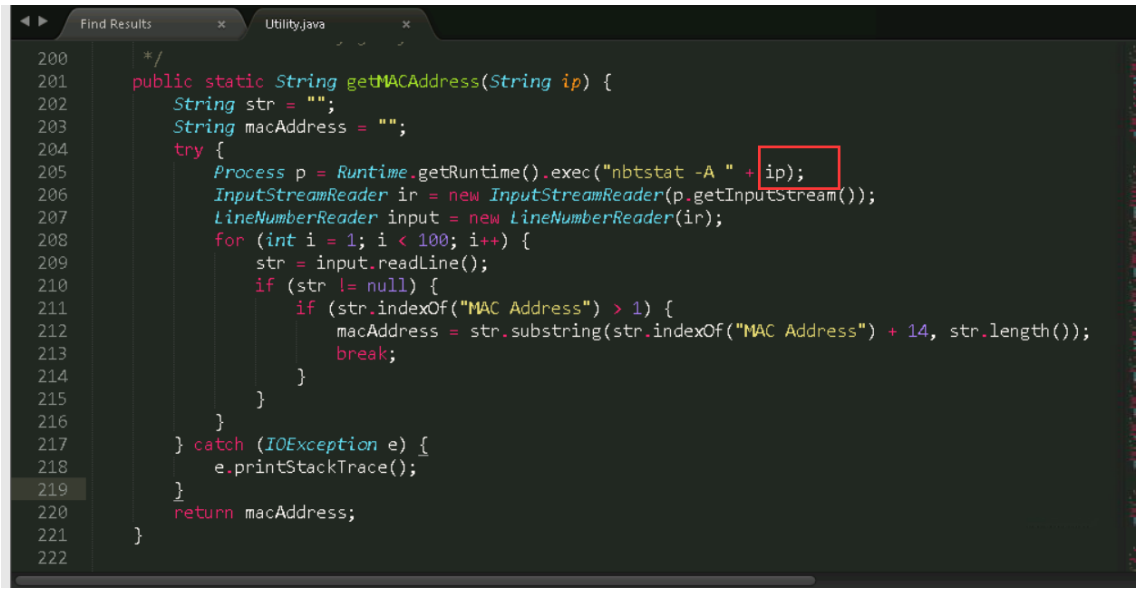
下面的这种没检查：

```
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



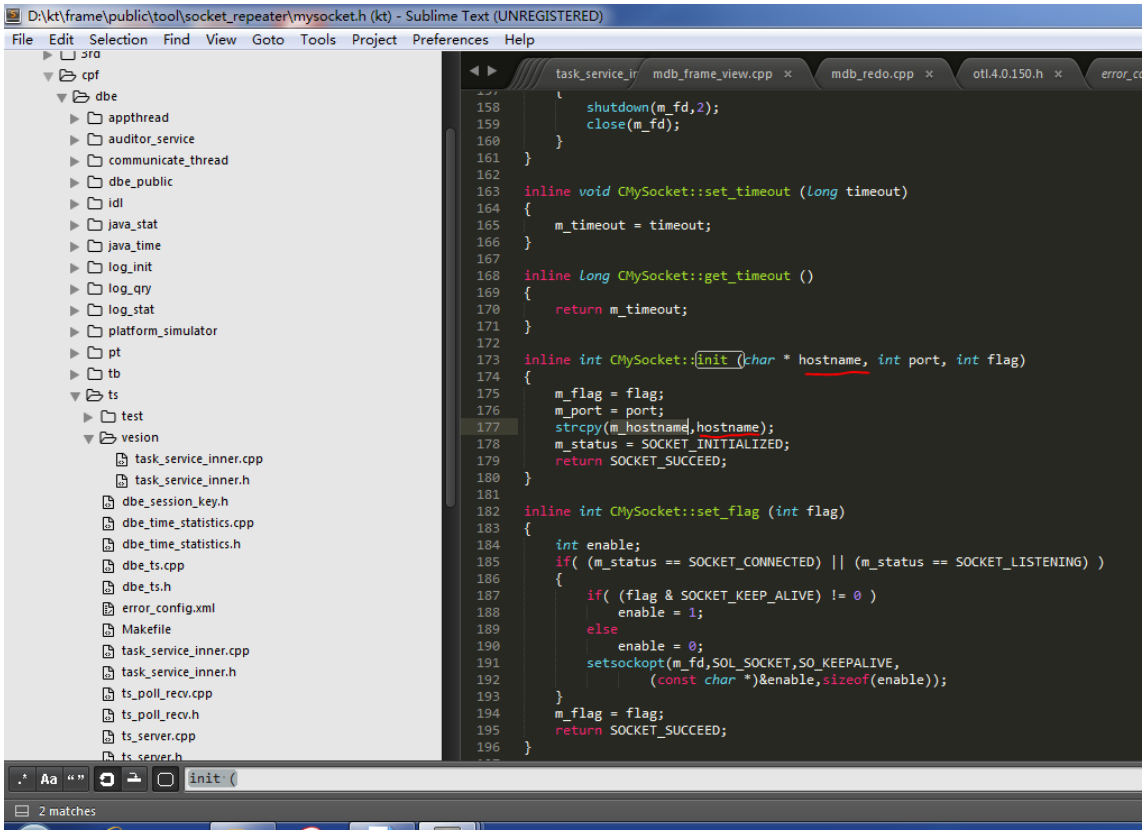
在第 205 行中，通过拼接传过来的 ip 值来执行命令。如果 ip 值通过外部传入，则可以构造以下的 ip 值来执行 net user 命令：

127.0.0.1&&net user

2.1.9 缓冲区溢出

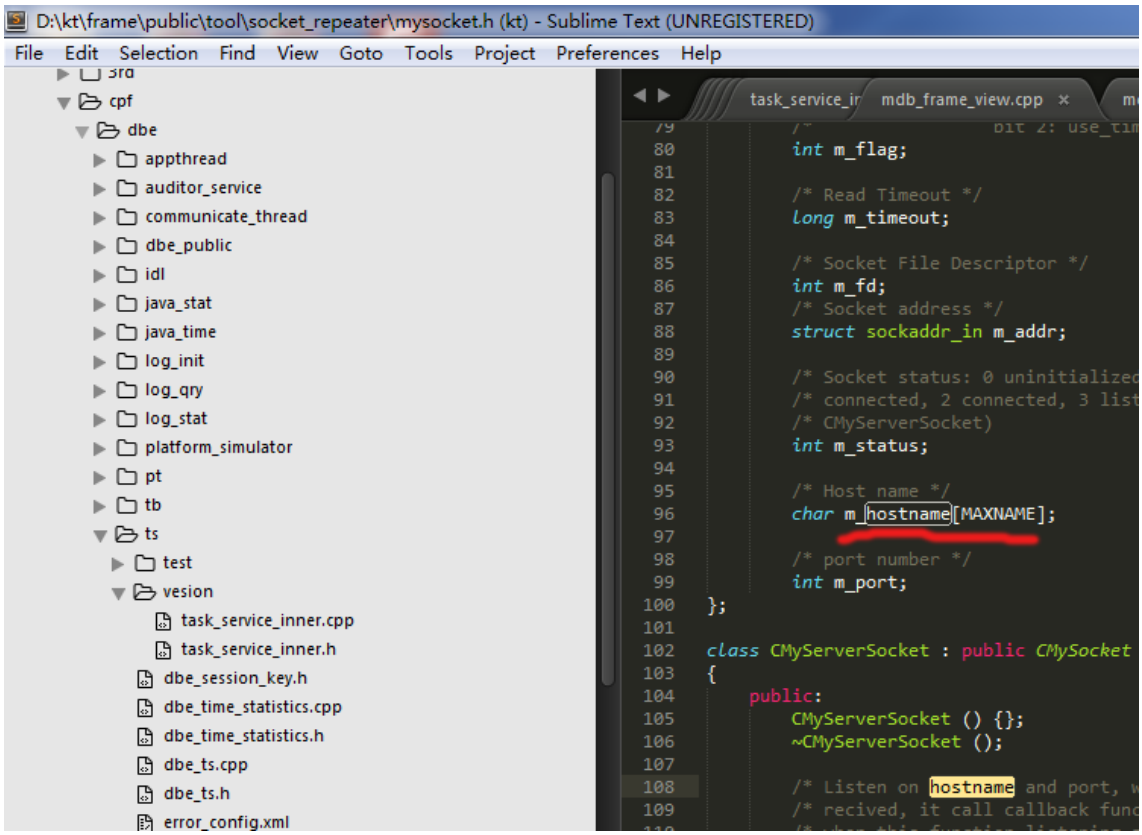
审计方法：主要通过搜索关键词定位，再分析上下文
可搜索以下关键字：
strcpy,strcat,scanf,memcpy,memmove,memeccpy Getc(),fgetc(),getchar;read,printf
漏洞示例：

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

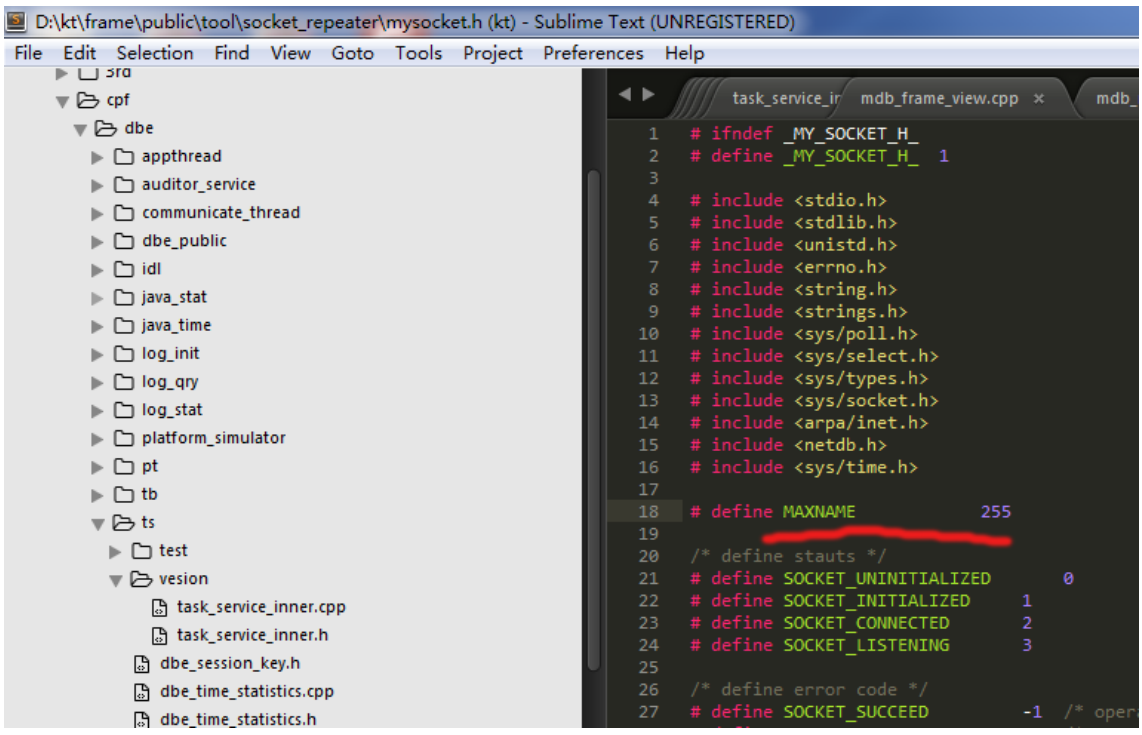


m_hostname 的大小为 MAXNAME

：



继续看，可以看到大小为 255



如果传入的长度比 255 要大，就会造成缓冲区溢出。

2.1.10XML 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

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String result="";
        try {
            //DOM Read XML
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.parse(request.getInputStream());

            String username = getValueByTagName(doc,"username");
            String password = getValueByTagName(doc,"password");
            if(username.equals(USERNAME) && password.equals(PASSWORD)){
                result = String.format("<result><code>%d</code><msg>%s</msg></result>",1,username);
            }else{
                result = String.format("<result><code>%d</code><msg>%s</msg></result>",0,username);
            }
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
            result = String.format("<result><code>%d</code><msg>%s</msg></result>",3,e.getMessage());
        } catch (SAXException e) {
            e.printStackTrace();
            result = String.format("<result><code>%d</code><msg>%s</msg></result>",3,e.getMessage());
        }
        response.setContentType("text/xml;charset=UTF-8");
        response.getWriter().append(result);
    }
```

如果没有禁用 DTD 则存在 XXE 漏洞，以下代码为 XXE 防御代码

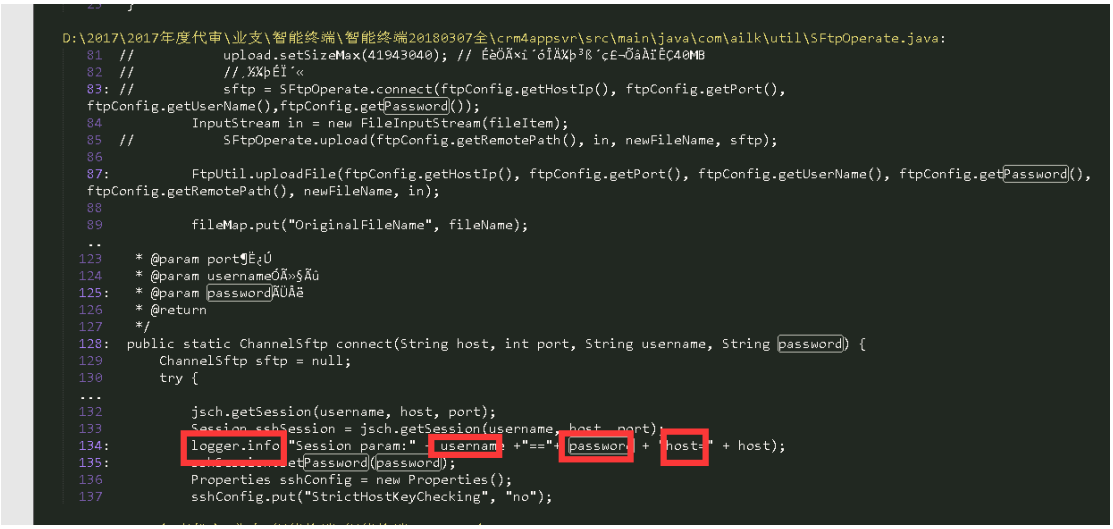
```
    dbf.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true); //禁用DTDs (doctypes),几乎可以防御所有xml实体攻击
//如果不能禁用DTDs,可以使用下两项，必须两项同时存在
    dbf.setFeature("http://xml.org/sax/features/external-general-entities", false);      //防止外部普通实体POC 攻击
    dbf.setFeature("http://xml.org/sax/features/external-parameter-entities",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



2.1.12URL 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 只是否为空，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

```
}
<%
}catch (Exception e){
%>
 alert('<%= e.getMessage()%>');
 window.close();
<%
}
```

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。

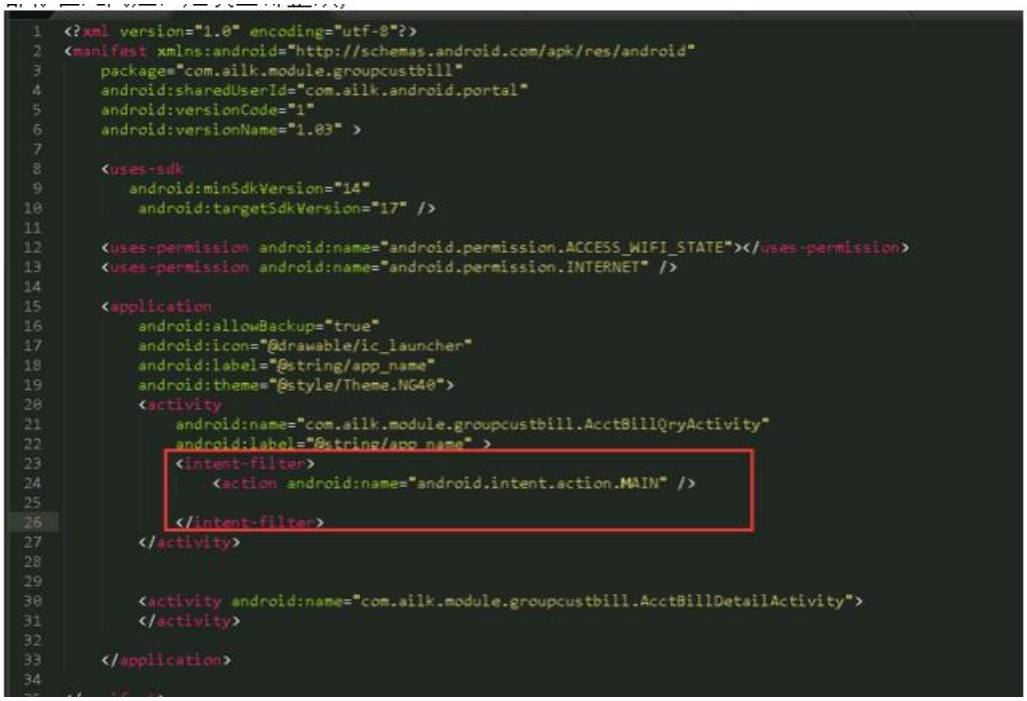
```
44
45  //读取输入流,并转换对象
46  InputStream in=request.getInputStream();
47  ObjectInputStream ois = new ObjectInputStream(in);
48  //恢复对象
49  ois.readObject();
50  ois.close();
51
```

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



3.1 其他漏洞审计方法

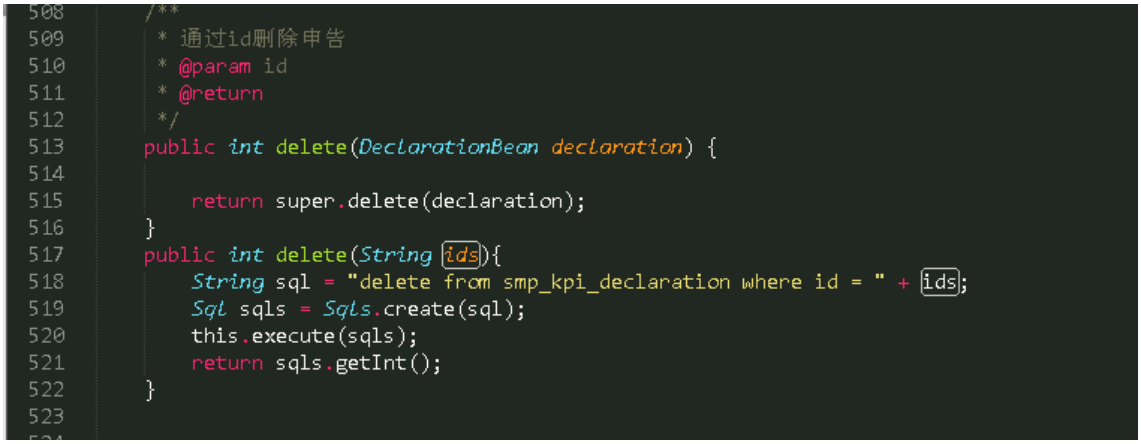
3.1.1CSRF

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     @At("/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.deleteb(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
```

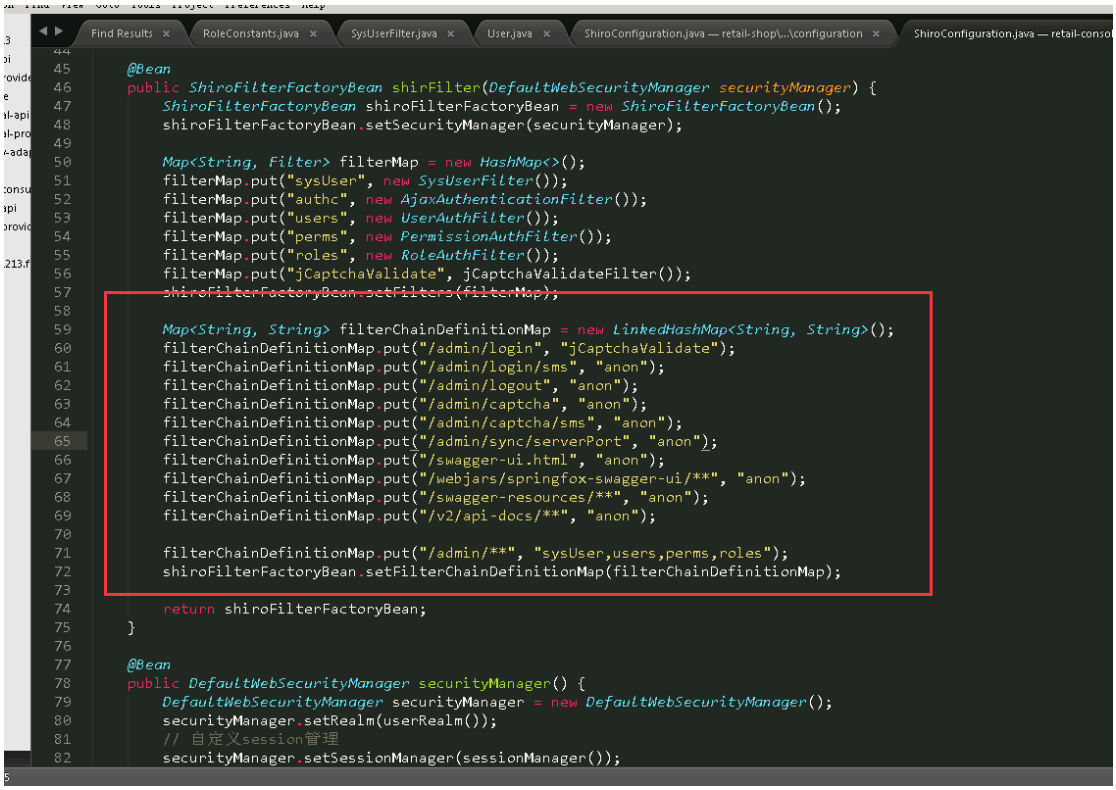
3.1.2Struts2 远程代码执行漏洞

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

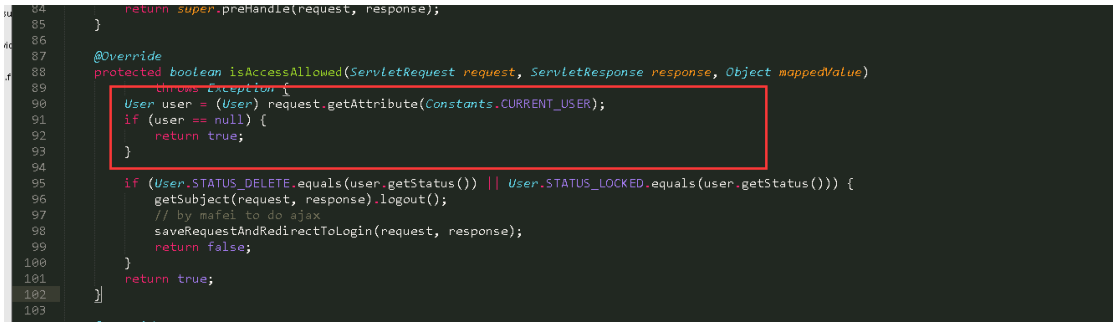
审计方法：重点关注用户操作请求时查看是否有对当前登陆用户权限做校验从而确定是否存在漏洞，有些厂商会使用一些主流的权限框架，例如 shiro ,spring security 等框架，那么需要重点关注框架的配置文件以及实现方法

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



```java
    @Bean
    public ShiroFilterFactoryBean shirFilter(DefaultWebSecurityManager securityManager) {
        ShiroFilterFactoryBean shiroFilterFactoryBean = new ShiroFilterFactoryBean();
        shiroFilterFactoryBean.setSecurityManager(securityManager);

        Map<String, Filter> filterMap = new HashMap<>();
        filterMap.put("sysUser", new SysUserFilter());
        filterMap.put("authc", new AjaxAuthenticationFilter());
        filterMap.put("users", new UserAuthFilter());
        filterMap.put("perms", new PermissionAuthFilter());
        filterMap.put("roles", new RoleAuthFilter());
        filterMap.put("jCaptchaValidate", jCaptchaValidateFilter());
        shiroFilterFactoryBean.setFilters(filterMap);

        Map<String, String> filterChainDefinitionMap = new LinkedHashMap<String, String>();
        filterChainDefinitionMap.put("/admin/login", "jCaptchaValidate");
        filterChainDefinitionMap.put("/admin/login/sms", "anon");
        filterChainDefinitionMap.put("/admin/logout", "anon");
        filterChainDefinitionMap.put("/admin/captcha", "anon");
        filterChainDefinitionMap.put("/admin/captcha/sms", "anon");
        filterChainDefinitionMap.put("/admin/sync/serverPort", "anon");
        filterChainDefinitionMap.put("/swagger-ui.html", "anon");
        filterChainDefinitionMap.put("/webjars/springfox-swagger-ui/**", "anon");
        filterChainDefinitionMap.put("/swagger-resources/**", "anon");
        filterChainDefinitionMap.put("/v2/api-docs/**", "anon");

        filterChainDefinitionMap.put("/admin/**", "sysUser,users,perms,roles");
        shiroFilterFactoryBean.setFilterChainDefinitionMap(filterChainDefinitionMap);

        return shiroFilterFactoryBean;
    }

    @Bean
    public DefaultWebSecurityManager securityManager() {
        DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();
        securityManager.setRealm(userRealm());
        // 自定义session管理
        securityManager.setSessionManager(sessionManager());
```

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



```java
        return super.preHandle(request, response);
    }

    @Override
    protected boolean isAccessAllowed(ServletRequest request, ServletResponse response, Object mappedValue)
            throws Exception {
        User user = (User) request.getAttribute(Constants.CURRENT_USER);
        if (user == null) {
            return true;
        }

        if (User.STATUS_DELETE.equals(user.getStatus()) || User.STATUS_LOCKED.equals(user.getStatus())) {
            getSubject(request, response).logout();
            // by mafei to do ajax
            saveRequestAndRedirectToLogin(request, response);
            return false;
        }
        return true;
    }
```

其他过滤文件均只设置了 onAccessDaniad()方法

```java
import com.ai.iisc.es.core.base.util.MessageUtils;
import com.ai.iisc.es.core.web.converter.MessageConverterUtil;
import com.ai.iisc.es.utils.AjaxUtil;
import com.alibaba.fastjson.JSON;
import org.apache.shiro.web.filter.authz.PermissionsAuthorizationFilter;
import org.apache.shiro.web.util.WebUtils;
import org.springframework.http.HttpStatus;

import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class PermissionAuthFilter extends PermissionsAuthorizationFilter {
    /**
     * shiro认证perms资源失败后回调方法
     * @param servletRequest
     * @param servletResponse
     * @return
     * @throws IOException
     */
    @Override
    protected boolean onAccessDenied(ServletRequest servletRequest, ServletResponse servletResponse) throws IOException {
        if(AjaxUtil.isAjaxRequest((HttpServletRequest) servletRequest)){
            ExceptionInfo result = new ExceptionInfo("1102", MessageUtils.getMessage("1102"));
            HttpServletResponse httpServletResponse = WebUtils.toHttp(servletResponse);
            httpServletResponse.setCharacterEncoding("UTF-8");
            httpServletResponse.setContentType("application/json");
            httpServletResponse.setStatus(HttpStatus.FORBIDDEN.value());
            httpServletResponse.getWriter().write(JSON.toJSONString(MessageConverterUtil.autoWrapException(result)));
//            httpServletResponse.sendError(HttpStatus.FORBIDDEN.value());
            return false;
        } else {//如果是普通请求进行重定向
            return super.onAccessDenied(servletRequest,servletResponse);
        }
    }
}
```

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。

```jsp
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8"
    isELIgnored="false"%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jstl/core_rt"%>
<!DOCTYPE html>
<%
    // 获取session ，如果为空重定向到登录页面
    String username = (String) session.getAttribute("username");
    if (username == null) {
        response.sendRedirect(request.getContextPath() + "/index.jsp");

        return;
    }
%>
<%
    String loginName="";
    String feeYear="";
    String taskType="";
    String starttime="";
 %>
<html lang="zh-cn">
```

以这个年度服务费用编制功能为例，测试一下，代码如图所示：

```jsp
                    </a> <b class="arrow"></b>

                    <ul class="submenu">

                        <li id="fulccx" style="display:none"><a
                            href="/cmccmtf/CMCCMis3/toCreateProcess.action?p
                            rocessId=CMCCMIS_2017_000003_1.0&
                            loginName=<%=username%>"
                            target="_blank"> 年度服务费用编制</a></li>
                        <li id="tzlccx" style="display:none"><a
                            href="/cmccmtf/CMCCMis3/toCreateProcess.action?p
                            rocessId=CMCCMIS_2017_000002_1.0&
                            loginName=<%=username%>"
```

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

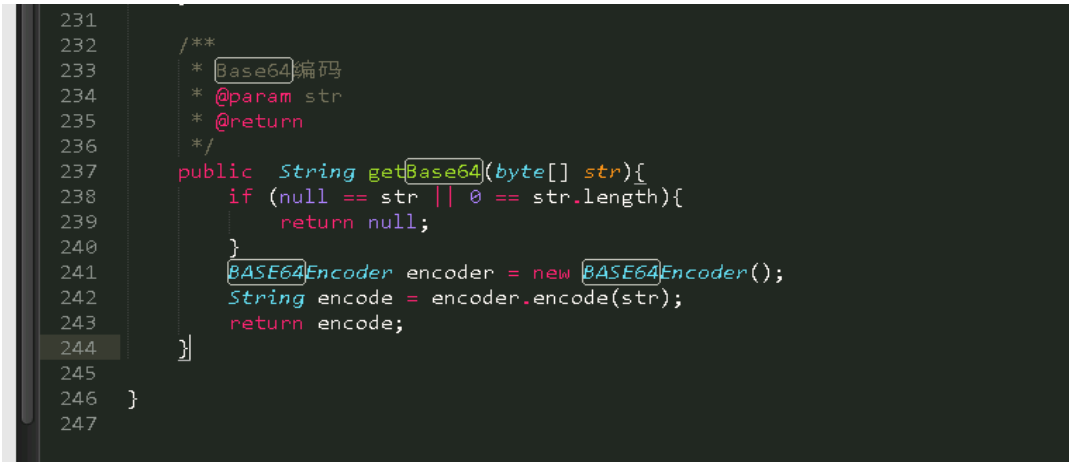
```
<session-config>
<session-timeout>15</session-timeout>
</session-config>
```

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      * Base64编码
234      * @param str
235      * @return
236      */
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：

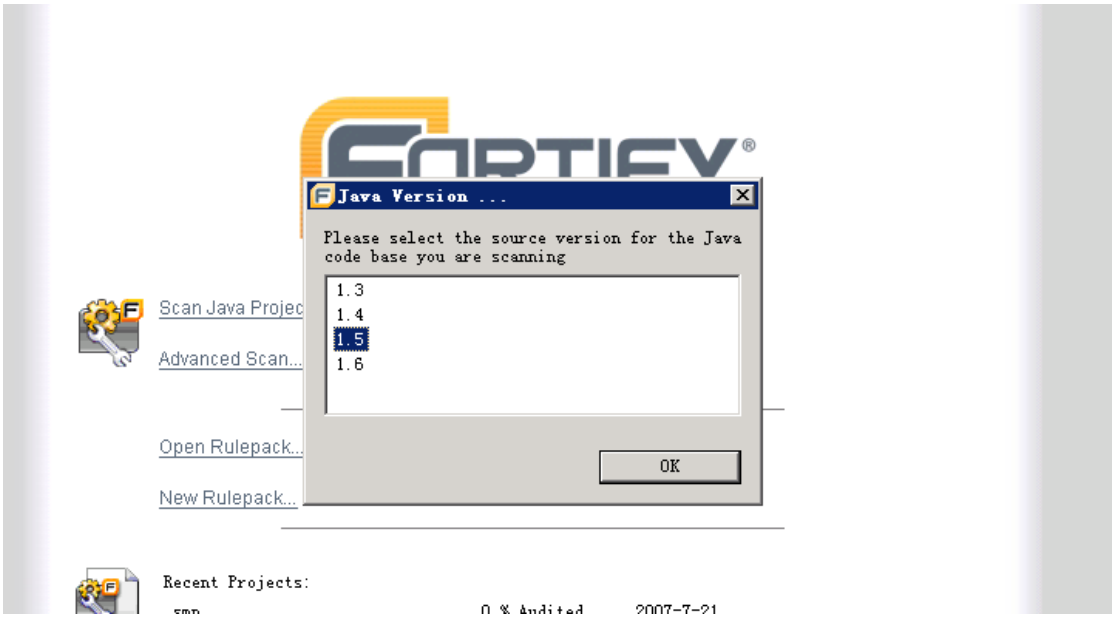
sourceanalyzer -scan -cp "lib/*.jar" "src/**/*.java" "web/**/*.jsp" -f result.fpr

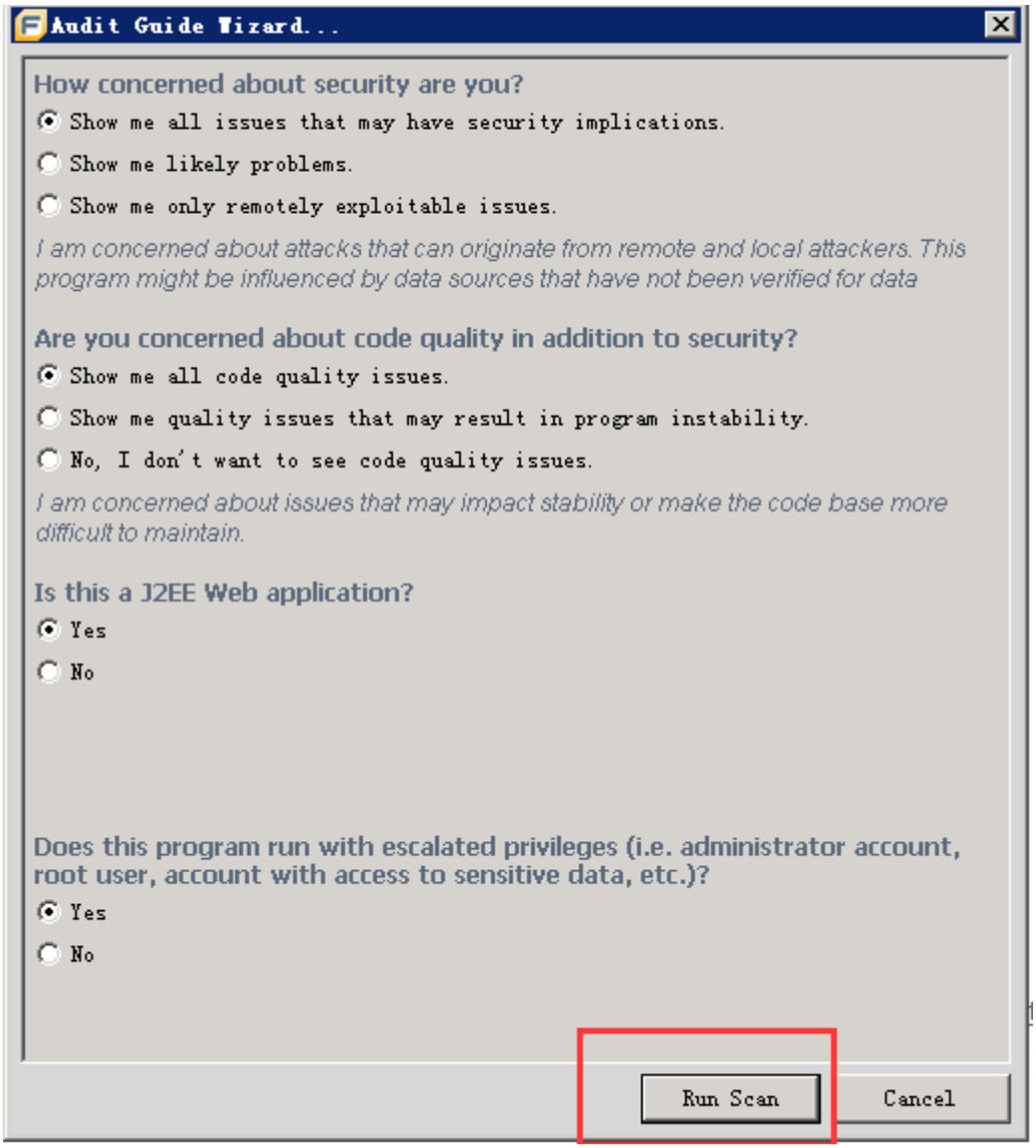
-cp 指定类库的路径，如果没有就不用这个选项

"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

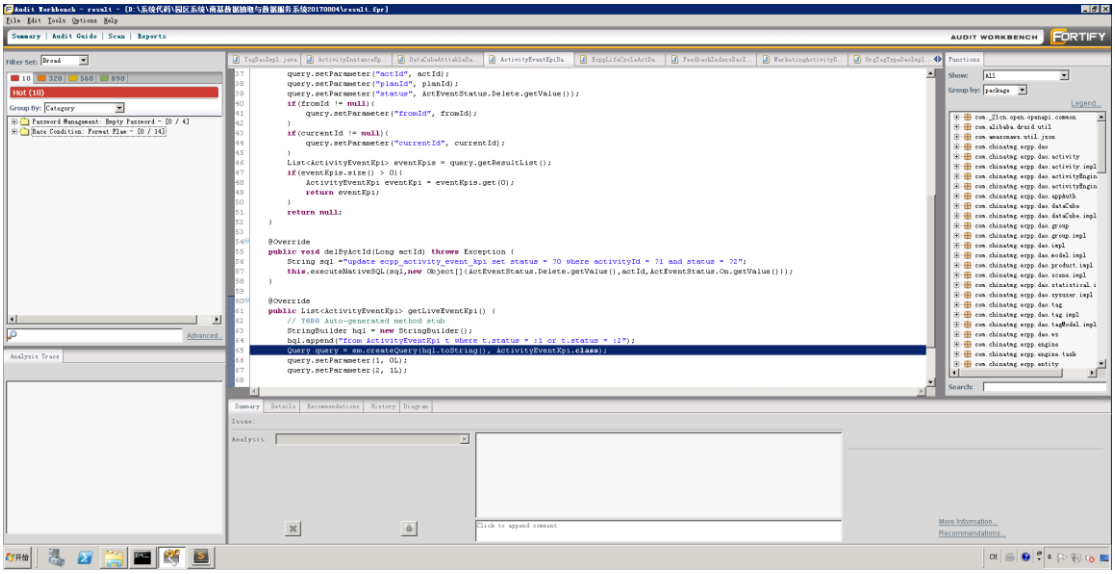
-f 指定扫描结果的输出文件为 result.fpr,扫描完后双击就可以通过 Fortity 查看了。

1.1.2 图形化界面

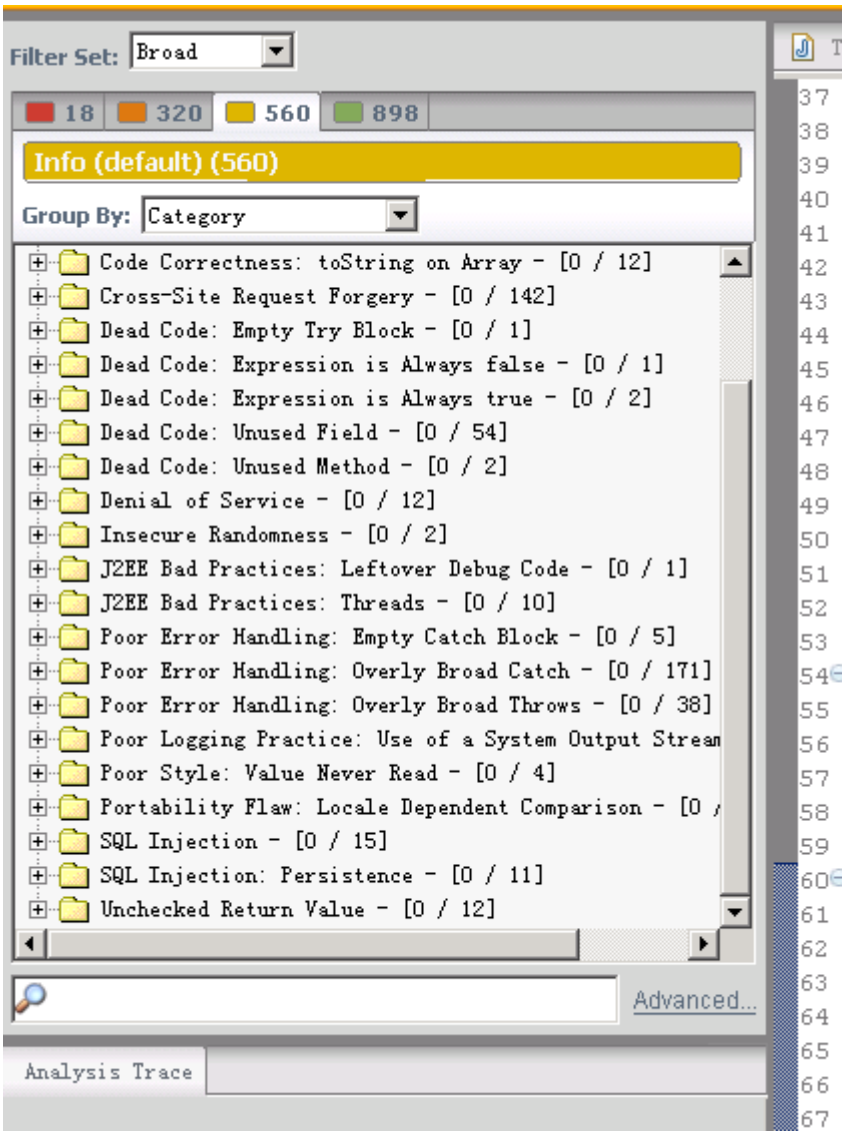

Java Version ...
Please select the source version for the Java code base you are scanning
1.3
1.4
1.5
1.6
OK
Scan Java Projec
Advanced Scan...
Open Rulepack...
New Rulepack...
Recent Projects:


Audit Guide Wizard...
How concerned about security are you?
Show me all issues that may have security implications.
Show me likely problems.
Show me only remotely exploitable issues.
I am concerned about attacks that can originate from remote and local attackers. This program might be influenced by data sources that have not been verified for data
Are you concerned about code quality in addition to security?
Show me all code quality issues.
Show me quality issues that may result in program instability.
No, I don't want to see code quality issues.
I am concerned about issues that may impact stability or make the code base more difficult to maintain.
Is this a J2EE Web application?
Yes
No
Does this program run with escalated privileges (i.e. administrator account, root user, account with access to sensitive data, etc.)?
Yes
No
Run Scan
Cancel

1.2 查看结果

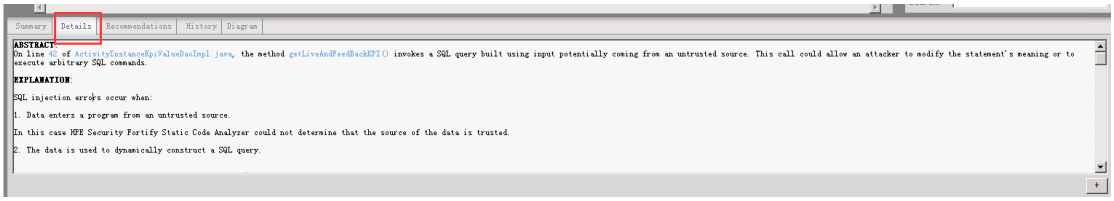


漏洞列表：


Filter Set: Broad
18
320
560
898
Info (default) (560)
Group By: Category
Code Correctness: toString on Array - [0 / 12]
Cross-Site Request Forgery - [0 / 142]
Dead Code: Empty Try Block - [0 / 1]
Dead Code: Expression is Always false - [0 / 1]
Dead Code: Expression is Always true - [0 / 2]
Dead Code: Unused Field - [0 / 54]
Dead Code: Unused Method - [0 / 2]
Denial of Service - [0 / 12]
Insecure Randomness - [0 / 2]
J2EE Bad Practices: Leftover Debug Code - [0 / 1]
J2EE Bad Practices: Threads - [0 / 10]
Poor Error Handling: Empty Catch Block - [0 / 5]
Poor Error Handling: Overly Broad Catch - [0 / 171]
Poor Error Handling: Overly Broad Throws - [0 / 38]
Poor Logging Practice: Use of a System Output Stream
Poor Style: Value Never Read - [0 / 4]
Portability Flaw: Locale Dependent Comparison - [0 /
SQL Injection - [0 / 15]
SQL Injection: Persistence - [0 / 11]
Unchecked Return Value - [0 / 12]
Advanced...
Analysis Trace

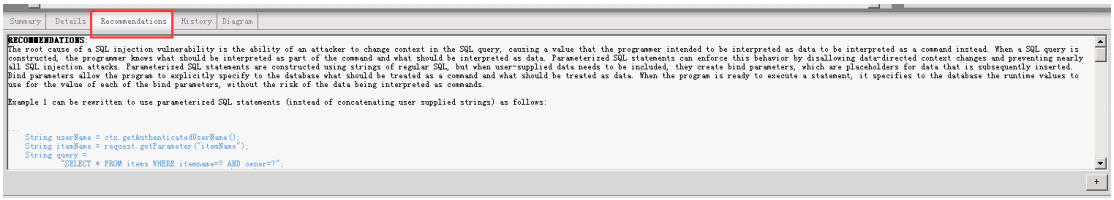
漏洞介绍：

Details



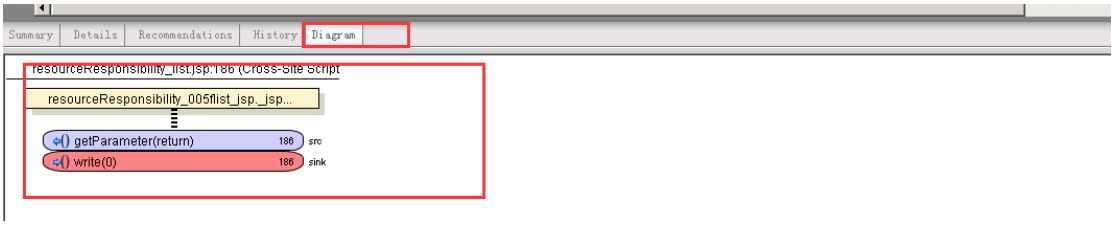
漏洞修复建议：

Recommendations



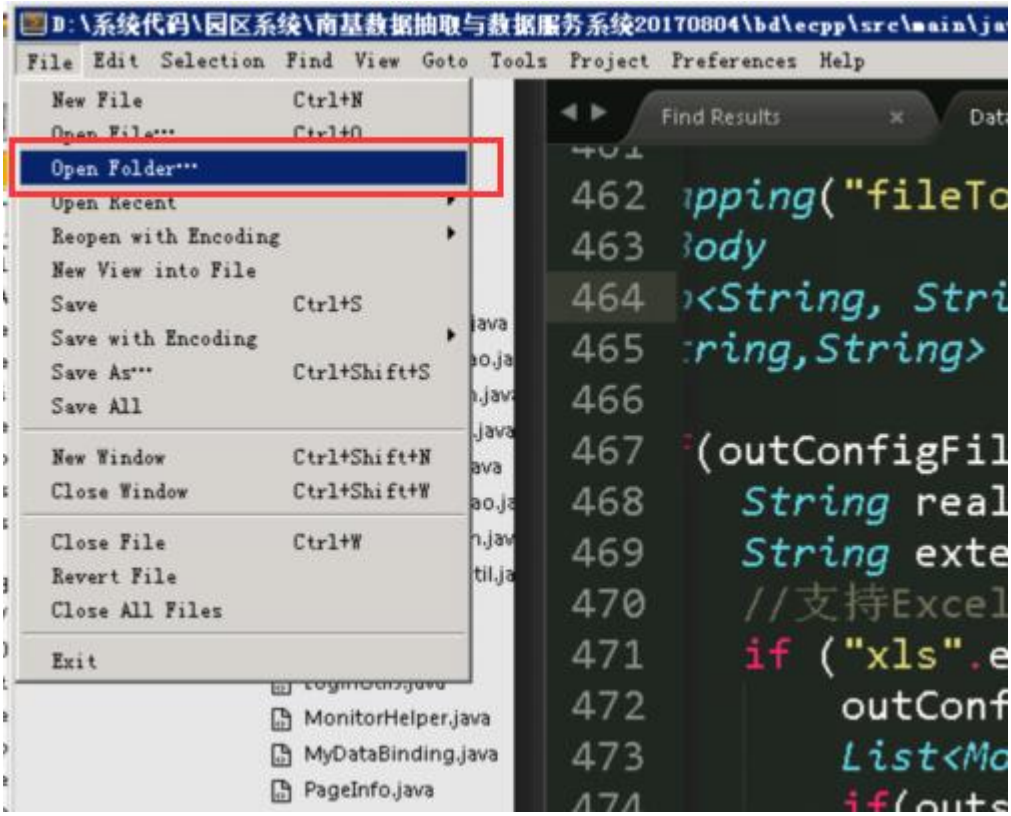
漏洞跟踪图：

Diagram



2. Sublime Text

2.1 打开项目相应目录：



2.2 全局搜索

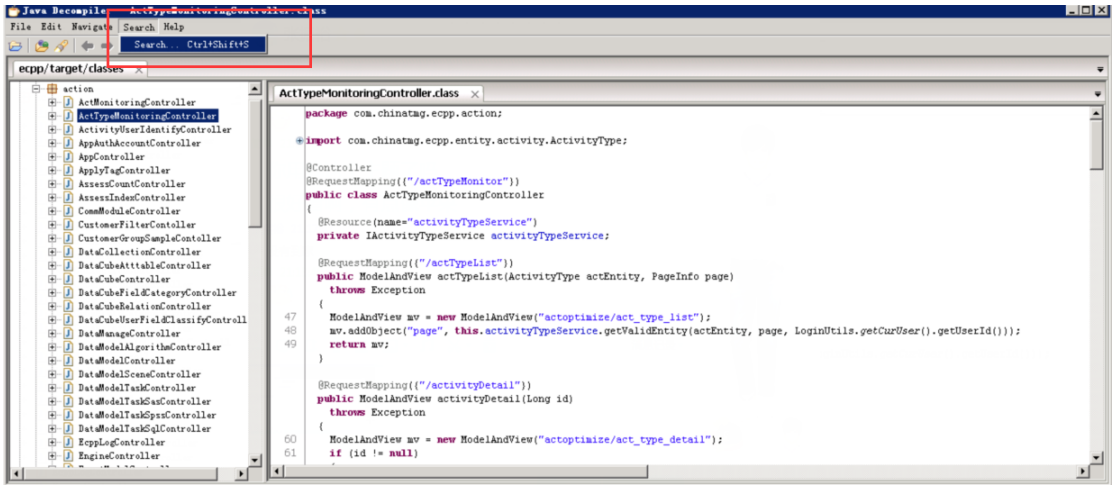
Ctrl + Shift + F 全局搜索

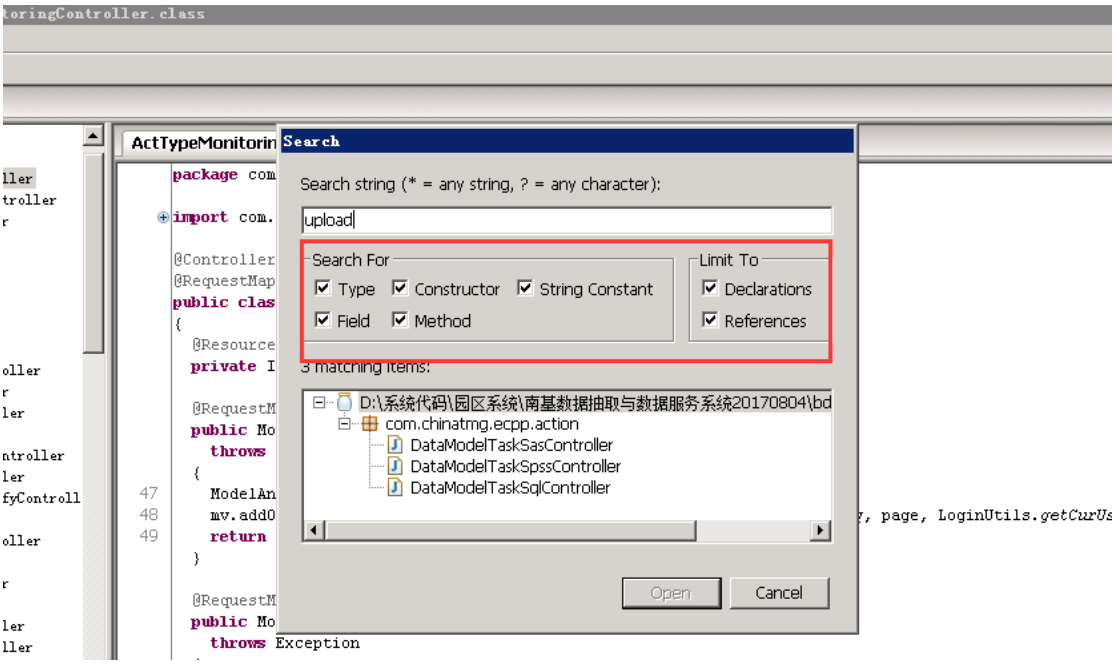
上面红框的几个图标可以设置是否大小写敏感，是否搜索整个单词。

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 和文件浏览器结合一起用最高效。